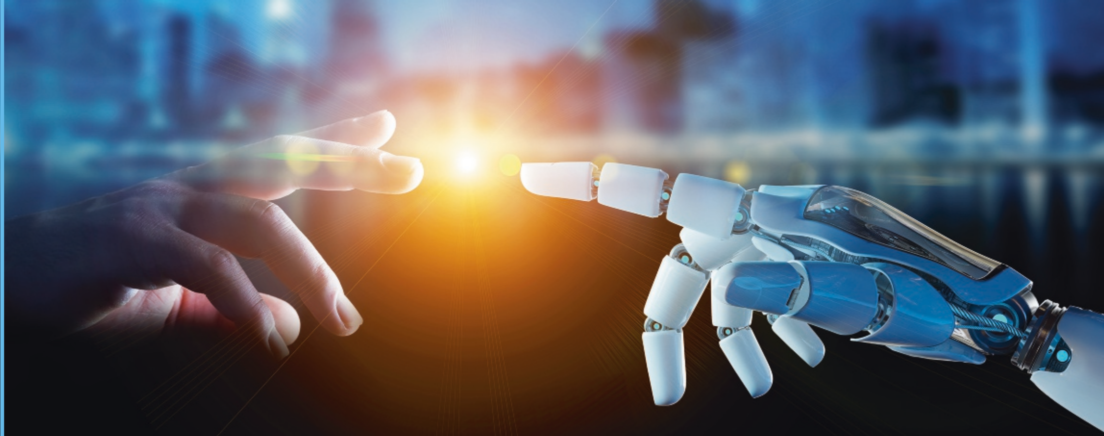


Gerhard Paaß · Dirk Hecker

Artificial Intelligence

What Is Behind the Technology of the Future?

Springer

Artificial Intelligence

Gerhard Paaß • Dirk Hecker

Artificial Intelligence

What Is Behind the Technology of the Future?

Springer

Gerhard Paaß
Fraunhofer Institute for Intelligent Analysis
and Information Systems (IAIS)
Sankt Augustin, Nordrhein-Westfalen,
Germany

Dirk Hecker
Institute Management
Fraunhofer Institute for Intelligent Analysis
and Information Systems (IAIS)
Sankt Augustin, Germany

ISBN 978-3-031-50604-8 ISBN 978-3-031-50605-5 (eBook)
https://doi.org/10.1007/978-3-031-50605-5

Translation from the German language edition: "Künstliche Intelligenz – Was steckt hinter der Technologie der Zukunft?" by Gerhard Paaß and Dirk Hecker, © Springer Fachmedien Wiesbaden GmbH 2021. Published by Springer Vieweg Wiesbaden. All Rights Reserved.

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2020, 2024
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Cover illustration: Illustration on the cover used with permission from sdecoret - stock.adobe.com

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

Preface to the German Edition

Artificial Intelligence (AI) is the buzzword of our time. As the central driver of digitization, it is fundamentally changing society, the economy and almost all other areas of life. The speed of this process is almost unprecedented compared to previous social and technical changes. Breakthroughs in the development of so-called deep artificial neural networks on high-performance computers have triggered this rapid technological development.

As early as the beginning of the 1950s, AI pioneer Alan Turing realized that computers could not be programmed by hand down to the last detail for many problems. There would have to be a more expeditious way to program computers: that method is Machine Learning. The techniques by which computers learn to improve their behavior from existing data are now so powerful that they are used in many places in everyday life and in our professional routines.

There are hardly any limits to the spectrum of application areas. Intelligent machines perceive the environment, make forecasts, give recommendations and make automated decisions. They relieve us of routine tasks and support us in responsible activities. The human-machine relationship is changing into a partnership model. Intelligent systems free us from routine work, expand our creative capabilities and increase our quality of life, but also lead to profound changes in society.

In this way, AI can make a contribution to tackling major societal challenges such as mobility and health. However, in order to be able to discuss the benefits, opportunities and risks of AI in a well-founded manner, users must understand how intelligent systems work in principle. To do this, he or she must grasp the most important concepts of the underlying Machine Learning technology. In addition, it is becoming clear in more and more areas of application that careful design is necessary to ensure that AI is in harmony with our societal values and our idea of sovereignty. Here, even non-computer scientists must be able to have an expert and informed say.

The purpose of this book is to clearly demonstrate the new possibilities of Machine Learning in different application areas, such as autonomous driving, medical diagnosis or the analysis of the meaning of language. The technical vocabulary but also concepts, methods and network architectures are explained with many graphics and pictures. Mathematical relationships are formulated when helpful, and always explained in a comprehensible way. It turns out that the methods used are composed of very simple operations, such as addition and multiplication. These gain their performance by being applied to very large sets of numbers and several times in succession. If one would like to understand the technical Chaps. 3, 4, 5, 6, 7, 8, and 9 in detail, a mathematical understanding at senior high school level is sufficient.

The book enables decision-makers, but also interested laypersons, to have a say in the design of intelligent systems and to better assess the impact of requirements. For data analysts, students, engineers, and researchers who are new to the field, this book is an ideal introduction to more advanced literature.

Preface to the English Edition

After the publication of the book, the authors received a very positive response from readers, who described it as a very easy-to-understand introduction to artificial intelligence and neural networks. Therefore, the authors and Springer Verlag decided to produce an English translation. The book describes the state of research up to early 2020, including transformers and GPT-3, and covers all major approaches used in AI models today. It is therefore suitable as an introduction to artificial intelligence and provides a basic understanding of current models such as ChatGPT.

In the meantime, a number of new Large Language Transformer models have been developed that are applicable to a wide variety of language tasks as well as to new media such as images, videos, DNA, or even control tasks. Because of their universality across most AI use cases, these approaches are called ‘Foundation Models’. These extensions are described in the book *Foundation Models for Natural Language Processing – Pre-trained Language Models Integrating Media* by G. Paaß, and S. Giesselbach, published Open Access by Springer Nature in 2023. It is recommended for in-depth study of large language models and requires a background knowledge of Machine Learning and Deep Learning provided in the current book.

Acknowledgements

This book was only made possible by the motivating and professionally stimulating environment of the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS in Sankt Augustin. We would like to thank all colleagues and people

from our personal environment who supported us in this book project—be it through professional discussions, proofreading of individual chapters, and helpful comments: Robert Babatz, Niklas Beck, Katharina Beckh, Sven Giesselbach, Harald Grund, Monika Hommes-Rüdiger, Birgit Kirsch, Franz Lutter, Dominik Paaß, Julia Paaß and Benjamin Schaarwächter. Mirco Lange and Dr. Henning Petzka have also worked through the entire book and provided valuable suggestions for improvements to the presentation and argumentation, many thanks for this.

A very special thank you goes to Dr. Angi Voss. Not only did she find encouraging words in difficult writing phases, but she also contributed quite significantly to this book with her many years of experience as an author, her driving questions and inspiring hints.

A lot of effort was also invested in the more than 400 charts, graphics, and photos. From the first ideas and graphics by Julia Paaß to the final and professional design by Svenja Niehues, who also made many content suggestions for the comprehensible design of the graphics. Many thanks for your perseverance and the long discussions together at the screen.

The book was translated into English by Margret Paaß. A special challenge was the translation of the text in graphics and pictures. Many thanks for her tireless efforts, which involved combining technical vocabulary with fluent translation.

The greatest thanks, however, are due to our families, who gave us the necessary freedom during the long period of writing. In particular, I, Gerhard Paaß, would like to thank my wife Margret Paaß, whose patience and encouragement played a major role in the success of this book and who was an indispensable help from the planning stage to translation and proofreading. And I, Dirk Hecker, would like to thank my family Katrin Berkler and Lara Hecker, who supported the work on this book even when I was on vacation and who also showed so much understanding in many other situations. Without your encouragement and support, we would not have been able to produce this book.

Thank you for all your support!

Sankt Augustin, Germany
January 2024

Gerhard Paaß
Dirk Hecker

Contents

1 What Is Intelligent About Artificial Intelligence? 1
1.1 Human Intelligence Has Many Dimensions 1
1.2 How To Recognize Artificial Intelligence 2
1.3 Computers Learn 3
1.4 Deep Neural Networks Can Recognize Objects 6
1.5 How To Understand Artificial Intelligence 8
1.6 The History of Artificial Intelligence 10
1.7 Summary 11
References 12

2 What Are the Capabilities of Artificial Intelligence? 15
2.1 Object Recognition in Images 16
2.1.1 Medical Diagnosis 17
2.1.2 Predicting the 3D Structure of Proteins 18
2.2 Speech Recognition 19
2.3 Machine Translation 20
2.4 Answering Natural Language Questions 22
2.5 Dialogs and Personal Assistants 24
2.6 Board Games 26
2.6.1 The Strategy Game Go 27
2.6.2 Artificial Intelligence Wins Against Five Poker Professionals 28
2.7 Video Games 29
2.7.1 Atari 2600 Game Console 29
2.7.2 Capture the Flag Quake 30
2.7.3 The Real-Time Strategy Game Dota2 31
2.8 Self-Driving Cars 32
2.8.1 Further Development of Self-Driving Cars 33
2.9 The Computer as a Creative Medium 35
2.9.1 Composing New Images 35
2.9.2 Inventing Stories 37

2.10 General Artificial Intelligence 38
2.11 Summary 39
References 39

3 Some Basic Concepts of Machine Learning 43
3.1 Main Types of Machine Learning 43
3.1.1 Supervised Learning 44
3.1.2 Unsupervised Learning 44
3.1.3 Reinforcement Learning 45
3.2 Programming and Learning 46
3.2.1 Models Transform an Input into an Output 46
3.2.2 Algorithms Process a List of Instructions Step by Step 48
3.2.3 A Learning Problem: The Recognition of Digits 48
3.2.4 Vectors, Matrices and Tensors 49
3.3 Learning a Relationship 51
3.3.1 Scheme for Learning: Model, Loss Function and Optimization 51
3.3.2 Detailed Process of Learning 52
3.4 A Simple Model: Logistic Regression 54
3.4.1 Calculation of a Score 54
3.4.2 The Simultaneous Calculation of All Scores 56
3.4.3 Affine Transformation 57
3.4.4 The Softmax Function Generates a Probability Vector 58
3.4.5 The Logistic Regression Model 59
3.5 Measuring Model Performance 60
3.5.1 A Criterion of Model Performance: The Likelihood of Complete Training Data 60
3.5.2 How to Measure Learning Success: The Loss Function 61
3.5.3 Illustration for Two Classes and Two Input Features 62
3.6 Optimization, or How to Find the Best Parameter Values 63
3.6.1 The Gradient Indicates the Direction of the Steepest Ascent 65
3.6.2 The Gradient for Several Dimensions 65
3.6.3 The Gradient of the Loss Function 67
3.6.4 Stepwise Minimization by Gradient Descent 68
3.6.5 The Learning Rate Sets the Length of an Optimization Step 69
3.6.6 Minibatch Gradient Descent Needs Less Computation 69
3.6.7 Applying the Model to New Data 71
3.6.8 Checking the Accuracy on the Test Set 72

3.6.9 Precision and Recall for Classes of Different Size 72
3.7 Summary 74
References 74

4 Deep Learning Can Recognize Complex Relationships 75

4.1 The XOR Problem Involves Interactions Between Features 75
4.2 Nonlinearities Create Curved Separating Planes 78
4.3 Deep Neural Networks Are Stacks of Nonlinear Layers 82
4.3.1 Vectors and Tensors Represent the Transformed Contents 83
4.4 Training of DNN with the Backpropagation Approach 85
4.5 Toolkits Facilitate DNN Specification and Training 88
4.5.1 Parallel Computations Accelerate DNN Training 88
4.5.2 Toolkits Simplify Work with DNN 89
4.6 How to Improve the Network? 91
4.6.1 Iterative Model Construction Using the Validation Set 91
4.6.2 Underfitting and Overfitting Lead to Higher Errors 92
4.6.3 An Example of Overfitting 93
4.6.4 Regularization Procedures Reduce Overfitting Errors 94
4.6.5 Penalizing Large Parameter Values Reduces Abrupt Output Changes 95
4.6.6 Dropout Disables Parts of the Network 96
4.6.7 Batch Normalization Avoids Extreme Values of Hidden Vectors 97
4.6.8 Mathematical Proof: Stochastic Gradient Descent Finds Well Generalizing DNN 98
4.7 Different Applications Require Different Networks Structures 98
4.7.1 Multilayer Feedforward Network 99
4.7.2 Convolutional Neural Network (CNN) 100
4.7.3 Recurrent Neural Network (RNN) 100
4.7.4 Reinforcement Learning Network 100
4.7.5 Generative Adversarial Network (GAN) 101
4.7.6 Autoencoder Networks Produce a Compressed Representation 101
4.7.7 Architectures for Specific Media and Application Areas 101
4.8 The Design of a Deep Neural Network Is a Search Process 102
4.8.1 Selection of the Hyperparameters of the Network 103
4.8.2 The Standard Model Search Process Leads to Better Models 104
4.8.3 Automatic Search of Model Architectures and Hyperparameters 106

4.9 Biological Neural Networks Work Differently 108
4.10 Summary and Trends 110
References 111

5 Image Recognition with Deep Neural Networks 113
5.1 What Does Image Recognition Actually Mean? 113
5.1.1 Types of Object Recognition in Images 113
5.1.2 Inspirations from Biology 114
5.1.3 Why Is Image Recognition Difficult? 116
5.2 The Components of a Convolutional Neural Network 116
5.2.1 Convolutional Kernels Analyze Small Image Areas 116
5.2.2 Different Kernels in a Convolution Layer Compute Many Features 119
5.2.3 The Pooling Layer Selects Most Important Feature Values 121
5.3 A Simple Convolutional Neural Network for Digit Recognition 122
5.4 ImageNet Competition Boosts Method Development 123
5.5 Advanced Convolutional Neural Networks 126
5.5.1 AlexNet Successfully Uses GPUs for Training 126
5.5.2 ResNet Facilitates Optimization by Residual Connections 126
5.5.3 DenseNet Employs Additional Residual Connections 129
5.5.4 Transformed Images Improve ResNeXt Training 129
5.6 Analysis of CNN Results 129
5.6.1 Individual Kernels Respond To Features of Different Types and Sizes 129
5.6.2 Similar Images Correspond To Neighboring Hidden Vectors 132
5.7 Transfer Learning Reduces the Need for Training Data 132
5.8 Localization of Objects in an Image 135
5.8.1 Object Localization by Rectangles 135
5.8.2 Pixel-Precise Localization of Class Objects 136
5.8.3 Max-Unpooling Assigns Values To an Enlarged Field 138
5.8.4 U-Net Detects Objects and Then Finds the Associated Pixels 139
5.9 3D Reconstruction of a Scene 141
5.10 Human Faces Can Be Matched with High Accuracy 141
5.11 Assessing the Accuracy of Model Predictions 144
5.11.1 Uncertainty of Model Predictions 144
5.11.2 Bootstrap Generates a Set of Plausible Models 145
5.11.3 Bayesian Neural Networks 146

5.12 Reliability of Image Recognition ... 149
5.12.1 Influence of Image Distortions ... 149
5.12.2 Targeted Construction of Misclassified Images ... 150
5.13 Summary and Trends ... 153
References ... 154

6 Capturing the Meaning of Written Text ... 157
6.1 How to Represent the Meaning of Words by Vectors? ... 159
6.1.1 The Concept of Embedding Vectors ... 161
6.1.2 Computation of Embedding Vectors with Word2vec ... 163
6.1.3 Softmax Function Approximation Reduces Computation ... 165
6.2 Properties of Embeddings Vectors ... 166
6.2.1 Nearest Neighbors of Embeddings Have Similar Meanings ... 166
6.2.2 Differences Between Embeddings Express Relations ... 168
6.2.3 FastText Uses N-Grams of Letters ... 170
6.2.4 StarSpace Creates Embeddings for Other Objects ... 171
6.3 Recurrent Neural Networks for Sequence Modeling ... 171
6.3.1 Recurrent Neural Networks as Language Models ... 173
6.3.2 Training of Recurrent Neural Networks ... 175
6.3.3 The Properties of RNN Gradients ... 176
6.4 The Long Short-Term Memory (LSTM) is a Long-Term Memory ... 178
6.4.1 Controling Memory Operations by Gates ... 178
6.4.2 LSTMs with Multiple Layers ... 181
6.4.3 Applications of the LSTM ... 181
6.4.4 Bidirectional LSTM Networks for Word Property Prediction ... 183
6.4.5 Visualization of Recurrent Neural Networks ... 185
6.5 Transformation of one Sequence into Another Sequence ... 186
6.5.1 Sequence-to-Sequence Networks for Translation ... 187
6.5.2 Attention: Improving Translation by Recourse to Input Words ... 191
6.5.3 Attention Generated Translation Results ... 193
6.6 BERT: A Model for the Representation of Meanings ... 196
6.6.1 Tokenization to Limit Vocabulary Size ... 196
6.6.2 Self-Attention Analyzing the "Correlation" of Different Tokens ... 198
6.6.3 BERT Computes Contextual Embeddings by Self-Attention ... 200
6.6.4 BERT Prediction Tasks for Unsupervised Pre-training ... 202

6.7 Transfer Learning with BERT Language Models ... 204
  6.7.1 Semantic Classification Tasks ... 205
  6.7.2 Question Answering ... 206
  6.7.3 Extraction of World Knowledge ... 208
  6.7.4 Using BERT for Web Search ... 211
6.8 Transformer Translation Models ... 212
  6.8.1 Cross-Attention Exploits the Input-Output "Correlation" ... 212
  6.8.2 Transformer Architecture Uses Self- and Cross-Attention ... 214
  6.8.3 Training the Transformer for Language Translation ... 216
  6.8.4 Translation Results for the Transformer Model ... 218
  6.8.5 Simultaneous Translation Requires a Time Delay ... 220
  6.8.6 Transfer Learning for Translation Models ... 222
6.9 The Description of Images by Text ... 224
  6.9.1 Explanation of DNN Forecasts ... 227
  6.9.2 Explanations Are Necessary ... 227
  6.9.3 Global Explanatory Models ... 228
  6.9.4 Local Explanatory Models ... 229
6.10 Reliability of Text Understanding ... 231
  6.10.1 Robustness in Case of Text Errors and Domain Change ... 231
  6.10.2 Vulnerability to Malicious Modification of Inputs ... 232
6.11 Summary and Trends ... 233
References ... 235

7 Understanding Spoken Language ... 239
7.1 Speech Recognition ... 239
  7.1.1 Why Is Speech Recognition Difficult? ... 240
  7.1.2 How to Represent Speech Signals in the Computer? ... 240
  7.1.3 Assessing Speech Recognition Accuracy ... 242
  7.1.4 The History of Speech Recognition ... 244
7.2 Deep Sequence-to-Sequence Models ... 246
  7.2.1 List-Attend-Spell Generates a Sequence of Letters ... 246
  7.2.2 Sequence-to-Sequence Model for Words and Syllables ... 249
7.3 Convolutional Neural Networks for Speech Recognition ... 249
  7.3.1 CNN Models ... 250
  7.3.2 Combined Models ... 253
7.4 Lip Reading ... 254
7.5 Generating Spoken Language from Text ... 255
  7.5.1 WaveNet with Dilated Convolution for Long Dependencies ... 255
  7.5.2 The Tacotron Generates a Spectrogram ... 258

7.6 Dialogs and Voice Assistants 259
7.7 Gunrock: An Extended Alexa Voice Assistant 261
7.7.1 Language Understanding 261
7.7.2 Dialog Management 263
7.7.3 Response Generation 264
7.7.4 Testing the Voice Assistant 264
7.8 Analysis of the Content of Videos 265
7.8.1 Tasks of Video Content Analysis 266
7.8.2 Training Data for Classification of Videos by Activities 266
7.8.3 Convolution Layers for Video Content Recognition 267
7.8.4 Accuracy of Video Classification 270
7.8.5 The Generation of Subtitles for Videos 271
7.9 Reliability of Processing Spoken Language 274
7.9.1 The Effect of Noise and Other Distortions on Speech Recognition 274
7.9.2 Adversarial Attacks on Automatic Speech Recognition 274
7.10 Summary 276
References 277

8 Learning Optimal Policies 281
8.1 Some Basic Definitions 283
8.2 Deep Q-Network 286
8.2.1 Policy to Maximize the Sum of Rewards 286
8.2.2 A Small Navigation Task 286
8.2.3 Discounted Future Reward Encourages Fast Solutions 287
8.2.4 The Q-Function Evaluates State-Action Pairs 287
8.2.5 The Bellman Equation Relates Q-Values to Each Other 288
8.2.6 Approximation of the Q-Function by a Deep Neural Network 289
8.2.7 Q-Learning: Training a Deep Q-Network 290
8.3 Application of Q-Learning to Atari Video Games 293
8.3.1 Definition of the Game State in Atari Games 293
8.3.2 Architecture of the Atari Q-Network 294
8.3.3 Training of Atari Q-Networks 294
8.3.4 Evaluation of Atari Q-Networks 295
8.4 Policy Gradients for Learning Stochastic Policies 297
8.4.1 Need for Policies with Random Elements 297
8.4.2 Direct Optimization of a Policy by Policy Gradients 298
8.4.3 Extensions of the Policy Gradient: Actor-Critic and Proximal Policy Optimization 301
8.4.4 Application to Robotics and the Go Board Game 302
8.4.5 Application to the Dota2 Real-Time Strategy Game 303

8.5 Self-Driving Cars ... 305
8.5.1 Sensors of Self-Driving Cars ... 305
8.5.2 Functionality of an Agent for Autonomous Driving ... 308
8.5.3 Fine-Tuning Through Simulation ... 309
8.5.4 Reliability of Reinforcement Learning ... 312
8.5.5 Simulation-Trained Models Difficult to Transfer ... 312
8.5.6 Adversarial Attacks on Reinforcement Learning Models ... 313
8.6 Summary and Trends ... 314
References ... 315

9 Creative Artificial Intelligence and Emotions ... 319
9.1 Image Creation with Generative Adversarial Networks (GAN) ... 320
9.1.1 Forger and Art Expert ... 320
9.1.2 Generator and Discriminator ... 320
9.1.3 Optimization Criteria for Generator and Discriminator ... 321
9.1.4 Results of Generative Adversarial Networks ... 322
9.1.5 Interpolation Between Images ... 325
9.1.6 Transformation of Images ... 326
9.1.7 Transformation of Images Without Training Pairs ... 329
9.1.8 Creative Adversarial Network ... 330
9.1.9 Generating Images from Text ... 332
9.1.10 GAN-Generated Persons in Three Dimensions ... 333
9.2 Composing Texts ... 335
9.2.1 Automatic Reporter: Convert Data to Newspaper Reports ... 335
9.2.2 Generating Longer Stories ... 335
9.2.3 GPT-2 Invents Complex Stories ... 337
9.3 Compose Music Automatically ... 343
9.3.1 MuseNet Composes Mixtures of Classic and Pop ... 344
9.3.2 The Music Transformer Invents Piano Pieces ... 346
9.4 Emotions and Personality ... 347
9.4.1 A XiaoIce Dialog ... 348
9.4.2 The Goal: Encourage People to Keep Talking ... 349
9.4.3 Architecture of XiaoIce ... 350
9.4.4 Number of User Responses as Optimization Criterion ... 352
9.4.5 Emotional Empathy and Support ... 353
9.4.6 Summary and Trends ... 356
References ... 359

10 AI and Its Opportunities, Challenges and Risks ... 363
10.1 Opportunities for Economy and Society ... 366
10.1.1 Smart Home, My House Takes Care of Me ... 366

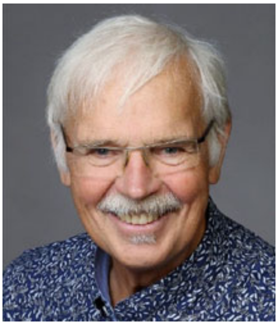
10.1.2 Diagnosis, Therapy, Care and Administration in Medicine 369
10.1.3 Machine Learning in Industrial Applications 374
10.1.4 Further Areas of Application for AI 377
10.2 Economic Impacts and Interrelationships 381
10.2.1 The Monetization of Data 382
10.2.2 The New Digital Service World: AI as a Service 386
10.2.3 Large Companies as Drivers of AI 388
10.2.4 Impact on the Labor Market 392
10.3 Challenges for the Society 397
10.3.1 Challenges of AI in Medicine 399
10.3.2 Orwell's 1984 Vers. 2.0: AI as a Surveillance Tool 400
10.3.3 War of the Machines 403
10.3.4 Artificial General Intelligence 404
10.4 Methodological Challenges 406
10.4.1 Combination of Data and Uncertain Reasoning 407
10.4.2 Fast and Slow Thinking 408
10.5 Building Trust in AI 412
10.5.1 How to Build Trustworthy AI Systems? 414
10.5.2 How to Test Deep Neural Networks? 415
10.5.3 Is Self-Determined, Effective Use of an AI System Possible? 417
10.5.4 Does the AI System Treat all Affected Parties Fairly? 418
10.5.5 Are the Functioning and Decisions of AI Comprehensible? 419
10.5.6 Are AI Systems Secure from Attack, Accident, and Error? 421
10.5.7 Do AI Components Work Reliably and Perform Robustly? 421
10.5.8 Does AI Protect Privacy and Other Sensitive Information? 422
10.5.9 The Challenges for an AI Seal of Approval 423
10.6 Summary 424
References 425

A Appendix 429
A.1 Mathematical Notation 429
A.2 Glossary 430
A.3 List of Images and Their Sources 446

References 461

Index 467

About the Authors



Dr. Gerhard Paaß was born on August 10, 1949 in Langenfeld near Cologne, Germany, and has held high-ranking scientific positions in the field of Artificial Intelligence as a mathematician and computer scientist since 1976. After completing his doctorate in 1985 on the topic of “Prognosis and Asymptotics of Bayesian Models” at the University of Bonn, Dr. Paaß worked in the context of numerous research stays at universities abroad (China, USA, Australia, Japan). Among them were renowned institutions such as UC Berkeley in California and the University of Technology in Brisbane. As an employee of the Gesellschaft für Mathematik und Datenverarbeitung (GMD), today’s Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Dr. Paaß was and still is a sought-after reviewer and conference leader at international conferences, e.g. as a member of the editorial board of the journal “International Journal of Uncertainty, Fuzziness and Knowledge-based Systems” or as a workshop leader at the “Conference on Knowledge Discovery and Data Mining” 2010 in Washington on the topic of analyzing texts for security applications. Dr. Paaß is the author of numerous publications and has received several best paper awards in the field of AI. In addition, he has been active as a lecturer for many years and, within the framework of the Fraunhofer Big Data and Artificial Intelligence Alliance, has played a very significant role in defining the new job description of the Data Scientist and successfully establishing it in Germany as well. As Lead Scientist at Fraunhofer IAIS, Dr. Paaß has contributed to the development of numerous curricula in this field.

Dr. Dirk Hecker was born in Cologne on August 29, 1976. He is deputy director of the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS and managing director of the Fraunhofer Big Data and Artificial Intelligence Alliance, the largest alliance of the Fraunhofer-Gesellschaft with more than 30 member institutes. In addition, Dr. Hecker is a member of the board of directors of the Fraunhofer Academy, which is a further education institution of the Fraunhofer-Gesellschaft aimed at specialists and executives of technology-driven companies. Dr. Hecker has many years of experience in leading research and industrial projects in the field of data mining and Machine Learning. His current work focuses on Big Data Analytics, Predictive Analytics, Deep Learning and Explainable AI. He studied geo-informatics at the Universities of Cologne and Bonn and received his PhD from the University of Cologne. Dr. Hecker is the author of numerous publications in the above-mentioned fields and is active as a subject matter expert and auditor on the topic of Artificial Intelligence in a wide variety of committees.

Chapter 1
What Is Intelligent About Artificial Intelligence?

Abstract Recently, the term Artificial Intelligence (AI) came into the focus of public discussion. An Artificial Intelligence system is supposed to be able to perceive its environment and behave intelligently, similar to humans. However, this definition is imprecise because the term “intelligence” is difficult to delineate. Therefore, this chapter discusses the individual dimensions of AI. Most AI systems are tasked with associating an input (e.g., an image) with an output (e.g., a class of images objects). Inputs and outputs are represented by sets of numbers. This mapping is not manually programmed, but successively adapted and trained based on observations and data. This process is also called “learning”.

Recently, the term Artificial Intelligence (AI) has been on everyone’s lips. Press, parliaments and governments regard AI as a crucial driver for the country’s further economic development. The German and other governments have therefore adopted a massive program to promote AI (Álvarez 2018). Experts from the consulting firm McKinsey estimate that AI will generate a global sales volume of around 12 trillion euros by 2030 (Tung 2018).

“Artificial Intelligence is the ability of a computer or computer-controlled robot to solve tasks normally performed by intelligent beings” (Copeland 2019). The system should be able to behave intelligently and learn on its own, similar to a human. However, this definition is imprecise because the term “intelligence” is difficult to delineate.

1.1 Human Intelligence Has Many Dimensions

There are a number of different descriptions of human intelligence. Gardner (1983) has developed a theory of multiple intelligences that lists eight dimensions of intelligence (Fig. 1.1). Movement intelligence is the ability to feel and move one’s body in a controlled manner. Figurative-spatial intelligence enables the recognition of images and the grasp of spatial relationships. Linguistic intelligence includes

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_1

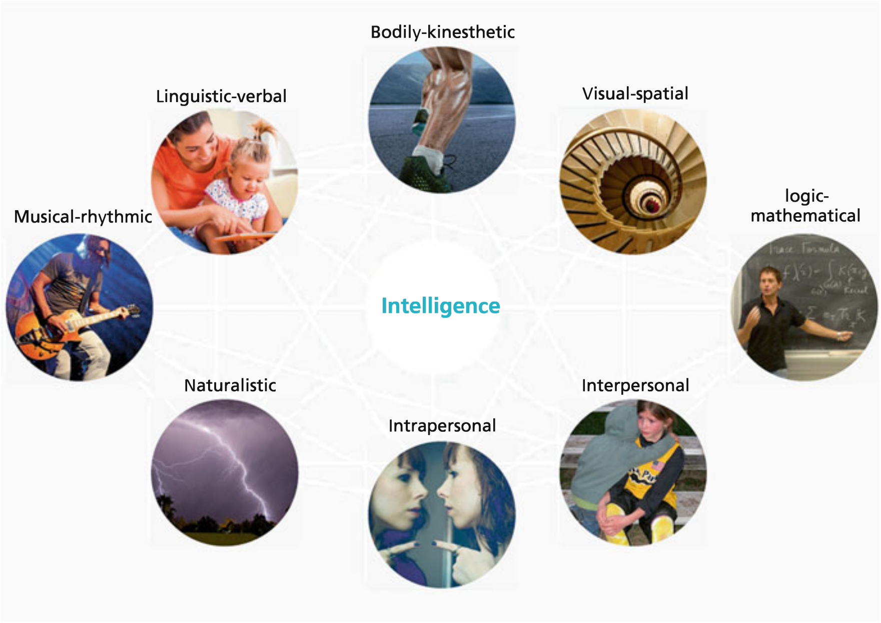


Fig. 1.1 The dimensions of human intelligence according to Gardner (1983). Image credits in Appendix A.3

the understanding of language and the appropriate verbal phrasing of matters. Logical-mathematical intelligence enables the analysis and solution of logical problems. Musical intelligence is required for listening to music with appreciation and for making music. Naturalistic intelligence includes the ability to observe, distinguish, and recognize nature, and to develop sensitivity for natural phenomena. Interpersonal or emotional intelligence is the ability to understand and predict the intentions, feelings, and motives of other people. Self-reflective intelligence includes the ability to recognize one's own moods, drives, motives, and feelings. It also includes an awareness of oneself and the capability to predict one's own behavior in new situations and motivate oneself to take action. We will see that AI is now applicable to many—but not all—of these dimensions.

1.2 How To Recognize Artificial Intelligence

To evaluate whether a computer system is intelligent, the British mathematician Alan Turing proposed a test procedure—the Turing test (Turing 1950). In the test, a human referee can communicate with two partners by exchanging text electronically and ask any questions: one partner is a human, the other a computer (Fig. 1.2). If,

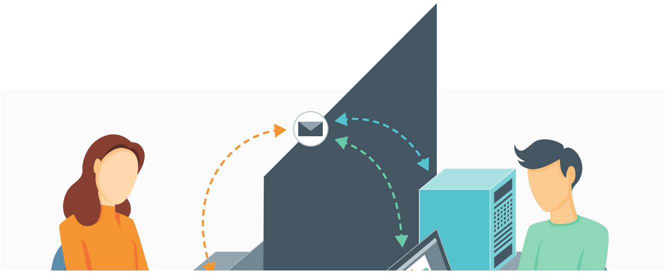


Fig. 1.2 In the Turing test, the referee on the left asks questions to partners he cannot see: a human and a computer on the right. He receives answers from both. If the referee cannot distinguish the computer from the human partner by their answers, the computer must also be intelligent

after asking many questions, the referee cannot decide from the answers which of the partners is the computer, the computer is said to be intelligent.

With regard to the dimensions of intelligence presented previously, however, the Turing test needs to be extended so that other dimensions (vision, movement, speech) can also be captured.

Many researchers have started to favor new test criteria that more closely examine how profound the computer system's understanding of a situation is. For example, the referee might talk to partners about a Netflix video. The question: "Why is this scene with Bill Murray funny?", for instance, would be more difficult for a computer to answer than "Tell me about your mother!".

Attempts were made early to manually program computers to exhibit intelligent behavior. Unfortunately, these approaches only achieved the desired success with severe limitations. As an alternative, the approach of developing a computer program capable of learning prevailed. This learning procedure trains the desired functionality using sample data. As a result, it is now possible to solve subtasks of AI satisfactorily. Examples are the diagnosis of diseases on the basis of symptoms or X-ray images, the transcription of spoken language into text or the recognition of objects in images.

1.3 Computers Learn

But what does "learning" mean for a computer system? Let's take the recognition of objects in images, e.g. a cat, as an example.

The computer receives the image of a cat (Fig. 1.3) as input. In the right part of Fig. 1.3 you can see an enlarged section of the image, from which it is clear that the image of the cat is a rectangle consisting of a number of small square color areas (pixels). Each of these pixels has a color, which can be characterized by the proportions of the three primary colors: red, green and blue. Thus, a pixel can be

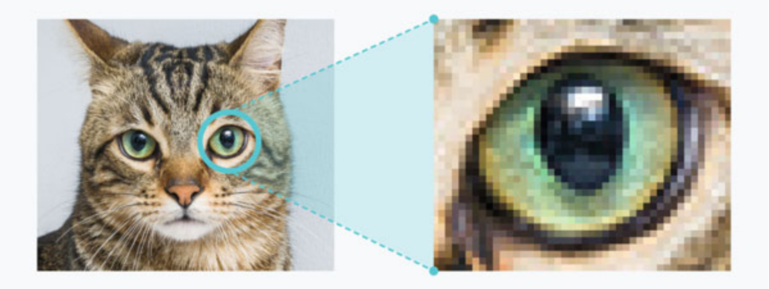


Fig. 1.3 Image of a cat and image section with individual pixels. Each pixel is described by three numbers, the color values for red, green and blue. Image credits in Appendix A.3

described by a triple of numbers and the whole image by a rectangular scheme of number triples.

The computer receives the image in the form of a rectangular scheme of number triples as input. The goal now is for the computer to be able to name the most important object in the image, in our case “cat”. This task is called object classification in images, a subtask of image recognition. So the computer is not told where in the image the object is that it should name.

Early approaches to this task attempted to first recognize given parts of the image objects, e.g., corners, edges, lines, and surfaces. The larger objects (e.g., eye) were then reconstructed as connections of the smaller parts. However, this approach did not generate good results.

Recently, methods have been tested in which the computer no longer uses human-defined features (corners, edges, lines and color blobs). Instead, it automatically selects important features, recognizes them in the image, and uses them to classify objects. To do this, however, it needs a large number of sample images in which the target image object (e.g., cat) occurs, as well as sample images in which the target does not occur. Only in this way can the computer recognize the similarities and differences between the objects. and the corresponding features.

Thus, the basis of object classification is a large set of examples, which consist of the input (image) and the corresponding output, the object class (e.g. monkey, cat, . . .) (Fig. 1.4). The set of examples is called the “training set” or “training data”. The elements of the training set are also called training examples.

The task of the computer is now to analyze the set of examples and the corresponding object classes. Subsequently, the system has to develop a computing instruction by itself, with which the object classes of new objects can be predicted as well as possible. The determination of such a calculation rule is called “learning”. The situation is comparable to that of a toddler to whom the mother, as in Fig. 1.5, tells the names of objects in the picture book. In the process, the child learns how to distinguish and name the different objects.

Learning is defined as the process by which new or modified skills, knowledge content, or behavior patterns are acquired (De Houwer et al. 2013).

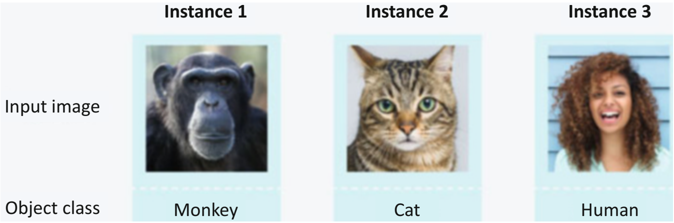


Fig. 1.4 Training data from different classes for an image recognition system. Each training example consists of an input image and the associated object class. A large number of training examples are required per class. Image credits in Appendix A.3

Fig. 1.5 A mother shows her child objects in a picture book. Image credits in Appendix A.3

Commonly, learning is understood as a profoundly human ability. Therefore, many people are unwilling to concede a computer program an ability to learn. However, animals can also learn, as many experiments from biology prove. In contrast to living organisms, learning in the field of AI is more akin to the term "training": Here, the system can acquire the ability to determine the appropriate outputs (e.g., object class) for given inputs. This does not mean that the system "learns by heart" the objects in the training set, but it can also assign the correct class to new images that have not yet been processed. This is the sense in which the term "learn" is used in this book.

There are a number of other verbs that are normally used in the context of humans, but also appear in the field of AI. These include "recognize," "know," etc. When humans perform these activities, it is always associated with human consciousness and emotions. In the field of AI, these aspects are completely excluded. This must always be taken into account when reading this book.

1.4 Deep Neural Networks Can Recognize Objects

Learning tasks, such as object classification in images, can now be performed by deep neural networks. As shown in Sect. 5.1.2, deep neural networks (DNNs) have structural similarities to information processing in the brain. They process the input in a number of successive layers, transforming the input data into more abstract features represented by packets of numbers. Each layer processes specific features of the scene—the higher the layer, the more complex the features. These features are selected and generated by the system itself. Figure 1.6 shows the features extracted in this way by Lee et al. (2011) for classifying an object as human. Finally, the desired results, e.g., the names of the objects, can be determined in a simple way from the features of the last layer.

However, the deep neural network can only recognize images if its parameters have been adjusted. The parameters are also a set of numbers—a number packet—which controls the properties of the DNN. Previously, the structure of the DNN and the count of numbers in the parameter number packet were specified by the designer of the network. The parameter number packet is initially filled with random number values. As shown in Fig. 1.7, the DNN in this state can neither recognize meaningful intermediate features nor identify the object in the image.

As discussed previously, the values of the parameter number package are adapted to a large set of training examples. These usually consist of the input (image) and the corresponding output, i.e. the class of the image object (e.g. monkey, cat, . . . , see Fig. 1.4). Usually hundreds of such training examples are required for each class. The computer now gradually adjusts the values of the parameter number packet so that the DNN outputs the correct class for each input image, if possible. In recent years, it has been possible to modify even millions of different parameter values

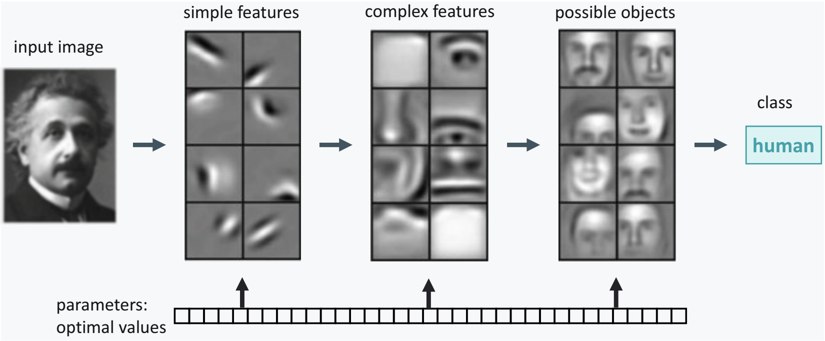


Fig. 1.6 A deep neural network (DNN) receives an input, e.g., the image of a person. From this image, simple features are extracted in the bottom layer, and more complex features are extracted in subsequent layers. The assignment to an object class takes place in the last layer. Image credits in Appendix A.3

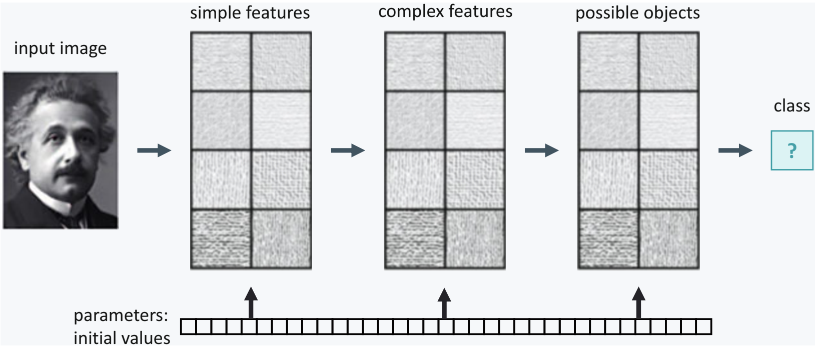


Fig. 1.7 Calculated intermediate features of the DNN and output classification at the beginning of training with randomly selected initial values for the parameter number package. The DNN has not learned anything yet. Image credits in Appendix A.3

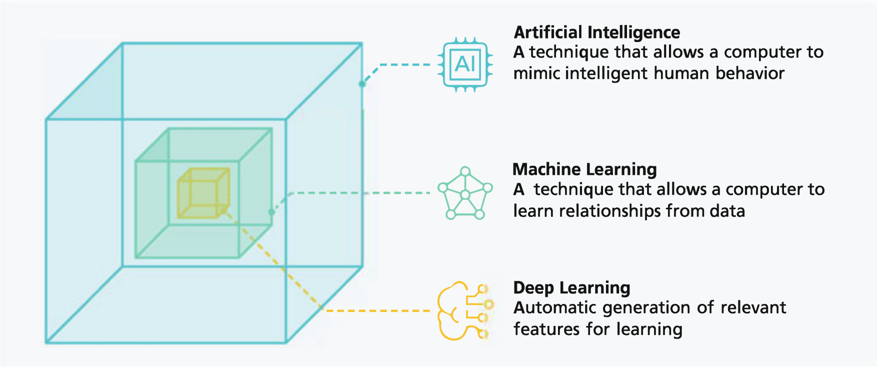

Artificial Intelligence
A technique that allows a computer to mimic intelligent human behavior
AI
Machine Learning
A technique that allows a computer to learn relationships from data
Deep Learning
Automatic generation of relevant features for learning

Fig. 1.8 Artificial Intelligence is an umbrella term of Machine Learning, which in turn encompasses deep learning

simultaneously by successive small changes in such a way that the correct output is generated in a high percentage of cases.

This approach has recently led to surprisingly good results in a variety of sophisticated recognition tasks. This process is also called "Deep Learning". The details of this learning process will be presented in later chapters.

Deep learning is a special technique of Machine Learning, which includes all methods for finding patterns and relationships in data (Fig. 1.8). For example, such a system can predict tomorrow's precipitation from today's measurements of air pressure, temperature and wind direction. Artificial Intelligence is an umbrella term of Machine Learning, which in turn encompasses Deep Learning.

Although Artificial Intelligence and deep neural networks are discussed in many journal articles and talk shows, for most people the workings of these computer programs are in the dark. This book therefore aims to clarify for an interested public what Artificial Intelligence and deep neural networks are and how they work. Not only the internal mechanisms will be presented, but also the current possibilities and limitations will be clarified.

1.5 How To Understand Artificial Intelligence

Most people will think of themselves as understanding how a car works. Figure 1.9 shows the functional diagram of a car. In the engine, pistons catch the pressure generated by combustion and convert it into rotary motion via the crankshaft. The transmission in interaction with the clutch determines the speed of the rotary motion, which is transmitted to the wheels via the differential. This rough sequence is sufficient for most people to understand the reactions of the car when controlled by the driver. Yet details of the electronic engine control, the transmission with its twisted gears, the power steering, the brake booster, etc. are extremely complicated and cannot be understood without an engineering education.

Artificial Intelligence can be understood on a similar level of abstraction. Here, forces are not transmitted by mechanical components, but number packets are sent through operators that transform an input number packet into an output number packet according to a simple scheme (Fig. 1.10). The input number packets represent the application's inputs, e.g. images, sound recordings, texts, videos. Each operator generates a new number packet, which is usually used as the input number packet of the next operator. The set of connected operators is called a model. The last output

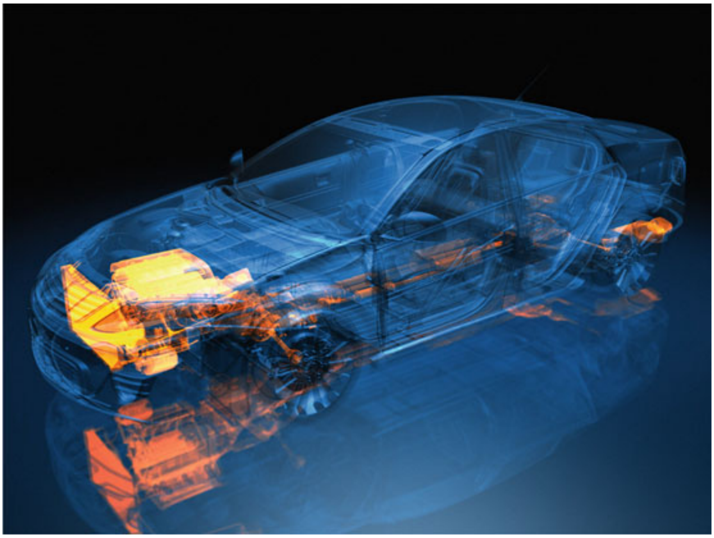


Fig. 1.9 Functional diagram of a car with engine, transmission, drive shaft, differential and wheels. Image credits in Appendix A.3

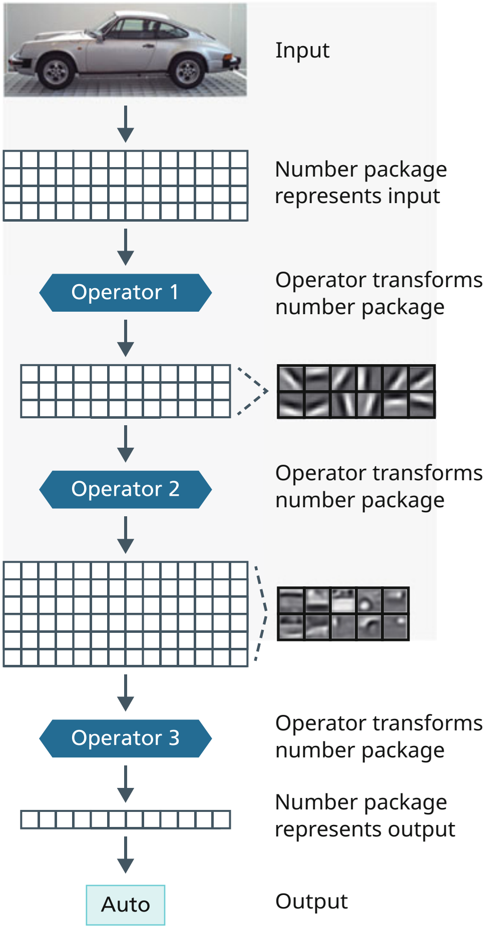

Input
Number package represents input
Operator 1
Operator transforms number package
Operator 2
Operator transforms number package
Operator 3
Operator transforms number package
Number package represents output
Auto
Output

Fig. 1.10 Functional diagram of an Artificial Intelligence model with input, number packages, operators, and output. The number packets represent different content depending on the layer. Image credits in Appendix A.3

number packet of the model represents the desired response, e.g. an image category, a translation, or a new image, which is generated by the model.

The understanding of Artificial Intelligence in this book is conveyed at this high level of abstraction. The rough function of the individual operators are explained, similar to the explanation of the engine, the transmission and the differential in a car. The flow of number packets through the model is explained, analogous to the transmission of power in a car. And it roughly outlines the operation of the optimization modules that adapt the model to the training examples. These modules are usually provided by the existing programming tools.

The idea of Artificial Intelligence conveyed in this book remains at this relatively abstract level. Many details are very complex, but also not necessary for a basic understanding.

1.6 The History of Artificial Intelligence

It is instructive to consider the mercurial history (Haenlein & Kaplan 2019) of Artificial Intelligence (Fig. 1.11). When the first programmable computers were developed in the middle of the last century, researchers soon wondered whether these devices could also exhibit intelligent behavior. To test a system for intelligent behavior, Alan Turing suggested the “Turing test” in 1950. In 1956, the Dartmouth Workshop was held by John McCarthy and Marvin Minsky, which coined the term “Artificial Intelligence.” A year later, Frank Rosenblatt developed a neural network, the perceptron, which could be trained to distinguish simple patterns. Around the same time, the first programs for logical reasoning were introduced. One application

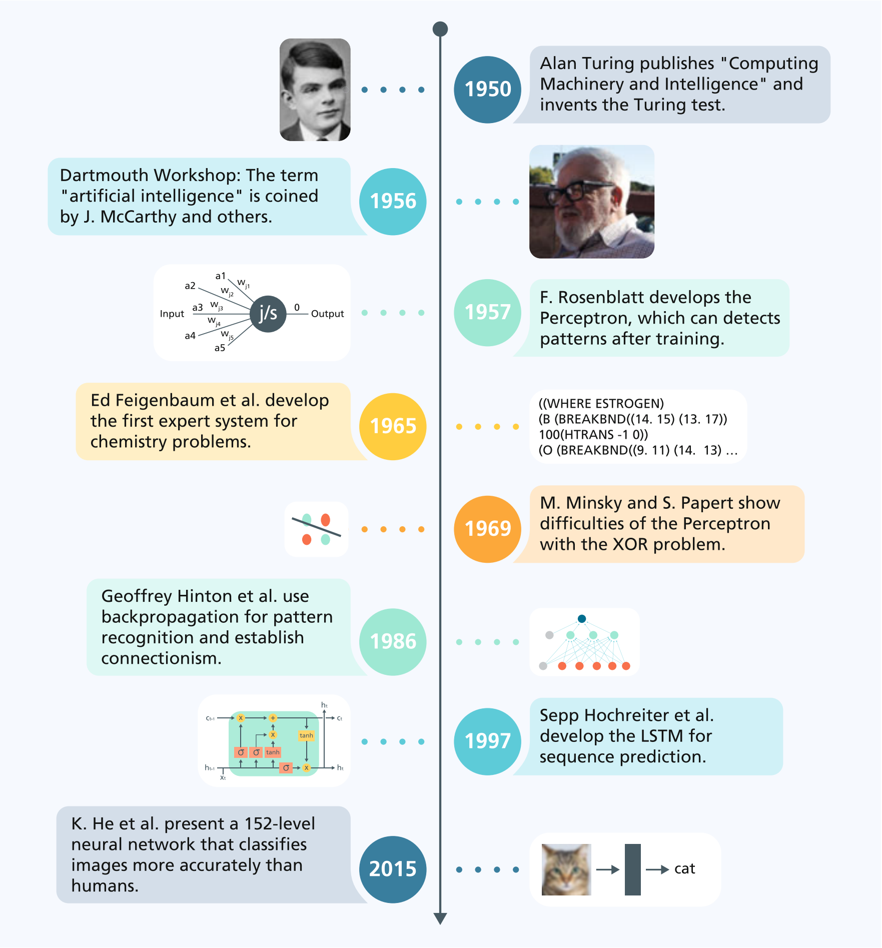

1950
Alan Turing publishes "Computing Machinery and Intelligence" and invents the Turing test.
Dartmouth Workshop: The term "artificial intelligence" is coined by J. McCarthy and others.
1956
a1
a2
a3
a4
a5
Input
j/s
0
Output
1957
F. Rosenblatt develops the Perceptron, which can detects patterns after training.
Ed Feigenbaum et al. develop the first expert system for chemistry problems.
1965
((WHERE ESTROGEN)
(B (BREAKBND((14. 15) (13. 17))
100(HTRANS -1 0))
(O (BREAKBND((9. 11) (14. 13) ...
1969
M. Minsky and S. Papert show difficulties of the Perceptron with the XOR problem.
Geoffrey Hinton et al. use backpropagation for pattern recognition and establish connectionism.
1986
tanh
1997
Sepp Hochreiter et al. develop the LSTM for sequence prediction.
K. He et al. present a 152-level neural network that classifies images more accurately than humans.
2015
cat

Fig. 1.11 Milestones in the history of Artificial Intelligence. Image credits in Appendix A.3

of this was the expert system DENDRAL, presented in 1965 by Edward Feigenbaum and others, which was based on rules and could solve problems in organic chemistry.

In 1969, Marvin Minsky and Seymour Papert showed that single-layer perceptrons cannot solve complicated problems. This almost brought research on neural networks to a halt. During this time, symbolic Artificial Intelligence was being developed in parallel, which aimed to create intelligent systems that could reason with facts and rules. Several years later it could be shown that multilayer neural networks with nonlinear elements can also represent complex relationships. In 1986, David Rumelhart, Geoffrey Hinton and Ronald Williams propagated the use of the backpropagation algorithm for training such networks. Based on this approach they founded connectionism, which aims to describe mental phenomena by networks of simple units. Artificial Intelligence at this time consists of two camps: symbolic AI as well as connectionism. In the 1990s, neither the symbolic expert systems could solve larger problems, nor the neural networks could handle complex recognition tasks with the computers available at that time.

In 1997, Sepp Hochreiter and Jürgen Schmidhuber suggested the Long Short-Term Memory, which promised much better results in the modeling of sequences (text, speech recognition). However, a decade passed before these advantages could be realized, and graphics cards with high computing power became available. In 2015, a Deep Neural Network with 152 layers that can recognize images better than humans is presented by Kaiming He and others. Similar successes are reported in subsequent years for translation into other languages, speech recognition, image captioning and other tasks.

1.7 Summary

When machines or computers exhibit cognitive or mental abilities similar to those of humans, this is called Artificial Intelligence. These abilities can be, for example, problem solving or learning from experience. To test whether a system is intelligent, the Turing test is used, in which an examiner can communicate via text messages with a computer and an intelligent human expert. If, after extensive communication with both communication partners, the examiner cannot decide who is a computer and who is a human, then—so the conclusion is—the computer can be called intelligent.

The assessment of whether a task requires intelligence or not has changed considerably in recent decades. At first, chess was considered one of the highest intelligence achievements of humans. Then, computer programs were developed that were able to beat even the world chess champion by logically evaluating the possible chess moves. After that, playing chess was devalued as "mechanistic" reasoning and no longer counted as part of the core of human intelligence (Fig. 1.12). If a problem can be solved by a machine, it is often subsequently stated that problem solving does not require intelligence (McCorduck 2004, p. 204). Thus, the definition of "true" human intelligence changes over time.

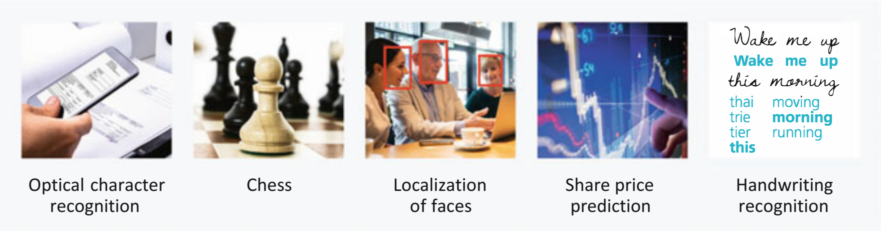


Fig. 1.12 Tasks solved by computers, which are usually no longer considered "intelligent" problem solutions by the public. Image credits in Appendix A.3

The concept of intelligence covers abilities in very different fields of application, from figurative-spatial intelligence to linguistic intelligence to interpersonal, social intelligence. The goal of the research field of Artificial Intelligence is, on the one hand, to develop systems that can perform intelligently in all of these areas. On the other hand, there is also the desire to use these systems to understand how humans accomplish these intelligence capabilities in their brains. Unfortunately, today's "intelligent" computer systems function according to largely different principles than the human brain. Therefore, the mechanisms of human intelligence are still largely in the dark.

Artificial Intelligence computer programs receive information from outside in the form of images, texts, sound sequences, etc. All this information is transformed into packets of numbers. The program itself consists of many "layers" or "operators" that receive number packets as input and transform them into new number packets by simple mathematical operations (addition, multiplication, application of simple functions). The output number packets generated in this way are further processed by other simple structured operators. In the process, the input is transformed into increasingly abstract representations that better and better represent the essential features of the inputs for the sought problem solution. Finally, the last operator can compute the desired output in a simple way from the last representation.

The program defined in this way is called a Deep Neural Network (DNN). It contains parameters, which themselves form a number package, with millions up to billions of numerical values. These numerical values are adjusted by optimization procedures so that the observed data can be reproduced as well as possible. The operation of the individual operators can be well understood and the contents of the intermediate number packages can usually be well visualized. In this sense, such a deep neural network can be understood.

References

Álvarez, S. (2018). Deutschland Will Bei Künstlicher Intelligenz Führend Sein. In *Tagesspiegel* (Zugriff am 22 03 2019).

Copeland, B. (2019). *Artificial Intelligence*. Britannica. https://www.%20britannica.com/technology/artificial-intelligence

De Houwer, J., Barnes-Holmes, D., & Moors, A. (Aug 1, 2013). What is learning? On the nature and merits of a functional definition of learning. *Psychonomic Bulletin and Review, 20*(4), 631–642. ISSN: 1531-5320. https://doi.org/10.3758/s13423-013-0386-3. Visited on 27 Feb 2023.

Gardner, H. E. (1983). *Frames of mind: The theory of multiple intelligences*. Basic Books.

Haenlein, M., & Kaplan, A. (2019). A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review, 61*(4), 5–14.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM, 54*(10), 95–103.

McCorduck, P. (2004). *Machines who think* (2nd ed.). A. K. Peters, Ltd. ISBN 1-56881-205-1.

Tung, L. (2018). AI Will Create $13 Trillion in Value by 2030 - But Get Ready to Change Your Occupation. ZDNet Online 10 September 2018.

Turing, A. M. (1950). Computing machinery and intelligence-AM Turing. *Mind, 59*(236), 433. ISSN: 0026-4423.

Chapter 2
What Are the Capabilities of Artificial Intelligence?

Abstract In recent years, advances in computer processing power and the availability of suitable programming environments and algorithms have made it possible to solve some Artificial Intelligence subtasks in a satisfactory manner. This chapter provides an informal overview of the state of the art. Of particular importance here is the interpretation of sensor data, such as recognizing objects in photos, diagnosing diseases from images, or transcribing spoken language into text. There has also been progress in analyzing the meaning of language, e.g., machine translation from one language to another, answering questions by an AI system, or conducting meaningful dialogues by intelligent assistants. Finally, AI systems have been able to beat human experts at computer games, automatically drive vehicles in real-world traffic, or perform creative acts, such as inventing new stories. The techniques used will be explained in later chapters.

Already in ancient times, there were myths and legends about artificial beings that were endowed with intelligence and consciousness by their master (Delipetrev et al. 2020). So the idea of creating artificial beings is an old dream of mankind. The term Artificial Intelligence was coined by John McCarthy at a workshop at Dartmouth College in 1956 (Moor 2006), shortly after the first computers had been developed. After an initial euphoria, it became clear that the difficulties in achieving this goal had been grossly underestimated. It is only in recent years that advances in computer processing power and the availability of suitable programming environments and algorithms have made it possible to solve some subtasks of Artificial Intelligence in a satisfactory manner.

In this chapter, a spectrum of solutions achieved is presented to give the reader an overview of the range of applications. The underlying techniques will be explained in later chapters. An up-to-date overview of the state of the art is provided by the website Paperswithcode (2019).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_2

2.1 Object Recognition in Images

As already described in Sect. 1.3, object classification in images aims at automatically identifying different classes of objects (dog, cat, car, . . .) in images. The identification of the objects is being learned by using extensive training data, which contain the class names of the depicted objects for each image.

ImageNet (Deng et al. 2009) is a large image database for testing image recognition software. It contains more than 14 million images that have been manually annotated by people. This was done with the help of so-called crowdworkers on the Internet, who were paid a small amount of money, e.g. 20 cents, for annotating each image. The annotation of an image contains the object classes that appear in an image. There are more than 20,000 object classes in total on ImageNet. For most object classes, such as "balloon" or "strawberry," ImageNet contains several hundred images.

Since 2010, an annual competition has been held in which different research groups could apply their image recognition software to a reduced problem with 1000 object classes from ImageNet. The quality of the software was evaluated using the top-5 error rate, which measures whether the true object category is one of the top five predicted categories. The predictions are to be ranked according to their plausibility. Figure 2.1 shows sample images from ImageNet along with the true object category and the top five most plausible predicted categories.

In 2011, a top-5 error rate of 25% was achieved with conventional methods. In 2012, there was a breakthrough with a Top5 error rate of 16%. This was AlexNet, a deep neural network (DNN). In 2015, another DNN achieved a Top5 error rate of 4.9%. This is better than the performance of human annotators, which achieved a Top5 error rate of 5.1% in extensive experiments on the same data. In 2017, a new neural network achieved a Top5 error rate of 3.8% (Zoph et al. 2018). The most plausible prediction was incorrect in 17.3% of the cases.

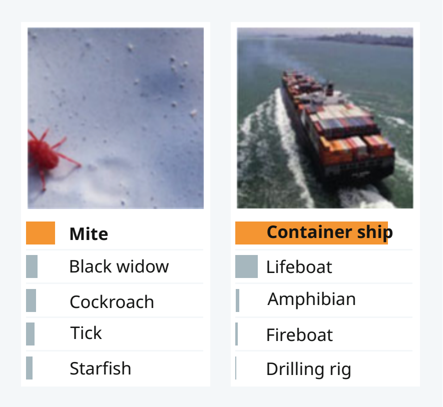


Fig. 2.1 Two images from the ImageNet collection. Below are the categories predicted by the system, ordered by decreasing plausibility (indicated by bar length). The orange coloring marks the correct image class. Neither image was used for training. Source: (Krizhevsky et al. 2012) with self-created graphic

This means that the most modern systems for automated image recognition are now practically as good as humans: if they are to identify an object in any photograph, they make on average as few errors as a trained observer. The technology of image recognition is presented in Chap. 5.

2.1.1 Medical Diagnosis

Image recognition techniques can also be applied to the interpretation of medical image data. Stanford University, for example, has developed a computer system that can detect skin cancer with high precision (Esteva et al. 2017). In this work, a team of dermatologists and computer scientists fed an image recognition system with about 130,000 images of different types of skin cancer, as well as images of normal skin and benign skin lesions. In total, they used 2000 different object classes.

Figure 2.2 contains example images of skin cancer and other skin abnormalities. They show how difficult it is to distinguish malignant from benign skin lesions. To evaluate the quality of the diagnoses, the results of the image recognition system were compared with the diagnoses of 21 dermatologists who had examined the same moles, skin lesions, and melanomas, respectively. The results showed that the automatic diagnosis was as good as that of the dermatologists. The DNN for skin cancer detection is shown in Fig. 5.27.

Skin cancer is, of course, an ideal field of application for computer diagnostics. It is visible from the outside, it occurs frequently, and early detection is particularly important, as this is when the chances of cure are highest.

In 2018, a new result was published (Mar & Soyer 2019), which used a different dataset with more than 100,000 images of skin photos for training. This time, dermatologists were able to detect 88.9% of malignant tumors, whereas the deep neural network was correct 95% of the time.

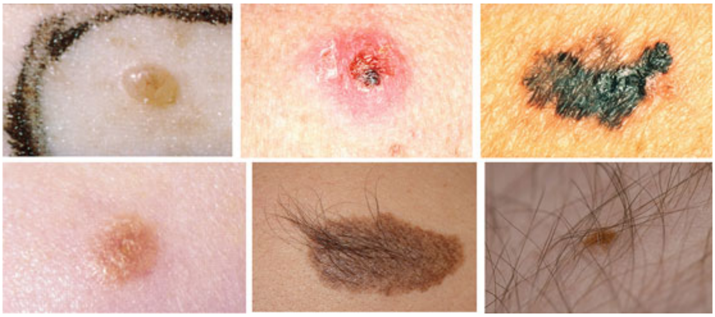


Fig. 2.2 Example images of skin cancer (top) and other skin abnormalities (bottom). In total, there are more than 2000 different classes of skin abnormalities, which are extremely difficult to distinguish. Image credits in Appendix A.3

Overall, the deep neural network detected more tumors than the dermatologists and diagnosed fewer benign skin areas as malignant. This was true even when the dermatologists received additional clinical information about the patients. This shows that deep neural networks now are able to provide better results than experienced dermatologists. There are plans to make a cell phone app available in the future that will allow patients to independently identify suspicious skin sites. Similar systems have been published in many other areas of medicine, including for diagnosing eye diseases (De Fauw et al. 2018), cardiac arrhythmias (Hannun et al. 2019), interpreting chest X-rays (Irvin et al. 2019), and diagnosing 50 different types of tumors from DNA in the bloodstream (Davis 2020). It should be noted, however, that the performance of AI systems tends to decline when they are used outside of carefully designed experiments.

2.1.2 *Predicting the 3D Structure of Proteins*

Proteins are large, complex molecules that make up our bodies and are the basic building blocks of life. Almost all bodily functions—digestion, respiration, vision, etc.—use proteins. The blueprints for proteins are encoded in our DNA and are called genes.

The function of a protein is determined by its 3D structure. For example, antibodies for immune defense have a Y-shape and collagen proteins are filaments and can transmit forces. However, it is very difficult to predict the spatial structure of proteins from the sequence of its components. This is because interactions between the different amino acids must be taken into account, which ultimately determine the 3D structure. Since many diseases are based on deviations in the spatial arrangement of proteins, the prediction of the 3D structure is extremely important.

Google subsidiary DeepMind has succeeded in correctly predicting the 3D structure of many proteins using Deep Neural Networks (Evans et al. 2018). The approach predicted the distances between pairs of amino acids and the angles between the chemical bonds connecting the amino acids. With this information, researchers were able to search the space of possible configurations to find suitable 3D structures. Figure 2.3 shows the predicted structures (blue) and the actual structures (green) for some proteins. Here, on the one hand, the complex spatial structure of the processed proteins becomes clear, and on the other hand, the degree of agreement between actual and predicted shape becomes visible.

In chemistry, there are a number of other applications of AI, such as planning the synthesis of organic molecules. Segler et al. (2017) train a DNN with 12 million known reactions and can generate a synthesis plan for 40% of the test molecules.

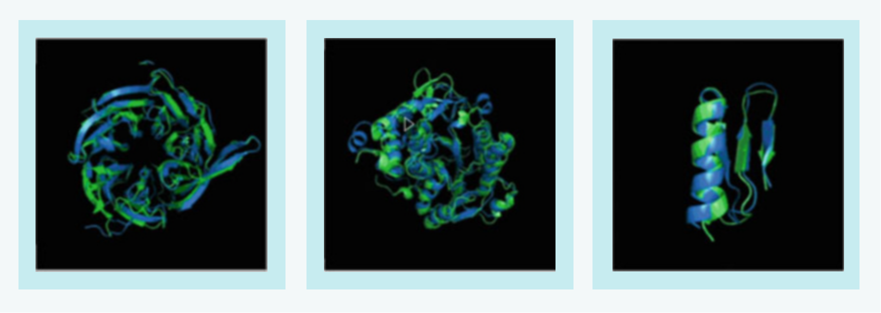


Fig. 2.3 The 3D structures of some proteins predicted by the deep neural network (blue) and the corresponding correct structures (green). The images are screenshots of a 3D animation. Image credits in Appendix A.3

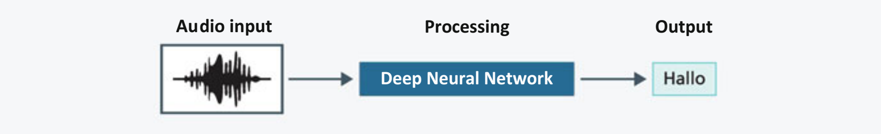


Fig. 2.4 Schematic of a speech recognition system. Input is the numerical representation of the speech signal and output is the recognized text

2.2 Speech Recognition

Speech recognition is the process of converting spoken language into text. Speech is the most natural way for humans to communicate, and with more than 5 billion mobile phone owners in the world, speech recognition on these mobile phones will play a very important role. The first computer systems for this purpose were already available in the 1980s, but they still had a high error rate. With the help of DNN, better results could now be achieved (Fig. 2.4).

This system used as training data the switchboard corpus (Godfrey et al. 1992), which contains a large number of spoken English sentences and their associated transcriptions into written form. These are telephone conversations between different people talking about topics such as sports and politics. In elaborate experiments, the word error rate, i.e., the number of misrecognized words, of professional human listeners was measured (Xiong et al. 2018). It was 5.9% for this dataset. When the conversations were transcribed several times by a group of listeners, the error rate dropped to 5.1%.

Finally, by using the data, a DNN was trained that achieved an error rate of 5.1% (Xiong et al. 2018). This makes the system better than individual human listeners and has the same quality level as a team of experts.

In the meantime, voice recognition is already used by a great number of users, especially with voice-controlled personal assistants, such as Amazon Alexa and

Fig. 2.5 Speech recognition is a key component of personal assistants that allows users to verbally request various information and services. Image credits in Appendix A.3

Google Home (Fig. 2.5), with which users can request different services. Speech recognition is mostly Internet-based, i.e., performed on company servers by using DNN, but can also be performed offline on the most current smartphones. Actual DNNs for speech recognition are presented in Chap. 7.

2.3 Machine Translation

Machine translation is the computer-aided translation of a text from one language into another. As the story of the Tower of Babel shows, the desire to understand unknown languages is very old. With the appearance of powerful computers, work began on automatic translation programs. They were based in particular on dictionaries and grammatical analysis, and increasingly on statistical methods.

Since 2016, deep neural networks have been used for translation. The deep neural network receives the training data as a list of sentence pairs, which consist of a sentence in the source language and its translation in the target language. The characteristic feature here is that the translation is not word for word, but sentences are translated as a whole. Figure 2.6 shows the process for an exemplary sentence.

Machine translation systems have undergone a rapid evolution over the last few years. Millions of people use online translation systems and cell phone apps every day to communicate across language barriers. Microsoft introduced a new version of its translation system in 2017 (Hassan et al. 2018), which consists of multiple DNNs but is trained as a whole. A training set of about 26 million sentence pairs was used for Chinese to English translation. Given a set of sentence pairs that were not used for training, this system was able to achieve the same translation quality as human translators. This was verified in extensive comparative experiments. What is surprising here is that the program uses neither dictionaries nor grammars, but learns the translations from the sentence pairs alone.

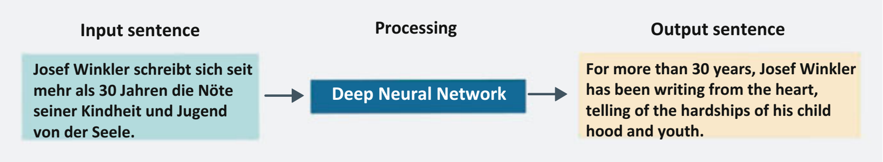

Input sentence
Processing
Output sentence
Josef Winkler schreibt sich seit mehr als 30 Jahren die Nöte seiner Kindheit und Jugend von der Seele.
Deep Neural Network
For more than 30 years, Josef Winkler has been writing from the heart, telling of the hardships of his child hood and youth.

Fig. 2.6 Deep neural network for translation. The example shows that translating a sentence into another language can lead to completely new sentence structures

Fig. 2.7 Simultaneous translator at a World Trade Organization event in 2017. Image credits in Appendix A.3

Every year, there is a competition at the Conference for Machine Translation (WMT) where the best research teams in the world demonstrate the effectiveness of their translation systems on given test data. In 2018, this also revealed for more language pairs that the accuracy of machine translation systems is "very close to the performance of human translators." (Bojar et al. 2018). DNN systems for machine translation are described in detail in Sect. 6.5.

In the book "Hitchhiker's Guide to the Galaxy" (Adams 1979), the babelfish is a small creature that can be put in the ear, allowing its owner to instantly understand all spoken languages. For a human translator such simultaneous translation is very difficult, because he has to listen and translate at the same time (Fig. 2.7). For example, when translating the sentence "Ich bin mit dem Zug nach Bonn gefahren" into English as "I traveled by train to Bonn", the word "traveled" cannot be pronounced until the verb "gefahren" is known. This long delay is very annoying for the listener. The quality of the translation is poor because the translators' error rates increase sharply after a few minutes. Translators are so exhausted after 15–30 minutes that they need to recover.

Ma et al. (2019) have achieved a breakthrough in the field of simultaneous translation. Their translation method requires only a short delay, but has almost the quality of the translation of the complete known sentence, i.e. no relevant content is omitted. Such methods have the potential to make real conversations with people speaking other languages possible in the first place for politicians, business travelers and tourists. Current DNNs for audio-to-audio translation are described in Sect. 6.8.5.

2.4 Answering Natural Language Questions

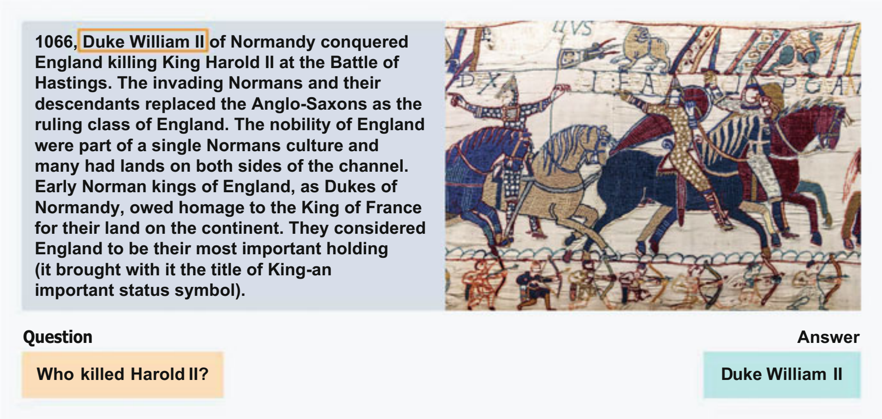
Automatic question answering is a subtask of Artificial Intelligence. Here, a computer system is developed that can automatically answer questions posed in natural language. For the answers, the system can either use a database of information or can extract the relevant facts from an unstructured set of text documents.

The Watson program system (Ferrucci 2012) became famous in 2011, when it was able to beat the top two previous contestants on the quiz show Jeopardy (Fig. 2.8). Watson is a semantic search engine that can search a 100 GB database of facts and relationships. It finds not only exact matches, but also paraphrases of a noun. It consists of many interconnected functional units that have the task to understand the questions, search and evaluate information, and compose the answer.

Many techniques of symbolic Artificial Intelligence were used by Watson. These represent the features of a problem by logical facts and rules (e.g. *authorVon*[*Dracula, Bram_Stoker*] and *bornIn* [*Bram_Stoker, Ireland*]). The solution to a problem is then found by logical reasoning.

Unfortunately, in other applications, such as medical diagnoses, the Watson system disappointed the high expectations (Krempl 2018). The medical researchers identified "multiple examples of unsafe and incorrect treatment recommendations." It turned out that pre-programmed search strategies are not flexible enough.

Fig. 2.8 Jeopardy competition between the Watson computer system (central desk on the right) and its human opponents. Image credits in Appendix A.3



1066, Duke William II of Normandy conquered England killing King Harold II at the Battle of Hastings. The invading Normans and their descendants replaced the Anglo-Saxons as the ruling class of England. The nobility of England were part of a single Normans culture and many had lands on both sides of the channel. Early Norman kings of England, as Dukes of Normandy, owed homage to the King of France for their land on the continent. They considered England to be their most important holding (it brought with it the title of King-an important status symbol).

Question	Answer
Who killed Harold II?	Duke William II

Fig. 2.9 Question from the SQuAD1.1 question collection. On the top left is the section of the Wikipedia article to which the question refers. The question is below it and the answer in the lower right corresponds to the passage in the text outlined in orange. The image shows a depiction of the Battle of Hastings in the Bayeux Tapestry. Image credits in Appendix A.3

Apparently, a system must discover on its own a way to find answers. This objection applies to many symbolic Artificial Intelligence approaches.

To demonstrate the power of learning systems, a set of standard tasks were defined, e.g., SQuAD (Stanford Question Answering Dataset) (Rajpurkar et al. 2016). To create it, sections of Wikipedia articles were first randomly selected from the entire Wikipedia. More than 100,000 questions were formulated by crowdworkers for these sections. The answer to these questions is a sequence of words from the respective section. Questions and answers should contain different formulations as far as possible. However, complex inferences involving multiple sentences should not be required to answer the question. Figure 2.9 shows a typical question, the corresponding Wikipedia section, and the answer (Squad 2019).

Over the past year, the performance of question-answering systems in solving the SQuAD task has increased dramatically. In particular, deep neural networks were used, which were trained exclusively with the 100,000 triples of questions, answers, and sections of the associated Wikipedia articles.

Devlin et al. (2018) introduced a DNN for answering SQuAD questions. It is able to answer questions with fewer errors than human readers. Only 6.8% of the answers are incorrect, compared to 8.8% for humans. To do this, the network obviously needs to develop an understanding of the meanings of words and alternative formulations, and be able to grasp the context of the words. Moreover, the procedures can be directly applied to other languages: no dictionaries, grammars or further lexicons are used. Meanwhile, research focuses on solving complex question problems where the answer must be found in large amounts of text and where world knowledge and logical reasoning are required. Question answering is a subproblem of capturing the meaning of written text and is described in Sect. 6.7.2.

A group of tissues that work together to perform a specific function is called:

1. an organ
2. an organism
3. a system
4. a cell

Which change would most likely cause a decrease in the number of squirrels living in an area?

1. a decrease in the number of predators
2. a decrease in competition between the squirrels
3. an increase in available food
4. an increase in the number of forest fires

Fig. 2.10 Two sample questions from the high school test (Metz 2019)

To attend high school, 8th grade students in New York must pass an entrance test. The Aristo program system managed to answer more than 90% of the questions on the test correctly (Metz 2019) and would thus have gained entry to the high school. Questions on pictures and graphs were excluded and only multiple-choice questions were answered. Two of the questions are reproduced in Fig. 2.10. While the first question requires general knowledge, the second question requires logical reasoning. On the corresponding post-12th grade tests, Aristo was able to answer 83% of the questions correctly.

Aristo's authors consider these tasks to be improved variants of the Turing test (Clark et al. 2019). Ideally, a benchmark should test a variety of skills in a natural and unconstrained way while being clearly measurable, understandable, and motivating. Standardized tests, especially scientific tests, are a rare example of tasks that fulfill these requirements. While they are not comprehensive tests of machine intelligence, they do require some skills that are closely related to intelligence, including language comprehension, logical reasoning, and the use of common sense.

2.5 Dialogs and Personal Assistants

In recent years, personal intelligent assistants (Dubiel et al. 2018) such as Siri, Google Home or Alexa have become popular, communicating with the user in spoken language and performing tasks for him (Fig. 2.11). Virtual assistants use speech recognition modules to translate the user's instructions and questions into text. Then, another part of the system interprets the user's statements, executes orders, and composes the text of a response. Finally, the response is generated by a spoken language generation subsystem. Often, the user continues the dialog, for example, to clarify an instruction or to answer follow-up questions from the assistant. Therefore, the virtual assistant must use the content of the previous

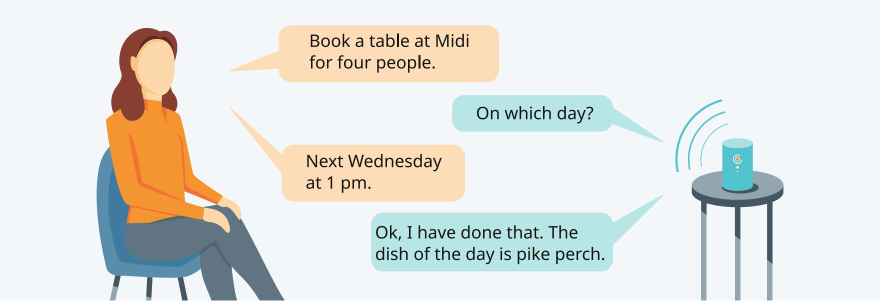

Book a table at Midi for four people.
On which day?
Next Wednesday at 1 pm.
Ok, I have done that. The dish of the day is pike perch.

Fig. 2.11 Dialog with a personal assistant

utterances as context in order to understand the meaning of what is being said and to be able to respond or act adequately.

Users can ask the system questions (for example, about the weather forecast), which can be answered from the Internet. Calendar entries, e-mails or tasks with verbal instructions can be created. In addition, music and other media can be played, restaurant reservations can be made, or purchases can be made on the Internet. Finally, the user can simply talk to the virtual assistant (small talk).

In a competition organized by Amazon, the quality of answers from virtual assistants was evaluated (Khatri et al. 2018, p. 19). It was found that 74% of the responses were rated as passable, good, or excellent. This is far from ideal ratings, but has steadily improved in recent years. The DNNs currently used by speech assistants are discussed in Sects. 7.6, 7.7, and 9.4.

IBM has developed a dialog system Project Debater that can engage in a lengthy discussion with a human opponent (Debater 2018). In a competition between a human speaker and the Debater system, both are given an unknown topic. Both then make a four-minute statement each on the topic, followed by a four-minute rebuttal of the opponent's arguments. Finally, the human audience votes on who had the better arguments.

The computer system has access to very large databases containing millions of newspaper articles. In the process, the system must learn to plan conversations, listen, and understand, and must be able to present a chain of arguments. In June 2018, a showcase event was held with two debates on "Should we fund space exploration?" and "Should we increase the use of telemedicine?" (Fig. 2.12). The machine spoke in a confident female voice with natural speech rhythm and appropriate transitions between sentences. In the following questioning, listeners found that the dialogue system's contributions were linguistically inferior to those of its human opponents, but contained more appropriate information and more persuasive arguments (Solon 2018).

As discussed previously, the recognition of spoken language by computer systems is now as good as of human listeners. Transforming text to speech (speech synthesis, TTS) has also recently seen a leap in quality due to the use of

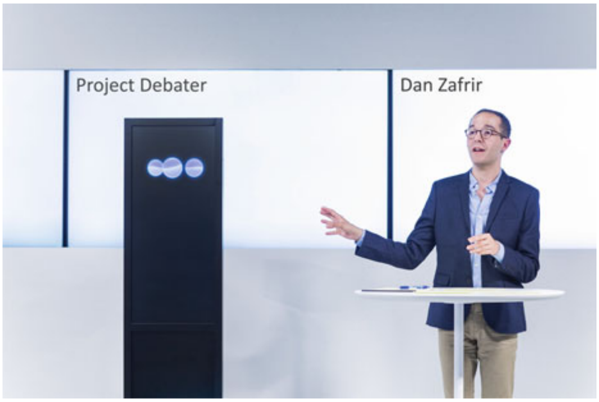


Fig. 2.12 Discussant from project debater with one of his human opponents. Image credits in Appendix A.3

deep neural networks (WaveNet 2016). In a demonstration of the Google Duplex system, listeners could hardly distinguish whether the human or the machine was speaking (Kremp 2018). The speech melody of the system's utterances sounded very convincing, and the system inserted small irregularities ("uh," "Mhmm") into sentences and also produced pauses in speech as if it was thinking. Models of spoken language generation and the structure of speech assistants are presented in Sects. 7.5 and 9.4.

2.6 Board Games

So far, we have mainly discussed applications of Artificial Intelligence that focused on interpreting an image, a speech signal, or a text and generating an adequate response to it. However, many real-world problems are more complex and have a time dimension.

As an example we consider games against an opponent. Here, the system successively receives new information, e.g. the next position on the chessboard, and must perform an action, e.g. choose the next chess move. Only after a larger number of such actions the systems is informed on its performance: e.g. victory, defeat or draw.

Problems of this kind have long been discussed in mathematical game theory and in the field of operation research. Here, solutions for simple problems were derived, which, however, were only applicable to small numbers of possible states (e.g. positions in the game of checkers). However, since e.g. more than 10^{43} positions are possible on the chess board (more than the weight of the complete Milky Way galaxy in kg), approximate solutions for the theoretically best method have to be found.

Fig. 2.13 Garry Kasparov plays simultaneous chess against other grandmasters in 2009. Image credits in Appendix A.3

A first success was achieved with the chess program Deep Blue (Campbell et al. 2002), which in 1997 was able to beat the then world chess champion Garri Kasparov in a competition of six games with 3.5:2.5 (Fig. 2.13). Deep Blue heuristically selected relevant positions and was able to calculate 200 million positions in one second and assess their potential with an evaluation function. Today, computing power has increased so much that chess programs on smartphones can play at grandmaster level.

2.6.1 The Strategy Game Go

In China, Japan and Korea, Go is a very popular game with fairly simple rules: The object of the game is to gain the largest possible territory on the board by placing black or white pieces on the 19×19 board and capturing opposing pieces. A game lasts on average 150 moves. More than 10^{170} positions are possible, which is more than there are atoms in the universe.

Deep Blue's massive search technology no longer helped here. Therefore, Google DeepMind developed the program AlphaGo using DNN. In particular, reinforcement learning was used, which estimates the best actions in a sequence of moves. Figure 2.14 shows a tournament of AlphaGo against the South Korean grandmaster Lee Sedol. In another tournament in May 2017, AlphaGo beat world number one Ke Jie three times (Bögeholz 2018).

Unlike Deep Blue, AlphaGo was not programmed specifically for winning the game of Go, but used a DNN that learned its strategies from a large number of stored Go games. It was then able to automatically select the most promising moves.

The successor program AlphaZero went one step further. It only gets the rules of a game (e.g. Go, Chess or Shogi). The system then plays against another AlphaZero

Fig. 2.14 Commentary of AlphaGo's 3rd game against South Korean Grandmaster Lee Sedol. Image credits in Appendix A.3

system to generate new data, which is used to train the deep neural network. This allows better and better versions of the DNN to be trained. In the end, AlphaZero was able to beat the respective best available programs of Go, Chess and Shogi (Silver et al. 2018).

The amazing thing is that AlphaZero invented new moves and strategies that were not known before in the player community. The chess portal chess.com declared AlphaZero the "most dangerous opponent ever" (Pete 2018). AlphaZero sacrificed pieces in an unprecedented way to win after all. "Its understanding of chess and ruthless, intuitive attacking style is so far above other opponents that they don't even notice when they lose."

2.6.2 Artificial Intelligence Wins Against Five Poker Professionals

Poker is a game, where it is important to keep your opponent in the dark about your strategy. In general, there are four types of players: conservative-aggressive, relaxed-aggressive, conservative-passive and relaxed-passive. Brown and Sandholm (2019) have developed the program system Pluribus, which can simultaneously play "No Limit Texas Hold'em" against five different opponents. When playing poker with several players, the difficulties are much higher than with two opponents (Fig. 2.15) and it is important to have a good strategy, tactical skills and psychological empathy. Pluribus takes into account that players can switch to a different strategy at any time during the games. The program was able to win more often than real poker experts, including Darren Elias, record World Poker Tour title holder, and Chris Ferguson, six-time World Series of Poker tournament winner, in 10,000 hands played with six players.

Fig. 2.15 When playing poker against five opponents, it is important to recognize strategy changes of each player. Image credits in Appendix A.3

2.7 Video Games

Successful computer control of video games is far more difficult than mastering board games. This is mainly because the player has to interpret the content and sequence of the video images in order to select the next move. While a board position can be described with a few words, the sequence of images in the video must be analyzed to recognize the current game situation.

2.7.1 Atari 2600 Game Console

Atari 2600 (Perry & Wallich 1983) was a widely used 1977 game console with a large range of video games (Fig. 2.16). These had a much simpler design than today's photorealistic video games and had an image size of only 210 × 160 pixels.

Video games are ideal for having one or more players replaced by software agents, also called bots. In 2015, Mnih et al. (2015) introduced a DNN with subnetworks for image recognition and reinforcement learning that could control the Atari game console. These nets used only video images and the game score as input. The bots were trained by using sample videos for each game, employing the same network architecture for each game. The bots were able to achieve better results than professional game testers on 29 out of 49 games. Meanwhile, the method was improved by extending the planning radius and tuning the exploration strategies. The bots were now able to beat average human players on 40 out of 42 Atari games (Pohlen et al. 2018). The application of deep neural network learning techniques to Atari video games is presented in Sect. 8.3.

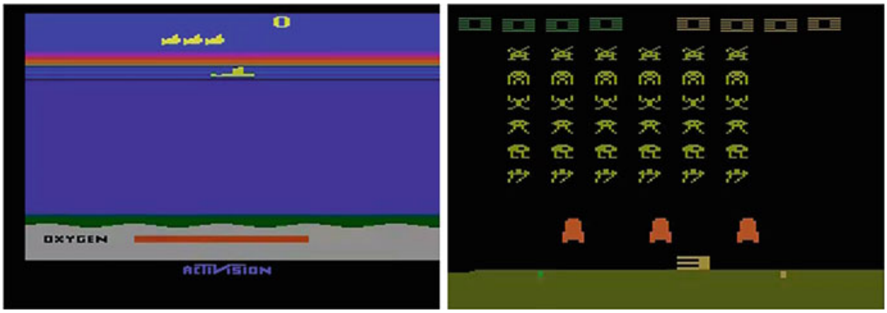


Fig. 2.16 The Seaquest and Space Invaders games for the Atari 2600 game console. Image credits in Appendix A.3

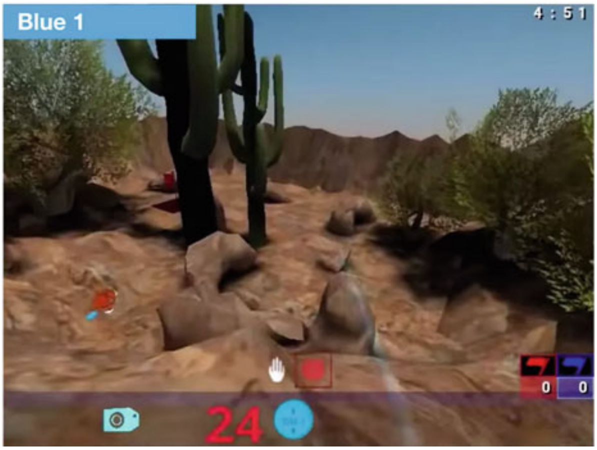


Fig. 2.17 Perspective of a bot from a game scene in the video game Quake III Capture the Flag. This demonstrates the level of difficulty of the game. Image credits in Appendix A.3

2.7.2 *Capture the Flag Quake*

Quake III Arena Capture the Flag is a popular 3D video game in which two teams of players play against each other. The teams operate in a given landscape and have the task of capturing the other team's flag while protecting their own flag (Fig. 2.17). To gain a tactical advantage, they can tag the opposing team members to send them back to their starting points. The team that has captured the most flags within five minutes wins. To make the game more interesting, the landscape is changed after each game.

This type of video games fascinates millions of players because of the realistic involvement in a game environment and because of the requirements of strategy, tactics, hand-eye coordination and cooperation with other players.

Google DeepMind trained a group of bots to play the capture-the-flag game (Jaderberg et al. 2019), giving the software agents only the pixels of their perspective's video as input and the final information about winning, tying, or losing. Each agent was trained independently of the others. In the process, he discovered the rules of the game, different complementary strategies, and became a specialist in certain tasks. As a byproduct, the ability to cooperate also developed. For example, the strategies "Defend home base," "Occupy the oppenent's base," and "Follow a teammate" were learned (Jaderberg 2019).

To test the capabilities of the AI agents, DeepMind organized a tournament with teams of two players each. If a team consisted only of bots, it was the most successful, with a probability of winning of 74%. In comparison, average human players won in 43% and strong human players won 52% of the matches. In a separate competition, two professional game testers with full communication played against a fixed pair of bots, which, of course, could not communicate. Even after twelve hours of practice, the human game testers could only beat the team of bots in 25% of the games (6.3% draws) (Jaderberg et al. 2019, p. 5). So the AI agents were clearly the better players.

2.7.3 The Real-Time Strategy Game Dota2

Dota2 is a real-time strategy game, published in 2013, in which two teams of five players compete against each other. The graphics and gameplay are much more complex than Quake Capture the Flag. In Dota2, the player controls a character from a bird's eye view. The aim of the game is to defend the "Ancient", the main building of each team, and destroy the opponent's "Ancient" (Fig. 2.18). The main buildings

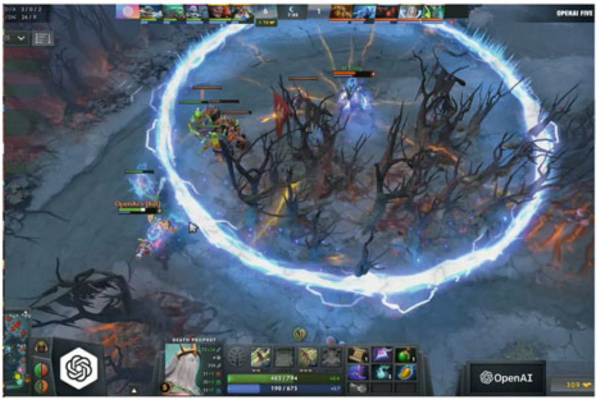


Fig. 2.18 Game scene of Dota2 from the point of view of "Death Prophet", a member of the team of OpenAI, attacking the opponents. Each agent has different abilities. The resources of "Death Prophet" can be seen below. For each teammate, their strength is inserted. Image credits in Appendix A.3

of the teams are located in a landscape. They are protected by towers and connected by paths. A character can gain strength by collecting gold and killing opposing characters, and by other methods. Each of the more than 100 game characters has different skills and talents.

OpenAI has trained bots for Dota2 that can handle this complex game (OpenAI-Five 2018). As input, they receive the state of the game from the bot perspective as a vector of 20,000 numbers. However, the situation is only partially observable for a bot, as many details remain hidden for them. Each bot has dozens of different options with different intensities for his actions which were divided into 170,000 discrete actions. During each day of training, the bots played 180 years of real-world gameplay against each other at an accelerated game speed. This was done using a DNN for reinforcement learning. In April 2019, at an event in San Francisco, the OpenAI software won against the reigning world champion, the e-sports team OG (Steinlechner 2019). It was amazing that the software agent was able to perform the planning for a game lasting on average of 45 minutes. Using similar approaches, a DNN from DeepMind has been able to beat professional gamers in the complex real-time strategy game StarCraft 2 (Vinyals et al. 2019). Details of the models used can be found in Sect. 8.4.5.

2.8 Self-Driving Cars

Self-driving cars are a long-cherished dream of engineers. As early as 1986, a car was developed in Germany that could follow normal roads (Dickmanns & Graefe 1988). In 2005, DARPA held a race that covered 212 km of desert-like terrain in Nevada Fig. 2.19). The course information consisted of GPS coordinates for course

Fig. 2.19 The Beer Bottle Pass, part of the DARPA race course. It is located about 12 km before the finish line and has over 20 hairpin bends. Image credits in Appendix A.3

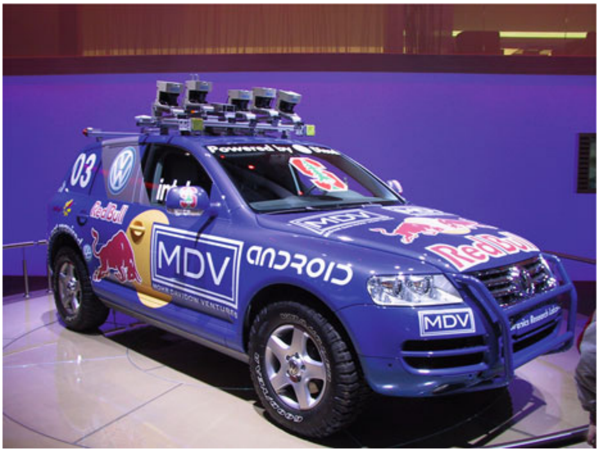


Fig. 2.20 With this VW Touareg with various sensors on the roof, Sebastian Thrun's team won the 2005 DARPA Grand Challenge. Image credits in Appendix A.3

points spaced 72 m apart that were distributed to all teams two hours before the start. Of the twenty-three starters, five cars reached the finish.

The winner was a VW Touareg from Stanford University with an average speed of 30.7 km/h. The winning team, led by Sebastian Thrun, took home $2 million in prize money. The Touareg (Fig. 2.20) used a large number of sensors (video camera, laser rangefinder, radar, GPS, attitude sensors). A software system with about 30 components for position tracking, path planning, and control processed the sensor information and controlled the vehicle. Only a few learning components were used in it. Thrun later led the development of autonomous cars at Google (Fig. 8.27).

2.8.1 Further Development of Self-Driving Cars

Since then, a number of manufacturers have been developing self-driving cars (Daimler, Honda, Audi, VW, Toyota, Tesla, Uber, GM, etc.). Google bundled its activities in the company Waymo, which equipped different models with the necessary sensor technology (Fig. 2.21). In recent years, Waymo has increasingly used deep neural networks to improve the performance of its sensors to detect, for example, lane markings, obstacles, and pedestrians on a road (Hawkins 2018). Figure 2.22 shows an everyday traffic situation analyzed by the control software, in which the autonomous vehicle must apply the correct control actions.

Developers of robotic cars promise that they will make road traffic much safer—after all, more than 90% of accidents with robotic cars were due to human error. In October 2018, Waymo announced that its vehicles had traveled 10 million miles on

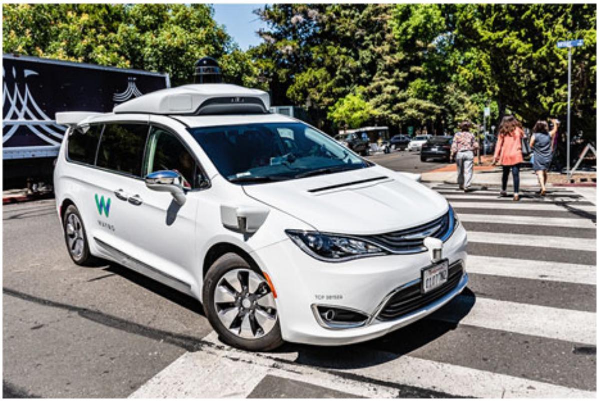


Fig. 2.21 Autonomous car from Waymo. It is a converted Chrysler Pacifica minivan. Image credits in Appendix A.3

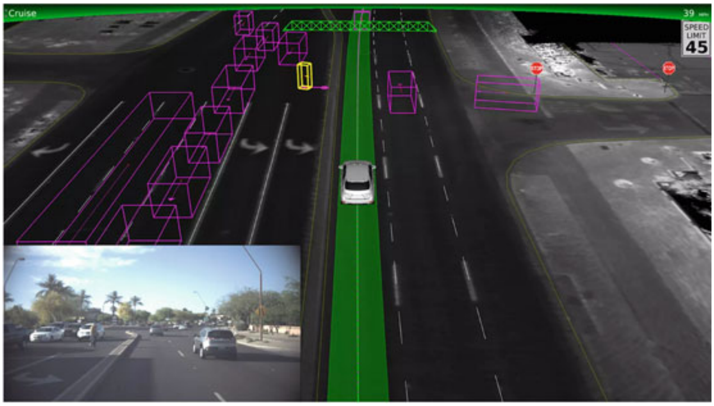


Fig. 2.22 Recognition of the traffic situation with a roadworker at a construction site. Vehicles in the lanes as well as people are represented by cubes. Traffic signs are shown separately. Image credits in Appendix A.3

public roads. While Tesla and other manufacturers have had a number of accidents, Waymo's computer system has recently had only one crash with fender bender (Vardhman 2019). Waymo safety drivers only had to intervene on average every 18,000 km in 2018.

In late October 2018, Waymo became the first company to receive permission from the governing authority in California to operate fully driverless cars (i.e., without human safety drivers). Waymo announced that it will optimize self-driving vehicles in an "early rider program" with hundreds of selected test customers (Marshall 2018). The sensors, model components and learning strategy of self-driving cars are discussed from Sect. 8.5.

2.9 The Computer as a Creative Medium

Computers are known for the logic and precision with which they can perform very complex calculations over and over again. Therefore, the idea that computers can be creative is rejected widely with the argument: “After all, the computer only executes what has been programmed in.” In this section, we show that computer programs can indeed be creative in certain applications. However, creativity is a badly defined term and rather describes the new, unexpected. This also means that it is difficult to measure creativity and often impossible to tell whether a program is more creative than a human. In the following, some examples of creativity through Artificial Intelligence are compiled.

2.9.1 Composing New Images

In October 2018, a computer-generated portrait of a person “Portrait of Edmond Belamy” was sold at Christie’s auction house for $ 432,500 (Quackenbush 2018). The portrait is shown in Fig. 2.23 and was signed with a statistical formula. It was created by an algorithm that had learned painting styles by using a collection of 15,000 portraits from the fourteenth to the twentieth century (Sect. 9.1). “We did

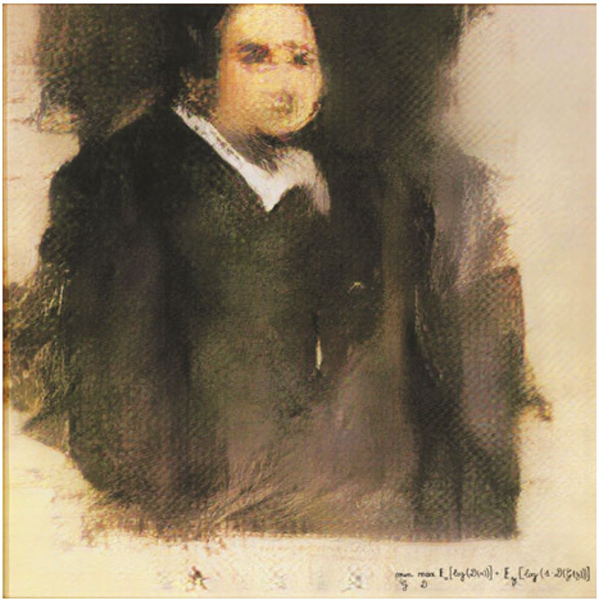

$\min_{G} \max_{D} E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]$

Fig. 2.23 The “Portrait of Edmond Belamy,” a painting generated by an AI program, was sold at Christie’s for $432,500. Image credits in Appendix A.3

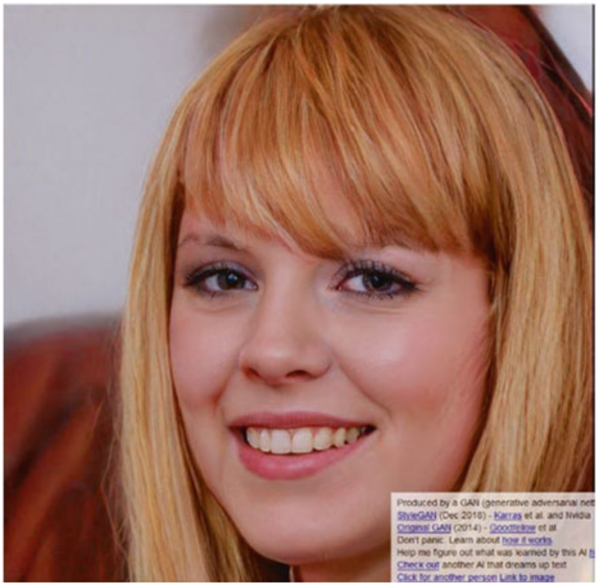

Produced by a GAN (generative adversarial net
StyleGAN (Dec 2018) - Karras et al. and Nvidia
Original GAN (2014) - Goodfellow et al.
Don't panic. Learn about how it works
Help me figure out what was learned by this AI
Check out another AI that dreams up text
Click for another person Link to image

Fig. 2.24 A high resolution portrait of a non-existing person generated with a DNN. On the website https://thispersondoesnotexist.com/ one can see quite a lot of such images (Karras 2019). Image credits in Appendix A.3

some work with nudes and landscapes and also tried to feed the algorithms with the work of famous painters. However, we found that portraits are the best way to illustrate our point: Algorithms are able to emulate creativity," said Hugo Caselles-Dupré, one of the creators of the program.

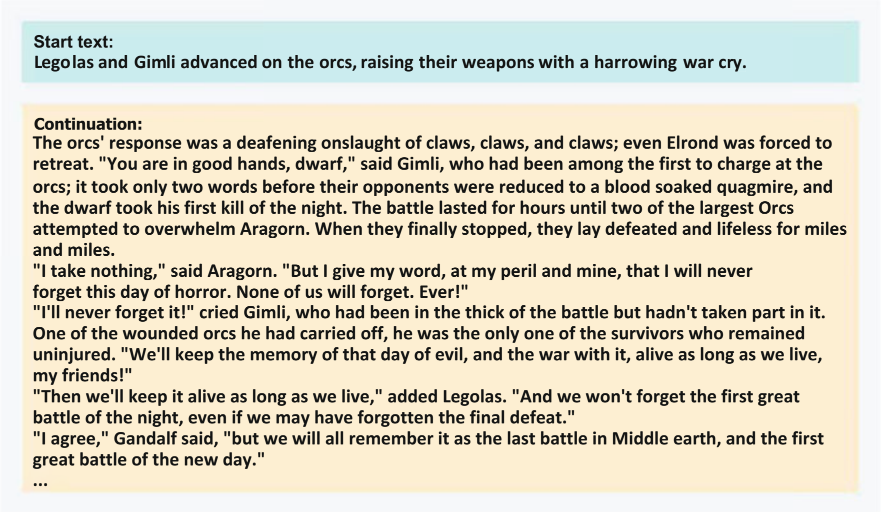
A similar DNN is able to generate photorealistic portraits of people. A wide variety of people can be seen: Caucasians and colored people, fat and thin people, men and women. They have freckles, wrinkles and stubbles. Figure 2.24 shows an example. Each time the procedure is started, a different portrait is produced. The portraits seem convincingly real, but the people shown do not exist; the images were generated by an algorithm (Sect. 9.1.4). This DNN does not assemble parts of images from the training set, but has captured the structure of portrait photos. At each call, it generates a new image where all parts of the image are in adjusted to each other.

Such algorithms can be used in the future to develop realistic images of people for computer games and advertising posters. However, computer-produced images can also be exploited for fake news or falsification of facts. Meanwhile, it is already possible to exchange the heads of actors in videos and put any statements into their mouths (Agarwal et al. 2019). Therefore, nowadays you have to pay special attention when you see a realistic photo or video of a scene intended to transmit a specific message. It is very difficult to debunk such artificial portraits and videos. Clues can sometimes be found in the hair or in the blendings between different regions in the image. Deep neural models for creative generation of images are presented in Sect. 9.1. Analysis and generation of videos is discussed in Sect. 7.8.

2.9.2 Inventing Stories

In February 2019, the research institution OpenAI published results of a DNN that was trained to predict the next word of a text. It takes into consideration the sentence structure, as well as contextual references between words. The DNN is trained with 8 million web pages with a total size of 40 GB and can learn from these web pages both the grammar of a language and contextual relationships between terms. In total, the model has 1.5 billion free parameters and can therefore store a very large number of correlations. An enormous amount of computing power is required to train the model.

The model receives a start text and generates a continuation. As Fig. 2.25 shows, the generated texts exhibit a fluid style and a previously unseen logical and argumentative consistency. At the first glance, they could be written by living humans imitating the style of “Lord of the Rings”. What is interesting is that the model can handle a great number of the topics being discussed in the 8 million training web pages, including Brexit, genetics, and nuclear technology. The authors investigated (Radford et al. 2018) whether the model simply pices together phrases from the training texts. However, this is not the case. The contents are reformulated and brought into a logical context of action and argumentation. The generation



Start text:
Legolas and Gimli advanced on the orcs, raising their weapons with a harrowing war cry.

Continuation:
The orcs' response was a deafening onslaught of claws, claws, and claws; even Elrond was forced to retreat. "You are in good hands, dwarf," said Gimli, who had been among the first to charge at the orcs; it took only two words before their opponents were reduced to a blood soaked quagmire, and the dwarf took his first kill of the night. The battle lasted for hours until two of the largest Orcs attempted to overwhelm Aragorn. When they finally stopped, they lay defeated and lifeless for miles and miles.
"I take nothing," said Aragorn. "But I give my word, at my peril and mine, that I will never forget this day of horror. None of us will forget. Ever!"
"I'll never forget it!" cried Gimli, who had been in the thick of the battle but hadn't taken part in it. One of the wounded orcs he had carried off, he was the only one of the survivors who remained uninjured. "We'll keep the memory of that day of evil, and the war with it, alive as long as we live, my friends!"
"Then we'll keep it alive as long as we live," added Legolas. "And we won't forget the first great battle of the night, even if we may have forgotten the final defeat."
"I agree," Gandalf said, "but we will all remember it as the last battle in Middle earth, and the first great battle of the new day."
...

Fig. 2.25 Automatically generated continuation of the start text by a deep neural network (Radford et al. 2019)

of the texts is highly random: Each repetition produces a completely different continuation. The authors report that for topics well covered in the training material, every other continuation is usable. Details of this and other DNN on creative generation of text can be found in Sect. 9.2.

2.10 General Artificial Intelligence

We have seen a number of use cases in the previous sections where AI systems match or exceed the performance of humans. Most of the sample applications are single perceptual capabilities, such as recognizing objects in images, transcribing audio into written text, or translating a text into another language. This type of AI is also called Weak Artificial Intelligence because it is a partial aspect of human intelligence (Fig. 2.26 on the left).

The long-term goal is Artificial General Intelligence (AGI). This could be a program system, which unites all intellectual abilities of a human being (Pennachin & Goertzel 2007). Such an AI system not only solves one aspect of a problem, but simultaneously processes sensory signals of different types, operates in an environment, and can solve problems (Fig. 2.26 right). Other requirements include an "awareness" that allows the system to reflect on its own role. Overall, an AGI should cover all eight aspects of human intelligence from Fig. 1.1.

This aim is far from being achieved, as will become clear in the course of the following chapters. However, there are already AI systems that cover several aspects of human intelligence. Advanced examples are AI systems for video games. As described previously, these can process different sensory inputs (albeit in simulated environments), pursue an optimal strategy, and even cooperate with other agents. The issue of general Artificial Intelligence is taken up again in the last chapter of the book (Sect. 10.3.4).

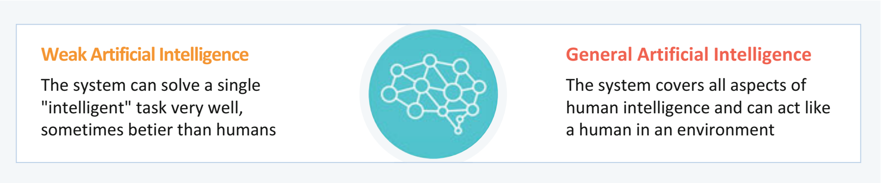

Weak Artificial Intelligence
The system can solve a single "intelligent" task very well, sometimes betier than humans
General Artificial Intelligence
The system covers all aspects of human intelligence and can act like a human in an environment

Fig. 2.26 Weak artificial intelligence and general artificial intelligence

2.11 Summary

The example applications listed in this chapter show that the major advances in Artificial Intelligence in recent years are mainly based on deep neural networks, while symbolic Artificial Intelligence plays a somewhat minor role. Application areas of DNN include image processing, speech recognition, translation into another language, question answering, natural language dialogues with people, board and video games, and self-driving cars.

The focus in this context is on the interpretation of media data, such as images, spoken language, or even texts. However, there are also integrated applications, for example in the gaming sector or for self-driving cars, which involve the interpretation of sensor data, the derivation of an optimal action strategy and acting in a world. Several factors were essential for the success, e.g. the availability of extensive training data, very fast parallel processors, suitable model architectures and new toolboxes with which the models can be easily formulated and trained.

However, the previous presentation should not give the impression that DNNs work well even in practice in every case. In the application examples listed previously, large teams of experts often optimized the DNN in lengthy test series so that the performance on the test data was achieved. This can lead to a very strong alignment of the model to the particular dataset. In a real application, discrepancies and data errors are to be expected, so that the accuracy is often lower. We will discuss these aspects in detail in the further chapters.

References

Adams, D. (1979). *The Hitchhiker's guide to the galaxy*. Del Rey.

Agarwal, S., Farid, H., Gu, Y., He, M., Nagano, K., & Li, H. (2019). Protecting world leaders against deep fakes. In *CVPR Workshop* (pp. 38–45).

Bögeholz, H. (2018). *Künstliche intelligenz: AlphaGo besiegt ke jie zum dritten mal*. https://www.bmbf.de/files/Nationale_KI-Strategie.pdf

Bojar, O., Federmann, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Monz, C., & Fishel, C. (2018). *Findings of the 2018 Conference on Machine Translation (WMT18)*. University of Edinburgh. https://doi.org/10.18653/v1/W18-6401

Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science, 365*(6456), 885–890.

Campbell, M., Hoane, A. J., Jr., & Hsu, F. H. (2002). Deep blue. *Artificial Intelligence, 134*(1–2), 57–83.

Clark, P., Etzioni, O., Khashabi, D., Khot, T., Mishra, B.D., Richardson, K., Sabharwal, A., Schoenick, C., Tafjord, O., Tandon, N., et al. (2019). From'F' to 'A' on the NY regents science exams: An overview of the aristo project. arXiv: 1909.01958.

Davis, N. (2020). New blood test can detect 50 types of cancer. *The Guardian*. 31032020.

De Fauw, J., Ledsam, J. R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., et al. (2018). Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine, 24*(9), 1342–1350.

Debater. (2018). *Project debater*. https://www.research.ibm.com/artificial-intelligence/project-debater/

Delipetrev, B., Tsinaraki, C., & Kostic, U. (2020). *Historical evolution of artificial intelligence*. JRC Publications Repository. Retrieved March 14, 2023, from https://doi.org/10.2760/801580.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv: 1810.04805.

Dickmanns, E. D., & Graefe, V. (1988). Dynamic monocular machine vision. *Machine Vision and Applications, 1*(4), 223–240.

Dubiel, M., Halvey, M., & Azzopardi, L. (2018). A survey investigating usage of virtual personal assistants. arXiv: 1807.04606.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115–118.

Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Zidek, A., Nelson, A., Bridgland, A., Penedones, H., et al. (2018). De Novo structure prediction with deeplearning based scoring. *Annual Review of Biochemistry, 77*(363–382), 6.

Ferrucci, D. A. (2012). Introduction to “This Is Watson”. *IBM Journal of Research and Development, 56*(3–4), 1–15.

Godfrey, J. J., Holliman, E. C., & McDaniel, J. (1992). SWITCHBOARD: Telephone speech corpus for research and development. In *The International Conference on Acoustics, Speech, & Signal Processing*(vol. 1, pp. 517–520). IEEE Computer Society.

Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine, 25*(1), 65.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C., Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. (2018). Achieving human parity on automatic chinese to english news translation. arXiv: 1803.05567.

Hawkins, A. (2018). Inside Waymo’s strategy to grow the best brains for self-driving cars. Verge 09052018. https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., et al. (2019). Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Proceedings of the AAAI Conference on Artificial Intelligence* (vol. 33, pp. 590–597).

Jaderberg, M. (2019). Capture the flag: The emergence of complex cooperative agents. https://deepmind.com/blog/capture-the-flag/

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science, 364*(6443), 859–865.

Karras, T. (2019). This person does not exist.

Khatri, C., Hedayatnia, B., Venkatesh, A., Nunn, J., Pan, Y., Liu, Q., Song, H., Gottardi, A., Kwatra, S., Pancholi, S., et al. (2018). Advancing the state of the art in open domain dialog systems through the alexa prize. arXiv: 1812.10757.

Kremp, M. (2018). Google duplex ist gruselig gut. Spieg. Online 09052018. http://www.spiegel.de/netzwelt/web/google-duplex-auf-der-i-o-gruselig-gute-kuenstliche-intelligenz-a-1206938.html

Krempl, S. (2018). Kampf gegen krebs: Dr. Watson enttäuscht erwartungen. Heise online 15.08.2018.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105).

Ma, C.-Y., Lu, J., Wu, Z., AlRegib, G., Kira, Z., Socher, R., & Xiong, C. (2019). Self-monitoring navigation agent via auxiliary progress estimation. arXiv: 1901.03035.

Mar, V. J., & Soyer, H. (2019). Artificial intelligence for melanoma diagnosis: How can we deliver on the promise? *Annals of Oncology, 30*(12), E1–E3.

Marshall, A. (2018). Waymo can finally bring truly driverless cars to California. Wired 30.10.2018. https://www.wired.com/story/waymo-self-driving-cars-california/

Metz, C. (2019). A breakthrough for A.I. Technology: Passing an 8th-grade science test. *New York Times*, 04-09-2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Moor, J. (2006). The dartmouth college artificial intelligence conference: The next fifty years. *AI Magazine, 27*(4), 87–91.

OpenAI-Five. (2018). OpenAI Five - Open Ai Blog 25.06.2018. https://openai.com/blog/openai-five/

Paperswithcode. (2019). Browse State-of-the-Art in AI. https://paperswithcode.com/sota

Pennachin, C., & Goertzel, B. (2007). Contemporary approaches to artificial general intelligence. *Artificial General Intelligence* (pp. 1–30). Springer.

Perry, T., & Wallich, P. (1983). *Inventing the Atari 2600 IEEE Spectrum, March 1, 1983*. Retrieved March 14, 2023, from https://spectrum.ieee.org/atari-2600

Pete. (2018). Die 5 Gefährlichsten Gegner Aller Zeiten. chess.com 12.05.2018. https://www.chess.com/de/article/view/die-5-gefahrlichsten-gegner-aller-zeiten

Pohlen, T., Piot, B., Hester, T., Azar, M. G., Horgan, D., Budden, D., Barth-Maron, G., Van Hasselt, H., Quan, J., Večerík, M. et al. (2018). Observe and look further: Achieving consistent performance on Atari. arXiv: 1805.11593.

Quackenbush, C. (2018). A painting made by artificial intelligence has been sold at auction for $432,500. *Time 26.10.2018*. http://time.com/5435683/artificial-intelligence-painting-christies/

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. Retrieved October 31, 2020, from https://www.mikecaptain.com/resources/pdf/GPT-1.pdf

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog, 1*(8), 9.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv: 1606.05250.

Segler, M. H., Preuss, M., & Waller, M. P. (2017). Learning to plan chemical syntheses. arXiv: 1708.04202.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science, 362*(6419), 1140–1144.

Solon, O. (2018). Man 1, machine 1: Landmark debate between AI and humans ends in draw. *The Guardian, 18*, 062018.

Squad. (2019). Squad explorer. https://rajpurkar.github.io/SQuAD-explorer/

Steinlechner, P. (2019). Open AI Besiegt Die Weltmeister. golem.de 16.04.2019. https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister-1904-140710.html

Vardhman, R. (2019). 29 Self-driving car statistics & facts. https://carsurance.net/blog/self-driving-car-statistics/Accident_Statistics_on_Googles_Self-DrivingCars.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R. et al. (2019). Alphastar: Mastering the real-time strategy game StarCraft II. Deep. Blog, p. 2. https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii

WaveNet. (2016). WaveNet: A generative model for raw audio. Deep-mind. Retrieved December 17, 2020, from https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2018). The microsoft 2017 conversational speech recognition system. *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP* (pp. 5934–5938). IEEE.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8697–8710).

Chapter 3
Some Basic Concepts of Machine Learning

Abstract Machine Learning is tasked with extracting relationships from data that are relevant to the user. In this chapter, we formulate a simple linear model, the logistic regression model, which predicts the corresponding output for given inputs. The goal is to automatically find the relations between the existing input values and the output category in the data. For this purpose, a large number of numerical parameter values are modified step by step using a simple optimization procedure in such a way that the predicted outputs successively approach the correct outputs. The chapter describes the necessary procedure in all details, but with a minimum of mathematical formulas. The more complex models of the following chapters are built according to exactly the same scheme and use the investigated linear model as a universal building block.

3.1 Main Types of Machine Learning

Humans solve a variety of real-world problems without being able to specify how they do it in detail. An example is distinguishing a dog from a cat. Almost no one can say exactly which features to look for in order to discern a dog from a cat. The same is true for complex movements such as riding a bike. People can almost never describe in detail the movements with which they manage to keep their balance on a bicycle. These are examples of implicit knowledge, which we cannot put into rules or instructions.

If one does not have explicit rules for programming a computer to perform a task, learning from experience or data is an alternative. As discussed previously, humans also learn from examples. Therefore, the idea of trying to do the same with computers came up very early.

Computer learning is a process by which a computer acquires knowledge by evaluating experience. During “Machine Learning”, a system analyzes the available data and modifies itself step by step so that it can perform its task more efficiently. There are three types of learning tasks:

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_3

Fig. 3.1 Supervised learning: the teacher tells the students the names of objects. Image credits in Appendix A.3

3.1.1 *Supervised Learning*

In supervised learning, the system is told what to learn. For example, it is supposed to distinguish cars and houses in pictures. For this purpose, it is shown many images of cars and other images with houses that have already been manually tagged (annotated) as cars and houses. After analyzing a large number of such examples, the system can find patterns by which the objects in the images can be distinguished. Subsequently, the system is able to apply these patterns to new images in order to discriminate images of cars from images of houses. The system is instructed by showing and naming examples, much like the students in the classroom in Fig. 3.1.

In supervised learning, we distinguish two important use cases. For classification, the system must select the answer from a usually small number of alternatives or classes. An example is the categorization of a product as “good” or “defective”. For regression, the system must predict one or more continuous variables, e.g., the maximum temperature and wind speed for the next day in Bonn.

3.1.2 *Unsupervised Learning*

In unsupervised learning, the system must proceed without such manual instructions. It reads the available data and automatically tries to find patterns and regularities in them (Fig. 3.2). For example, it can group data according to their similarity.

This approach is particularly promising when the data has different components. An example are the different words of a written sentence. Then one can try to predict one component (word) on the basis of the others. This type of learning is also called

Fig. 3.2 In unsupervised learning, patterns and regularities in the data are found on their own. Here, a child finds the connections between water and gravity. Image credits in Appendix A.3

self-supervised learning. It is supervised learning, but no humans are needed to provide annotations. This form of learning is widely used in Artificial Intelligence.

For Yann LeCun, a leading AI researcher, self-supervised learning will greatly advance Artificial Intelligence. For example, if a system is given the task to predict the next frames of a given video, it will have to develop a representation of the scenes and predict the possible movements and actions of the objects (Smith 2020). Given enough training videos, it will gain a basic "understanding" of what happens in a video and can apply this to predicting new video scenes.

3.1.3 Reinforcement Learning

In reinforcement learning, the system must first generate a series of actions before it obtains the final outcome. This is the case, for example, in board games (chess, Fig. 3.3) or in robot control. After each action, the environment (e.g., the opponent in chess) reacts and the system receives new information about the current state (e.g., board positions), possibly also rewards (e.g., numerical points, or victory or defeat messages). The goal of the system is to develop an action policy that for arbitrary states generates actions which lead to the highest possible sum of rewards. The technique of reinforcement learning with the aid of rewards and punishments is adapted from psychology and is used, for example, when training dogs.

Fig. 3.3 In reinforcement learning, the agent must react to different states with an action. In doing so, it should learn a strategy allowing it to gain the highest possible sum of rewards during the steps of the process. Image credits in Appendix A.3

3.2 Programming and Learning

In this section, we present the basic operations of Machine Learning for the simplest case, supervised learning.

3.2.1 Models Transform an Input into an Output

In the previous chapter we have introduced a number of successful applications of deep neural networks (DNN): object recognition in images, speech recognition to transfer spoken language into text, or translation of a sentence from one language to another. Deep neural networks are specialized Machine Learning models. In supervised learning these models operate by using the following scheme (Fig. 3.4):

- The model receives an input, such as an image, an audio signal, or written text.
- The model processes this input and generates an output.
- Depending on the aim of the analysis, different outputs are produced: for example, the labeling of the objects in the image, the text from the audio signal, or the translation of the text into the target language.

Mathematically, a model is therefore a function $f : x \to y$ that generates an output y from an input x. It is therefore also an operator (Sect. 1.5) and can consist of several suboperators. For each task, there is a different model that can perform this task.

The model is trained using training data, which contains a large set of inputs together with the corresponding correct outputs. To understand how this training works, one must first understand how the model can produce the outputs from the inputs.

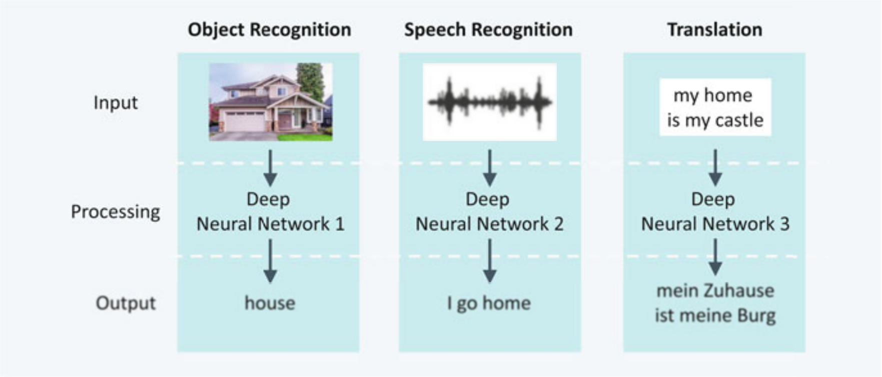


Fig. 3.4 In the simplest case, deep neural networks receive an input, do a computation, and produce an output. Image credits in Appendix A.3

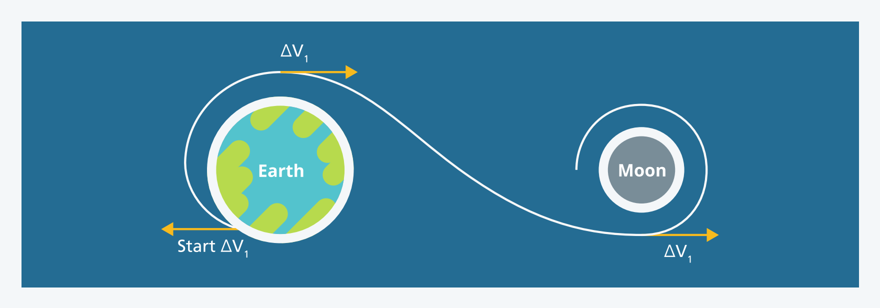


Fig. 3.5 The trajectory of a rocket from the Earth to the Moon can be calculated by using a physical model

In its original meaning, a model is a simplified representation of reality. A system whose complex details are not fully understood is represented in a simplified way. Examples are physical models, which show the trajectory of a rocket to the moon (Fig. 3.5), economic models, which predict the economic growth for the next year, or meteorological models, which predict temperature and precipitation for the next day.

In the field of data analysis, this concept has been generalized. There, a model is a description of the relationship between arbitrary inputs and outputs. This relationship is not known in detail at the beginning and is to be reconstructed from data. Modeling takes complex real-world problems and packages them into neatly formulated mathematical objects called models. They can be subjected to rigorous analysis and are more appropriate for computers. However, modeling is lossy: it cannot capture the whole complexity of the real world, and so simplifications must be made. The model designer must therefore decide which relationships and aspects to consider and which to ignore by testing models of varying complexity.

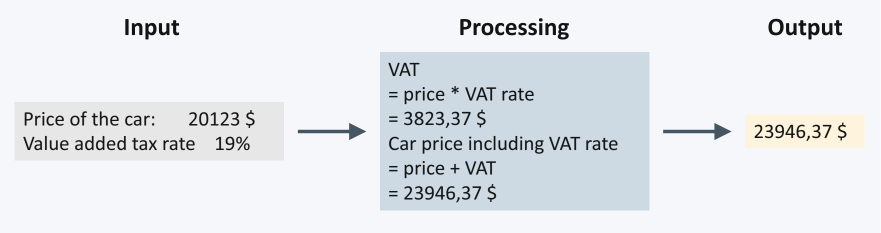


Fig. 3.6 Example of a simple algorithm for calculating VAT

3.2.2 Algorithms Process a List of Instructions Step by Step

Computers are capable of storing and processing data. Processing takes place by using a series of instructions, which are also stored on the computer and executed one after the other at a very high speed. Such a sequence of precisely defined instructions is called an algorithm.

Algorithms are also known from everyday life. In Fig. 3.6 the algorithm for calculating the price of a car with VAT is listed. But also a cooking recipe is an algorithm, if all ingredients are listed accurately and if the execution of all subtasks, e.g. frying or stirring, are defined precisely.

It is usually almost impossible to construct a model that produces the desired outputs right away. Instead, a model is modified (trained) step by step on the basis of data in such a way that it can produce the desired outputs better and better. In the remainder of the chapter, we have the task of implementing the **training of a model** as an algorithm on a computer. Looking at the introduction above, one might conclude that such an algorithm always performs the same operations and thus cannot learn. However, as we will see, even simple algorithms can extend their capabilities through learning processes. Such systems also act differently from conventional software because they do not only operate in a rigid binary manner—i.e., they know either “yes” or “no”—but also make decisions based on probabilities.

3.2.3 A Learning Problem: The Recognition of Digits

We would now like to illustrate the construction and training of a model by using a simple learning task. This is about the recognition of handwritten digits. Each digit is represented by a grayscale image consisting of 28×28 pixels. As shown in Fig. 3.7, the gray value of a pixel is a number between 0.0 (white) and 1.0 (black). The size of the figures has been normalized and centered in the middle of the image. This is the MNIST dataset with handwritten digits, which has been used as a benchmark dataset for many learning methods since 1998 (LeCun et al. 1998). So

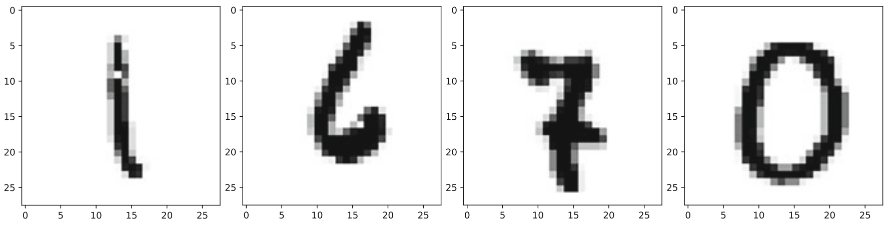


Fig. 3.7 Sample data from the MNIST dataset of photographs of handwritten digits. The goal is to match each image to one of the digit classes "zero", "one" to "nine"

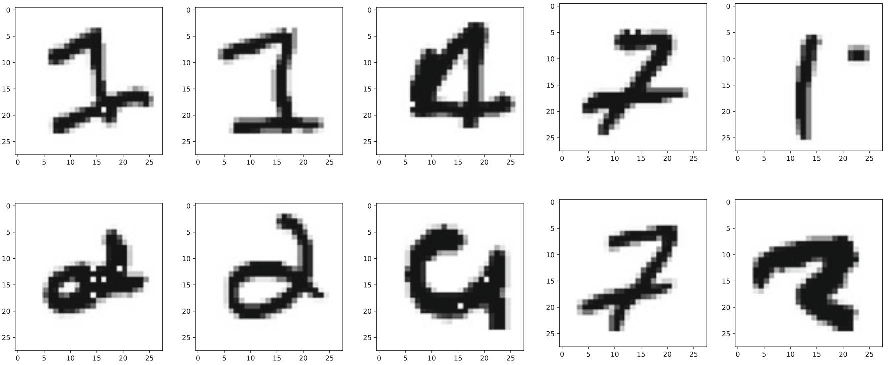


Fig. 3.8 Examples in the MNIST dataset where digit images are difficult to classify

here we have a classification problem with 10 different class names: "zero", "one", to "nine". This illustrates that classes can be denoted by arbitrary names.

Figure 3.7 contains example images from the MNIST dataset for four of the ten digits. Figure 3.8 shows some digits that are particularly difficult to categorize. It becomes clear that a number of mismatches exist, e.g. between "one" and "seven".

3.2.4 Vectors, Matrices and Tensors

Figure 3.10 shows a handwritten "two" and the corresponding numeric grayscale of a pixel image. Such a rectangular number packet is also called a matrix (plural: matrices). A 28 × 28 matrix has 28 rows and 28 columns. Each row of a matrix consists of a single 28 long line of numbers. Such a row of numbers is also called a vector.

The numbers in the matrix are in the range 0.0 (white) to 1.0 (black), and numbers in between correspond to a gray value. One could also consider a colored image of a digit. Here, for each pixel, the color can be specified by three numbers: the proportions of red, green, and blue. Thus, a colored image can be represented by three matrices, the red value, green value and blue value matrix. If you stack these

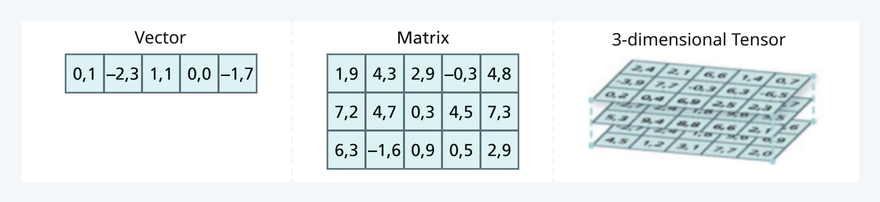


Fig. 3.9 Vector, matrix and tensor with arbitrary real numbers. The inputs to deep neural networks are usually encoded in this form. One can also add the dimension to the term. In the image, we have a 5 vector, a 3 × 5 matrix, and a 3 × 3 × 5 tensor

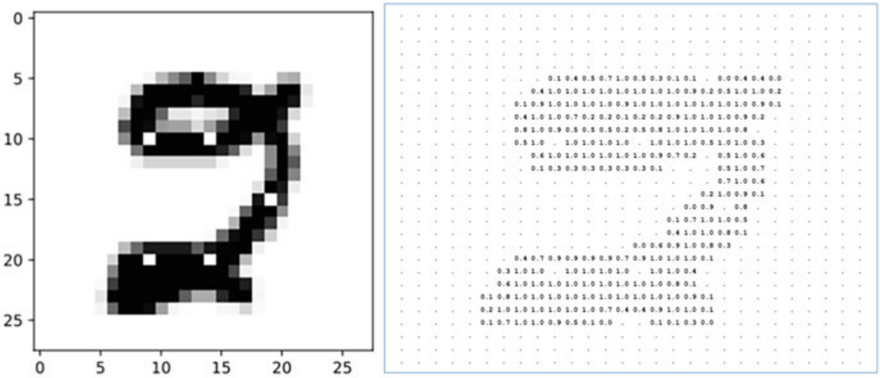


Fig. 3.10 Grayscale image of a handwritten "two" (on the left) and associated numerical values in the form of a 28 × 28 matrix. The numerical values range from 0.0 (white) to 1.0 (black). To improve the visual impression, the value 0.0 is represented as a dot

three matrices on top of each other, you get a cuboid-shaped three-dimensional number package, a three-dimensional "tensor".

In the original meaning, a tensor describes special quantities in physics (Abraham et al. 2012). Nowadays, a tensor is a collective name of scalars (simple numbers), vectors, matrices and higher dimensional number packages (Fig. 3.9). Thus, a vector is a one-dimensional tensor. A two-dimensional tensor is a matrix with rows and columns (i.e., two dimensions). A single number can be represented as a vector of length 1. The term tensor is obviously independent of the number of dimensions present.

The inputs to a model are, in the vast majority of cases, one or more tensors. In the case of digit recognition, the input is the 28×28 matrix of gray values (Fig. 3.10). The output is again an arbitrary tensor, in the case of classification a single integer. This class index is assigned to the names of the possible classes.

3.3 Learning a Relationship

3.3.1 Scheme for Learning: Model, Loss Function and Optimization

Almost all Machine Learning algorithms have the task to reproduce the relationship between the values of an input x and the corresponding values of the output y in the training data. Before we get to know a concrete learning algorithm, we want to sketch the general procedure of learning. This procedure is used in most Machine Learning methods and also for DNNs (Fig. 3.11):

- **Prediction**: The model predicts (calculates) an output $\hat{y}$ from the training input x. The model characteristics are controlled by a parameter vector w, whose values are randomly chosen at the beginning.
- **Loss**: The prediction value $\hat{y}$ is compared with the output value y in the training data. Then a loss, a real number that measures the difference between the two values, is calculated. The loss should be as small as possible.
- **Optimization**: changes the parameter w so that the loss becomes smaller. This is possible because the prediction of a model depends on the value of parameter w. Usually this is done in the form of many small changes of the parameter vector w. The procedure ends when there is no more improvement.

The adjustment of the parameter vector w is generally repeated very often with different training inputs until a model with small loss is found. This process is called training.

This is the basic scheme that can be used to determine a model that can predict the output y in the training data from the input x with little error. It has the same relevance to DNN as wheels being driven through the engine , transmission, clutch, and axle are relevant for the driving of a car (Sect. 1.5). Therefore, it is essential to understand the logic of this approach. The next section describes in detail how this scheme can be applied to the problem of digit classification.

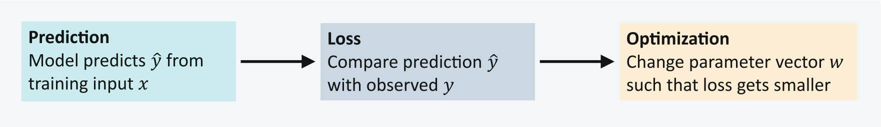

Prediction
Model predicts $\hat{y}$ from training input x
Loss
Compare prediction $\hat{y}$ with observed y
Optimization
Change parameter vector w such that loss gets smaller

Fig. 3.11 Basic steps for Machine Learning training

3.3.2 *Detailed Process of Learning*

The starting point is the training data whose i-th element $(x, y)^{(i)}$ consists of an input x (pixel matrix) and the corresponding output y (digit class). The prediction of the model operator is influenced by a parameter vector w filled with real numbers. When learning the classification, we can now proceed as following:

- **Prediction**: The model operator receives as input x a pixel matrix and a parameter vector w filled with real numbers. It does not directly compute a prediction $\hat{y}$ for the associated class y for x, but takes a deviation. It first predicts for x and w a probability vector p containing the probabilities of all possible classes (digits) (Fig. 3.12).
  A probability vector contains positive numbers between 0.0 and 1.0 whose sum is equal to 1.0. Each class is assigned a position (class index) in the probability vector p, i.e. the class “zero” the first position, the class “one” the second position, and so on. The trained model operator should now predict a high probability for the observed class y at the associated position. Since all probabilities sum up to 1.0, the remaining probabilities must be small. Thus, if the pixel matrix x corresponds to a “two”, the calculated probability value should be high for class “two”.
- **Loss**: To determine the quality of the prediction, the loss operator is used (❷ in Fig. 3.13), which calculates the value of the loss function. It evaluates a difference measure between the predicted probability vector p with the probabilities of the individual digits and the observed class y (e.g., “two”) with a loss value, a real number. The loss should be as small as possible (i.e., the probability of the observed digit as high as possible). Thus, this procedure does not predict the class directly, but rather evaluates the probability of the observed class y. In this way, the model calculates a continuous value that can be increased smoothly. This is essential if one wants to improve the prediction in small steps, e.g. by optimization procedures.

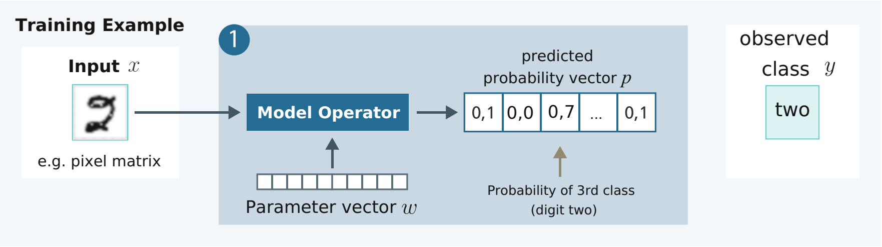


Fig. 3.12 For a training example $(x, y)^{(i)}$, the prediction computes a probability vector p with the probabilities of the different classes (digits) from the input x using the model operator. The prediction of the model depends on a parameter vector w. The goal is to modify the parameter vector to predict a high probability for the observed class y (e.g. “two”)

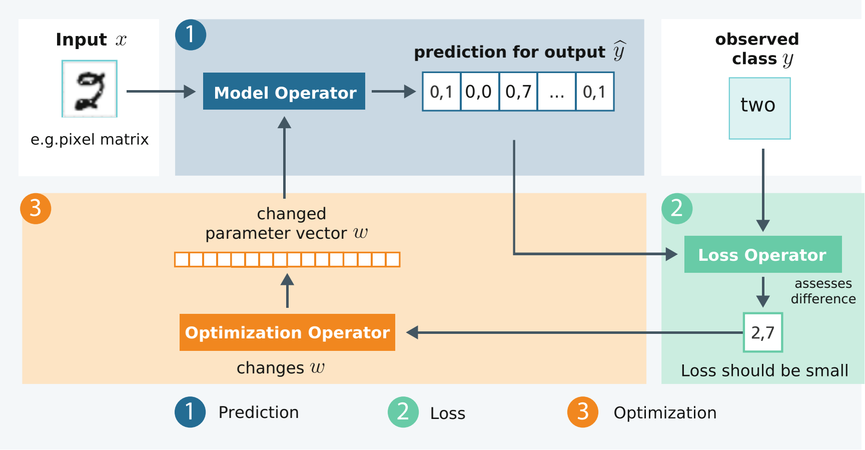


Fig. 3.13 For a training example $(x, y)^{(i)}$, the loss operator ❷ evaluates the difference measure between the probability vector p with the probabilities of the classes and the observed class y ("two"). The optimization operator ❸ analyzes the loss and modifies the parameter vector w so that the loss decreases (i.e., the probability of the observed digit increases)

- **Optimization**: An optimization procedure analyzes the loss function and the model. If one changes the parameter vector w, the predicted probability vector p also changes. Therefore, by changing the parameters, one can increase the probability of the observed class (digit). The optimization method (❸ in Fig. 3.13) determines how to change the values of the parameter vector w to reduce the loss of the model. Then, a usually minor change of w is carried out according to this scheme.

The loss is usually only minimally reduced with such an optimization step. Therefore, the ❶→❷→❸ step sequence must be repeated very often to actually arrive at a small loss. The optimization stops when there are no more improvements. The individual calculation steps can be realized by operators (Sect. 1.5) with suitable input and output tensors.

Through the parameter vector, the model has the ability to represent a rich repertoire of behaviors. The idea behind Machine Learning is to get the detailed information about the application from the data. Instead of manually constructing a sophisticated model, one specifies a skeleton of a model, that is, a model with parameter values not yet specified. The parameter vector typically includes several million to billions of parameter values. It is thus capable of capturing very complex relationships between the input and output sensors. With suitable and sufficient data, one can then run the optimization procedure described previously to adapt the parameters of the model and adjust the behavior of the model to the data.

Conceptually, this approach allows us to represent the complexity of the real world not by the model structure alone, but by the parameters that can be automatically determined from the data. The almost magical main part of learning is that—if done correctly—the trained model allows good predictions even for data that were not used for training. This is called generalization and is explicitly or implicitly at the heart of any Machine Learning algorithm.

A very basic model and the associated loss function will now be formulated in full detail.

3.4 A Simple Model: Logistic Regression

In order to simplify things, we first ignore the two-dimensional arrangement of the pixels in the grayscale image of the digits. We form a vector from the pixel matrix by concatenating all rows of the pixel matrix. This vector is called x. It has the length $784 = 28 * 28$ and serves as input for our learning procedure.

We now need a model that calculates the desired output from our input vector x, i.e. the digit corresponding to the pixel matrix. In this case, there are exactly $k = 10$ possible output values. Such a model is also called a classifier and the k different possible outputs "zero", ..., "nine" are the names of the classes.

As a concrete algorithm we choose here a simple approach: For each of the k possible classes we calculate a score u_i. The higher this score u_i is, the more plausible is the corresponding i-th class.

3.4.1 Calculation of a Score

In our simple model, we assume that each pixel has a separate influence on the score u_i of the i-th class. To get an intuition for this, we consider Fig. 3.14. Here we show some training examples of the classes "zero" and "one". In the lower part, the average gray values are for the images with a "zero" and for the images with a digit not equal to "zero". It can be clearly seen that the black pixels of the "zero" are mostly located in a circular area (bottom left) and the black pixels of the other classes are distributed in the middle area (bottom center). Each pixel is to be used as an independent indicator for a class.

You can see that high values in the circular area and low values in the center indicate a "zero". The pixels in the outer area have hardly any significance. The model should find these pixel weightings automatically.

Based on these considerations, we would like to assign a weight value $w_{i,j}$ to each pixel j for each of the classes $i = 1, \ldots, 10$. At the beginning, we know nothing about the weight values and assign randomly chosen initial values to them. To obtain a score from for a training example (x, y), e.g. u_1 for the first class "zero", we multiply the gray value x_j of the j-th pixel from the range [0.0, 1.0] in our input vector x by the associated weight $w_{1,j}$. We perform this calculation for each of the 784 gray values of the input vector and add the results. Finally, we add another value b_1, which does not depend on any pixel. This causes a global increase (or

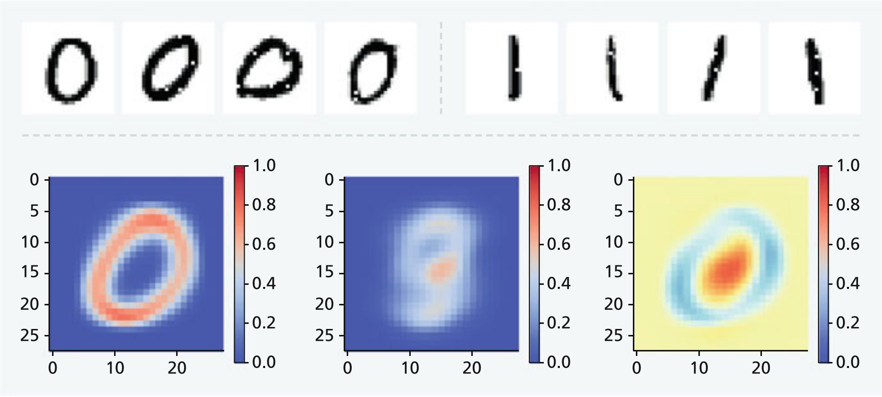
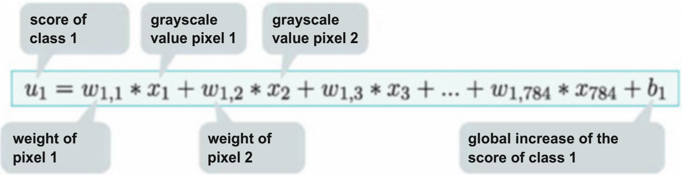


Fig. 3.14 The top two rows contain some training examples of the classes “zero” and “one” from MNIST. The bottom row contains the average values of the pixel values for class “zero” (bottom left) and those of the other classes (bottom center). On the bottom right is the difference of the average values. High pixel values in the blue area of the difference and low pixel values in the red area indicate a “zero”

decrease) in the score u_1 for the first class, regardless of the inputs, and can improve the prediction of the model. For u_1 of the first class “zero” thus yields:



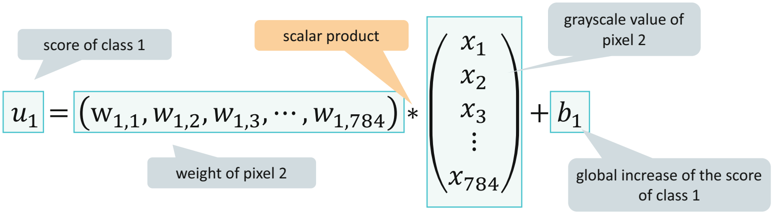
$$u_1 = w_{1,1} * x_1 + w_{1,2} * x_2 + w_{1,3} * x_3 + \ldots + w_{1,784} * x_{784} + b_1$$

The result of this calculation is a single real number u_1, the score of the first class.

If $y = i$ is the observed class of input x, then:

- the score u_i of this class should be high.
- all other scores should be as small as possible.

The weight $w_{1,j}$ is the influence of the j-th pixel on the score of the 1st class. We do not know this influence at the beginning and we want to determine it during learning. Therefore, the number $w_{1,j}$ is changed during learning to improve the classification: $w_{1,j}$ is a parameter. From the discussion of Fig. 3.14, it is clear that for the class “zero”, a pixel x_i in the annular region should be assigned a high positive $w_{1,i}$ value, while a pixel x_j in the center should be assigned a high negative $w_{1,j}$ value. Altogether, the numbers $(w_{1,1}, w_{1,2}, w_{1,3}, \ldots, w_{1,784})$ form a vector denoted w_1. The previous calculation of u_1 can then alternatively be written in vector notation:



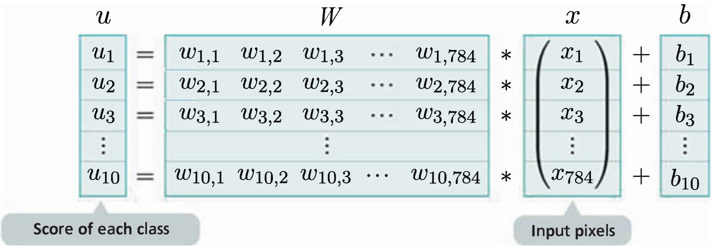
$$u_1 = (w_{1,1}, w_{1,2}, w_{1,3}, \cdots, w_{1,784}) * \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{784} \end{pmatrix} + b_1$$

score of class 1
scalar product
grayscale value of pixel 2
weight of pixel 2
global increase of the score of class 1

This way of multiplying the two vectors w_1 and x is also called scalar product, because the result is a single number (scalar).

3.4.2 *The Simultaneous Calculation of All Scores*

This calculation can be done for each of the 10 possible classes ("zero", . . . , "nine"). The weights $w_{i,j}$ of the j-th pixel for the i-th class are of course different in each case. Therefore, the separate calculation of 10 different scores $u_1, \ldots, u_{10}$ for each class is obtained.



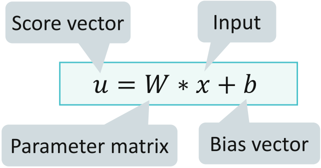
$$\begin{array}{ccccccccccc} u & & & & W & & & & x & & b \\ u_1 & = & w_{1,1} & w_{1,2} & w_{1,3} & \cdots & w_{1,784} & * & x_1 & + & b_1 \\ u_2 & = & w_{2,1} & w_{2,2} & w_{2,3} & \cdots & w_{2,784} & * & x_2 & + & b_2 \\ u_3 & = & w_{3,1} & w_{3,2} & w_{3,3} & \cdots & w_{3,784} & * & x_3 & + & b_3 \\ \vdots & & & & \vdots & & & & \vdots & & \vdots \\ u_{10} & = & w_{10,1} & w_{10,2} & w_{10,3} & \cdots & w_{10,784} & * & x_{784} & + & b_{10} \end{array}$$

Now it becomes clear that the numbers $w_{i,j}$ form a rectangular number packet, thus can be treated as a matrix W with 10 rows and 784 columns. Likewise, the u_i, b_i and x_i can be combined to form vectors of lengths 10 and 784, respectively:

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_{10} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_{10} \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ \vdots \\ x_{784} \end{pmatrix}$$

3.4.3 Affine Transformation

Thus, the previous calculation of the scores $u_1, \ldots, u_{10}$ can be written in the following matrix notation:



This is the multiplication of the vector x by the matrix W. The vector b is then added to the result and forms the vector u. If i is the class index of the observed class of the input x, the thus calculated score u_i of this class should be high and the other scores lower.

The parameter matrix W contains all influence factors of the individual pixels on the score of the respective digits, in total $10 * 784 = 7840$ parameters. The parameter vector b is called Bias. These 10 parameters globally increase/decrease the score regardless of the inputs. In total 7850 parameters are available to adjust the score values for the individual inputs.

For simplicity, all parameter components are combined into a global parameter vector w by concatenating the parts

$$w = (W, b)$$

The multiplication of a vector by a matrix is called linear transformation. When subsequently another vector is added, we called this an affine transformation, which is the most commonly used operation in deep neural networks. Today's computers have optimized instructions that can perform this operation extremely fast.

This transformation has interesting properties. In Fig. 3.15 two lines and a circle were drawn on the left. These were transformed with the affine transformation

$$u = W * x + b = \begin{pmatrix} -0.8 & -0.2 \\ -0.5 & 0.4 \end{pmatrix} * x + \begin{pmatrix} 1.0 \\ 0.5 \end{pmatrix}$$

The transformed points can be seen in the right graphic of Fig. 3.15. The vector b causes a shift to the upper right and the matrix W causes a rotation and stretching. It is important to note that an affine transformation always transforms all points on a straight line into points that are again on a straight line, as can be seen for the red and the blue line.

Fundamentally, the affine transformation causes a shift, a rotation as well as a compression or stretching. As can be seen from the transformation of the circle into an ellipse, the extent of the compression/stretching depends on the direction. These statements are also valid for affine transformations with arbitrarily large matrices

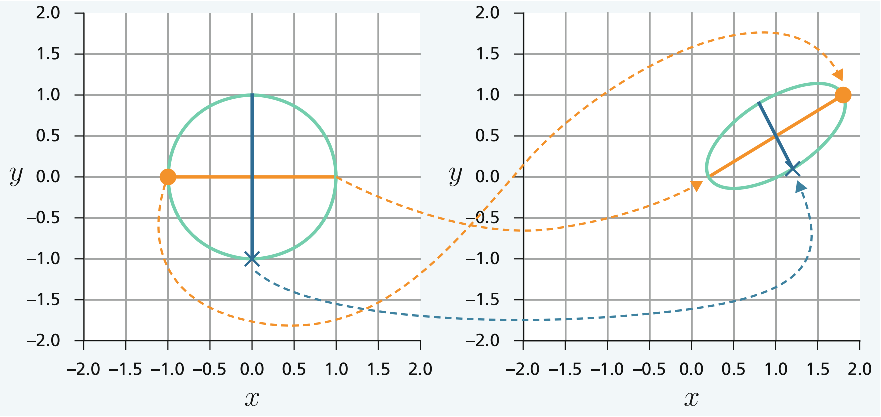


Fig. 3.15 Two lines and a circle in 2-dimensional space (on the left). Each point $x = (x1, x2)$ on the lines and the circle is transformed by the mapping $u = W * x + b$. The transformed vectors u were drawn on the right picture. It is essential that a straight line is always transformed into a straight line. Obviously a circle is transformed into an ellipse

and correspond to high-dimensional input or output vectors. Here, the length of the inputs may be different from the length of the outputs.

3.4.4 The Softmax Function Generates a Probability Vector

If y is the observed class of the input x, the score of this class should be high and the scores of the other classes should be lower. However, to obtain a learning algorithm, we need to specify how much higher the score of the correct class should be. Unfortunately, scores are difficult to interpret in terms of content. Therefore, the vector u of scores is transformed into a vector of probabilities that can be used to specify a plausible learning algorithm (Fig. 3.16).

A probability vector consists of positive real numbers between 0.0 and 1.0. Furthermore, the sum of these numbers must be 1.0. One can now convert any vector with positive and negative numbers, e.g. $u = (1.5, -1.2, 0.5)$, into such a probability vector in three steps. This is demonstrated in Fig. 3.17.

In the first step, one applies to each component of u the exponential function shown in Fig. 3.16 and one obtains a vector v of positive numbers. In the second step, the sum of the numbers in v is calculated. In a third step, one divides the components of vector v by this sum and obtains a probability vector p. This small algorithm is executed by the softmax function:

$$p = \text{softmax}(u)$$

The result $p = (p_1, p_2, \ldots, p_k)$ is always a probability vector.

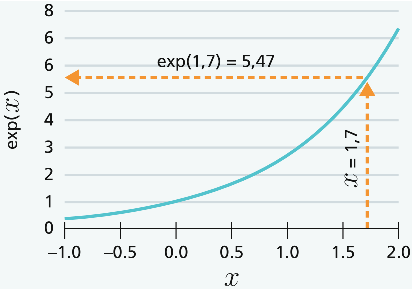


Fig. 3.16 The exponential function keeps increasing and has only positive numbers as a result

Fig. 3.17 An arbitrary vector u is transformed by the softmax function into a probability vector p

Input vector $u = (1.5, -1.2, 0.5)$.

1. Apply the exponential function (Fig. 3.16) to each element of u
   $v := \exp(u) = (4.48, 0.30, 1.65)$.
   v contains only positive numbers.
2. Calculation of the sum of elements in v:
   $4.48 + 0.30 + 1.65 = 6.43$
3. Division of the elements of v by the sum

$$p = \frac{(4.48, 0.30, 1.65)}{6.43} = (0.70, 0.05, 0.25)$$

The result is a probability vector p.

3.4.5 The Logistic Regression Model

We are now able to specifiy a simple classification model for digit recognition (Sect. 3.2.3). Performing the affine transformation $u = Wx + b$ and the softmax function $p = \text{softmax}(u)$ in sequence, we obtain a model that transforms the input vector x into a vector p of class probabilities. This is shown in Fig. 3.18. The predicted most plausible class $\hat{y}$ is obtained by selecting the class with the highest probability. The model is called a logistic regression model. According to a survey by Kaggle, the logistic regression model is the most commonly used Machine Learning technique (Kaggle 2017).

Thus, we now have a *model* (Sect. 3.3.1), which, for an input vector x (encodes the image) and the given parameters $w = (W, b)$, computes the probability vector of the individual classes (digits):

$$p = \text{softmax}(Wx + b)$$

The observed output y (e.g. “eight”) is assigned the class index (sequence number) of the class. For simplicity, one uses the class index instead of the class name when a numerical value is needed. For example, if the model has calculated the vector $p(x, w) = (0.02, 0.0, 0.08, 0.09, 0.22, 0.01, 0.0, 0.0, 0.57, 0.0)$ as the

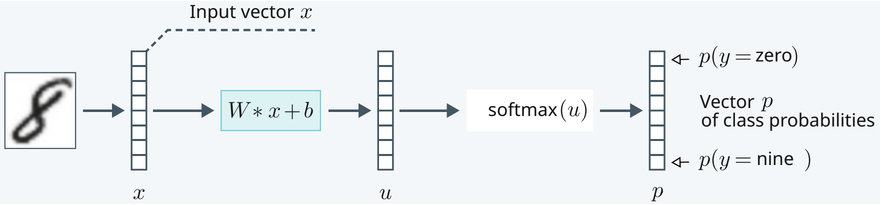


Fig. 3.18 Components of the logistic regression model. A vector p of class probabilities is computed from the inputs x via the affine transformation and the softmax function. The matrix W and vector b are parameters fitted to the training data

result, $p_y(x, w) = 0.57$ is the highest class probability, where y here is index 9 of class "eight". Here $p_y(x, w)$ is the y-th value of the probability vector.

3.5 Measuring Model Performance

3.5.1 A Criterion of Model Performance: The Likelihood of Complete Training Data

The probability $p_y(x, w)$ of the "correct" class y will never be exactly equal to 1.0 even after optimization. The question is therefore how to evaluate the quality of a probability vector $p(x, w)$ for an input vector x. A simple way would be to take the difference between $p_y(x, w)$ and 1.0 as a measure. However, it has been shown that such an approach incorrectly weights the elements of the probability vector. Such problems are avoided with an alternative evaluation measure. For this purpose, we examine the set of all $n = 60,000$ training examples in the training set

$$\text{TrainSet} = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(60,000)}, y^{(60,000)})\right\}$$

In this case, $x^{(m)}$ is the input vector (pixel values) and $y^{(m)}$ is the index (sequence number) of the associated class for the m-th training example. We can calculate the probability vector $p(x^{(m)}, w) = \text{softmax}(Wx^{(m)} + b)$ for each training example $x^{(m)}$ by using the model for the given parameters $w = (W, b)$. If $y^{(m)}$ is the corresponding correct class in the training set, the $y^{(m)}$-th component $p_{y^{(m)}}(x^{(m)}, w)$ is the probability of the correct class $y^{(m)}$ of the m-th training example $(x^{(m)}, y^{(m)})$.

To find the best parameters $w = (W, b)$, one can derive from statistical theory the requirement that the probability of the entire training set TrainSet should be as high as possible. This probability is the product of the probabilities of the observed classes of the elements of the training set. The product can be used when the examples in the training set were independently observed, i.e. did not influence each other directly.

Probability of the full training set

Probability of the class of the 60,000-th training example

$$p(\text{TrainSet}|w) = p_{y^{(1)}}\left(x^{(1)}, w\right) * \cdots * p_{y^{(60,000)}}\left(x^{(60,000)}, w\right)$$

Probability of the class of the first training example

The parameters $w = (W, b)$ can now be changed to maximize the probability of the entire training set. A successful adaption makes the observed classes of the training examples highly plausible for the training data on average. If one then applies the model to new input data, one can assume that the predicted probabilities for the actual class will also be high. The probability of the entire training set is also called *likelihood*. The search for the parameter with the largest likelihood is the *maximum likelihood principle*. This is a fundamental guideline for determining model parameters in modern statistics (Efron& Hastie 2016, p. 38).

Maximum Likelihood Principle
The parameters w of the model should be changed such that the probability of the entire training set is maximized.

3.5.2 How to Measure Learning Success: The Loss Function

If you calculate the product of 60,000 small numbers (e.g., 0.1) for probability of the entire training set, the result becomes very small (e.g., 0.1^{60000}, i.e., a number with 60,000 zeros after the decimal point). Such a number is very difficult to handle on a computer. Therefore, one transforms the previous formula with the logarithm $\log(\cdot)$. This function preserves the "$>$"-relationship, i.e. if $a > b$ so is $\log a > \log b$. Therefore, one can also optimize the transformed formula and obtain the same optimum as for the initial formula. The transformed computation has the following form:

$$\log p(\text{TrainSet}|w) = \log p_{y^{(1)}}(x^{(1)}, w) + \cdots + \log p_{y^{(60000)}}(x^{(60000)}, w)$$

Here the product of the probabilities was replaced by a sum, because $\log(a * b) = \log a + \log b$. It has also become a common practice to use minimization instead of maximization. This simply requires another multiplication by -1. Thus we have the loss function $L(w)$, which evaluates the deviation between the predicted classes (class probabilities p) and the observed classes y for the complete training set and the given parameters $w = (W, b)$ (Sect. 3.3.1):

$$L(w) = -\log p(\text{TrainSet}|W, b)$$

If the forecast of the model $\hat{y} = f(x; w)$ is a real number, the normal distribution with expected value $f(x; w)$ and a fixed variance σ^2 is often chosen to describe the distribution of possible output values around the expected value:

$$p(y^{(i)}|x^{(i)}, w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y^{(i)} - f(x^{(i)}; w)\right)^2}{2\sigma^2}\right)$$

Calculating the probabilities of the entire training set of n elements and applying the negative logarithm, we obtain the least square error criterion as the loss function

$$L_{\text{least-squares}}(w) = \left(y^{(1)} - f(x^{(1)}; w)\right)^2 + \cdots + \left(y^{(n)} - f(x^{(n)}; w)\right)^2$$

Fig. 3.19 The least-squares loss function for predicting continuous values

If the probabilities of all correct classes are 1.0, the loss function takes its minimum value of 0.0; otherwise it has a positive value. This loss function is called log-likelihood loss function or maximum entropy loss function. This loss function is used by the large majority of classification models. Figure 3.19 describes for completeness the least-squares loss function for models with continuous outputs, which also results from the maximum likelihood principle.

3.5.3 Illustration for Two Classes and Two Input Features

One can greatly simplify the classification problem by considering only two classes (0 and 1) and using only two input features x_1 and x_2. In Fig. 3.20, the training examples of the training set are represented by their input features and class: the characters 0 and 1 symbolize the elements of the two classes.

At the beginning, the parameter vector w is chosen randomly. In Fig. 3.20, the corresponding classifier (logistic regression) is denoted by a red and a blue region: in the red region, the probability of 1 is larger than 0.5, and in the blue region, the probability of 0 is larger than 0.5. The separating line between the two regions is a straight line. However, a number of the observed training examples lies on the "wrong" side of the line. These misclassified training examples are marked in magenta. For these training examples, the probability value $p_y(x, w)$ is less than 0.5 and the total value of $p(\text{TrainSet}, w)$ is correspondingly low.

Figure 3.21 shows the situation with the same input data for a logistic regression model with a different parameter w^+. Here the cutoff line of the model is very well adjusted to the classes. There are only a total of three training examples that are on the "wrong" side: one 0-training example and two 1-training examples. For all other training examples, the probability value $p_y(x, w^+)$ is larger than 0.5 and the value of $p(\text{TrainSet}, w^+)$ is correspondingly high. In fact, the best logistic regression classifier is shown in Fig. 3.21.

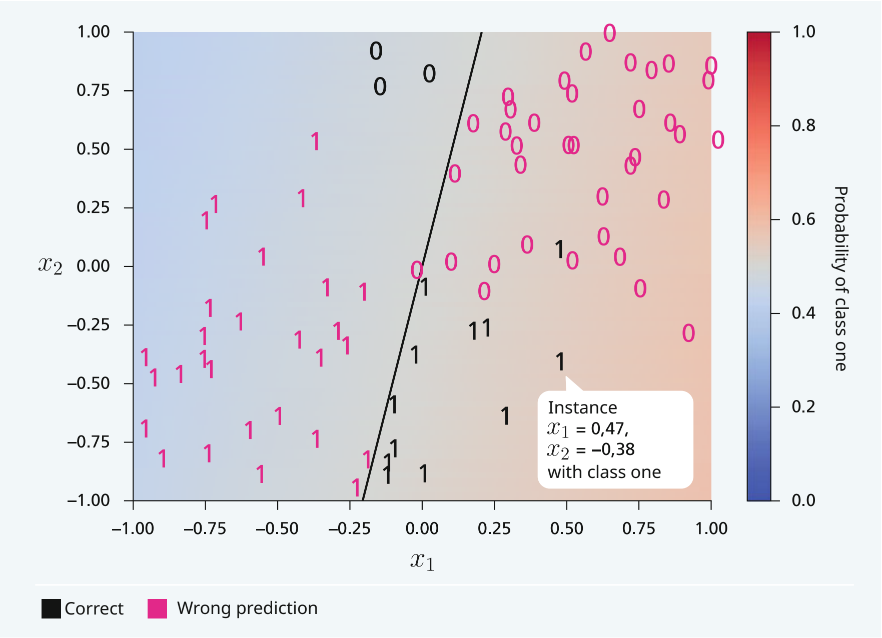


Fig. 3.20 The classification model separates the input level into two areas. In the reddish region, the probability of class 1 is larger than 0.5 and all training examples are classified as 1. In the blue area, the probability of class 0 is larger than 0.5 and all training examples are classified as 0. Many training examples are misclassified (magenta), i.e. the probability of the correct class is less than 0.5

As seen in Fig. 3.21, one cannot guarantee that every training example will be placed in the correct class. It can happen that the probability of the correct class of a single training example drops because by changing w one can improve the probabilities of several other training examples.

3.6 Optimization, or How to Find the Best Parameter Values

We now need to find an optimization algorithm that modifies the 7850 parameters of the parameter vector w of the model so that the loss function $L(w)$ becomes as small as possible.

Minimizing the loss function is comparable to the situation of a mountain climber who wants to descend as quickly as possible to the lowest point of the valley in the fog (Fig. 3.22). He does not know where the valley is, and can only overlook his nearest surroundings. The only thing he can see in his immediate surroundings is where it goes downhill. Therefore, he always chooses the direction of steepest descent at the point where he is (here we assume that falling down is not a problem).

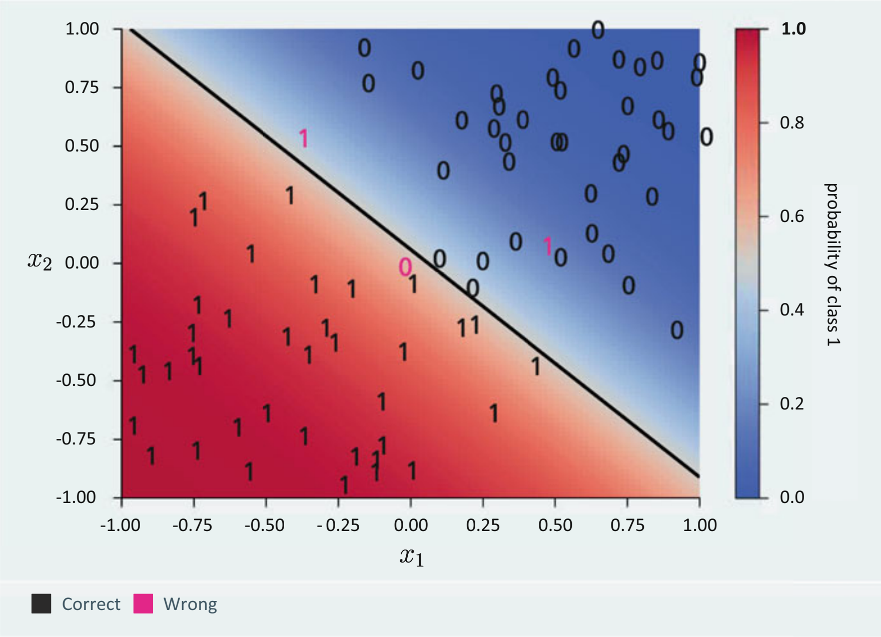


Fig. 3.21 The classification model separates the red area, where all training examples are classified as 1, from the blue area, where all training examples are classified as 0. There are only 3 misclassified training examples in total (magenta). For all other training examples, the probability of the correct class is greater than 0.5

Fig. 3.22 Minimizing the loss function is like descending a mountain in the fog. Image credits in Appendix A.3

3.6.1 The Gradient Indicates the Direction of the Steepest Ascent

In our optimization problem, the loss function $L(w)$ can also be interpreted as a mountain landscape, but in a high-dimensional space, since w has a total of 7850 components. In mathematics, the term **gradient** denotes a vector pointing in the direction of the steepest ascent of a function. The steepest descent is then in the direction of the gradient multiplied by -1.0. In Fig. 3.23 such a function landscape is drawn. The aim is to reach the lowest point coming from the mountain (red). The black lines show the gradient in the respective position. The procedure follows the negative gradient for a small distance. Then the gradient has to be determined again, because the direction of the steepest ascent has changed. Depending on the starting point, this gradient descent optimization can end up in different sinks.

3.6.2 The Gradient for Several Dimensions

But how can we now determine the gradient for functions in several dimensions? Let us first recall the school lessons in which the derivative of a function was discussed. In Fig. 3.24 a one-dimensional function and its derivative are plotted. The derivative $\frac{\partial f(x)}{\partial x}(\tilde{x})$ of the function, for example, at the point $\tilde{x} = -1.8$, is the slope of the tangent line of the triangle drawn in the graph, going a small distance s from $\tilde{x}$ to the right. The tangent is a line that touches the curve at the point $\tilde{x}$. In the graph, the distance s is the lower side of a triangle (green). The slope here is negative and forms the vertical side of the triangle (red). The diagonal side of the triangle (blue) is just the tangent of the curve at the point $\tilde{x}$. You can see that the derivative for the value $\tilde{x}$ is negative, so the function becomes smaller to the right. If you change

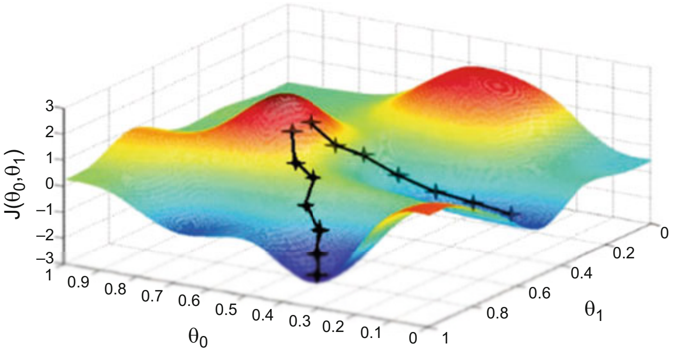


Fig. 3.23 Descent in a landscape defined by a function. The black lines show the direction of the steepest descent in the particular point

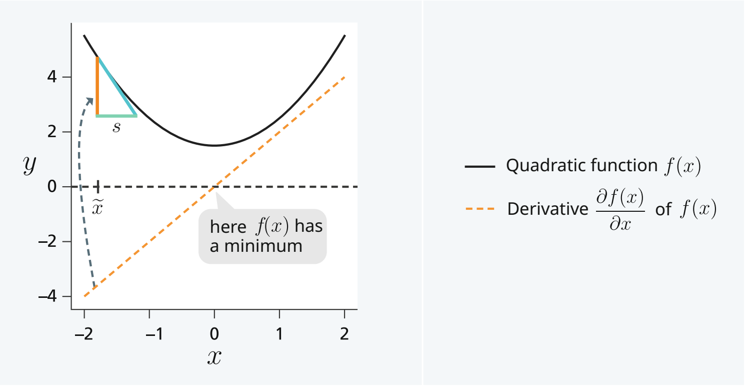


Fig. 3.24 Quadratic function $f(x)$ and its derivative $\partial f(x)/\partial x$. The derivative at a point $\tilde{x} = -1.8$ is the value $s * \frac{\partial f(x)}{\partial x}(\tilde{x})$ by which the function increases as one moves a small distance s from $\tilde{x}$ to the right. The derivative in $\tilde{x}$ (red) is negative, so the function decreases

the position $\tilde{x}$, the derivative changes too. At the point $\tilde{x} = 0.0$ the function has a horizontal tangent and the derivative 0. Here is a minimum of the function. On the right side of the graph, the function increases and the derivative is positive.

Given a function $L(w)$ which has a vector $w = (w_1, \ldots, w_k)$ as argument, this function depends on several variables, the components $w_1, \ldots, w_k$ of the vector. For each component w_i of the vector, one can compute a derivative at a point $\tilde{w}$, leaving the other components unchanged. This is the tangent $\frac{\partial L(w)}{\partial w_i}(\tilde{w})$ of the function $L(w)$ in the i-th dimension of the k-dimensional representation of $L(w)$. This derivative $\frac{\partial L(w)}{\partial w_i}(\tilde{w})$ is also called partial derivative, because only one of the possible variables is changed at a time. Here $\partial L(w)$ denotes the function for which a derivative is computed. ∂w_i is the vector component whose slope is computed, and the argument $\tilde{w}$ is the value of the vector for which the partial derivative is calculated.

For clarification Fig. 3.25 shows the function $L(w)$ with the vector argument $w = (w_1, w_2)$. The function has its minimum in the red point and increases in all directions. The low values are marked by a yellow hue, the high values by a blue hue. Also drawn in are “contour lines”, i.e. lines of the same function value. For an example point $\tilde{w} = (5.0, 2.5)$ the partial derivative $\frac{\partial L(w)}{\partial w_1}(\tilde{w})$ is drawn as an orange arrow and the partial derivative $\frac{\partial L(w)}{\partial w_2}(\tilde{w})$ as a cyan arrow. If these two arrows are combined to a vector we get the gradient

$$\frac{\partial L(w)}{\partial w}(\tilde{w}) = \left(\frac{\partial L(w)}{\partial w_1}(\tilde{w}), \frac{\partial L(w)}{\partial w_2}(\tilde{w}) \right)$$

a vector in the direction of the steepest ascent of the function at the point $\tilde{w}$. It is represented by a red arrow. The gradient is always perpendicular to the contour line.

But for the minimization we need the direction of the steepest descent. This is the dark green vector opposite to the gradient, also called negative gradient. The

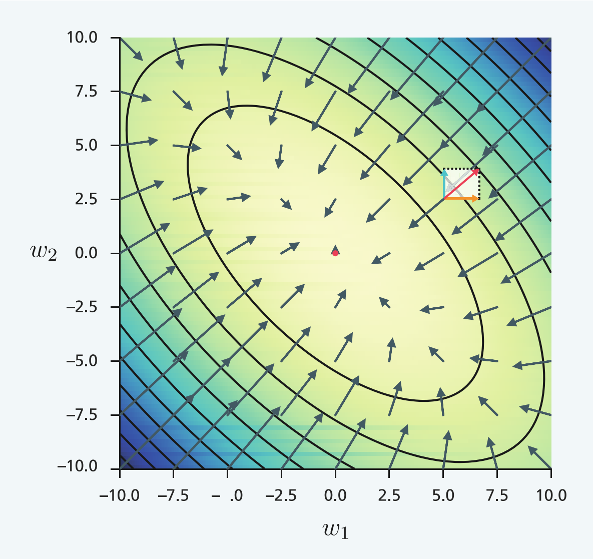


Fig. 3.25 Representation of a function $L(w)$ of two variables $w = (w_1, w_2)$. Small values are displayed on a yellow background and large values on a blue background, and the minimum is at the red point. The gradient (red arrow) is the vector sum of the partial derivatives of $L(w)$ with respect to w_1 (orange arrow) and w_2 (green arrow). This vector points in the direction of the steepest ascent. The negative gradient is the vector opposite to the red arrow (dark green arrow), the direction of steepest descent. All other negative gradients were calculated analogously

negative gradients at other points were calculated in the same way. It can be seen that by using the negative gradient, one can always get to the direction of the minimum from all positions. However, the negative gradient does not point exactly to the minimum, but after a certain distance the direction to the minimum usually changes. At the minimum (red point) the negative gradient is 0, i.e. both partial derivatives are also 0.

3.6.3 *The Gradient of the Loss Function*

To modify the loss function with respect to w, one has to calculate the following derivative:

$$\frac{\partial - \log p(\text{TrainSet}|w)}{\partial w} = -\frac{\log p_{y^{(1)}}(x^{(1)}, w) + \cdots + \log p_{y^{(60000)}}(x^{(60000)}, w)}{\partial w}$$

Today, however, it is no longer necessary to determine the derivative formulas manually. In fact, many programming environments for neural networks are able to calculate the formulas for the derivatives symbolically. Once the model and the loss function have been specified the actual work is already done. One only has to select the optimizing method from a list of available methods and the learning algorithm can run on the training data. Nevertheless, it is important to understand the principles of the optimization procedure.

3.6.4 Stepwise Minimization by Gradient Descent

If the gradient of the loss function can be calculated at each point, the minimization of the loss function can now be performed in the following steps—as shown in Fig. 3.26 for 2 dimensions. The parameter vectors $w^{(0)}, w^{(1)}, w^{(2)}, \dots$ are generated successively until the procedure stops.

1. The parameter vector $w^{(0)}$ in step 0 is filled with randomly drawn values.
2. If $w^{(i)}$ is the current parameter vector in step i, the negative gradient of the loss function $-\frac{\partial L(w)}{\partial w}(w^{(i)})$ is calculated for the parameter vector $w^{(i)}$. This negative gradient is itself a vector and points in the direction of steepest descent of the loss function in position $w^{(i)}$.

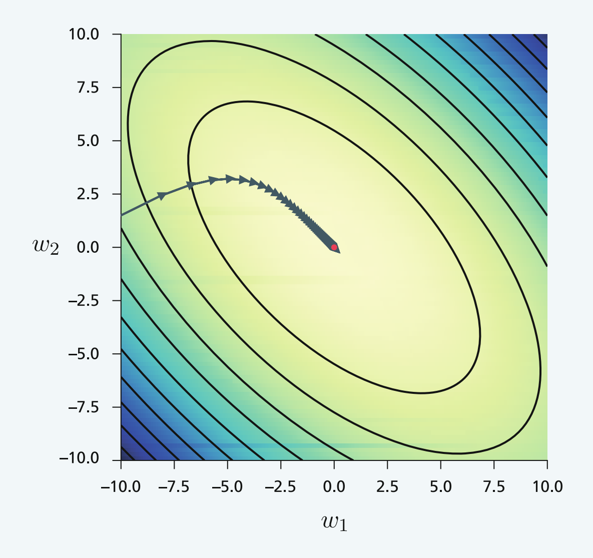


Fig. 3.26 Minimization of the loss function $L(w)$ by gradient descent in the direction of the steepest descent of the loss function. The procedure starts with a randomly chosen initial value $w^{(0)}$

3. The next parameter vector $w^{(i+1)}$ is now computed by

$$\boxed{\underset{\text{new parameter vector}}{w^{(i+1)}} = \overset{\text{prior parameter vector}}{w^{(i)}} - \underset{\text{learning rate}}{\lambda} * \underset{\text{gradient of loss function}}{\frac{\partial L(w)}{\partial w}(w^{(i)})}}$$

   Here λ is a small number, the learning rate. This formula changes the current parameter vector $w^{(i)}$ a small bit in the direction of the steepest descent and the loss becomes slightly smaller.
4. If the difference between $L(w^{(i+1)})$ and $L(w^{(i)})$ is very small, the procedure stops and $w^{(i+1)}$ is output as the parameter with minimum loss. Otherwise, the procedure continues with step 2. Because the negative gradient for $w^{(i+1)}$ is different from that for $w^{(i)}$, it must be recomputed.

3.6.5 The Learning Rate Sets the Length of an Optimization Step

The learning rate λ is a small positive number that must be specified. It determines how fast the learning procedure reaches the minimum. If λ is very small, the procedure needs a lot of steps and has to calculate many gradients to reach the minimum. If λ is too large, the next step may pass over the region of the minimum, the procedure oscillates, and the minimum value is not reached (Fig. 3.27). The learning rate is a control value of the optimization procedure, whose value must be specified beforehand. However, it cannot be simply optimized like a model parameter. There exist quite a number of such quantities, which are called hyperparameters. The determination of their values is discussed in Sect. 4.8.1.

3.6.6 Minibatch Gradient Descent Needs Less Computation

To determine the gradient, the loss function must be evaluated. To compute the loss function for all 60,000 training examples $(x^{(m)}, y^{(m)})$ of the training set, one must determine the model output softmax$(W * x^{(m)} + b)$ for all these training examples. Since many gradient steps must be performed during training, this leads to a high computational cost.

Alternatively, one can randomly split the training set into small subsets of m training samples (e.g., $m = 30$). These subsets are called minibatches. Then the gradient for a minibatch is calculated and a gradient step is performed. To obtain a gradient, the model output softmax$(W * x^{(m)} + b)$ has to be computed only

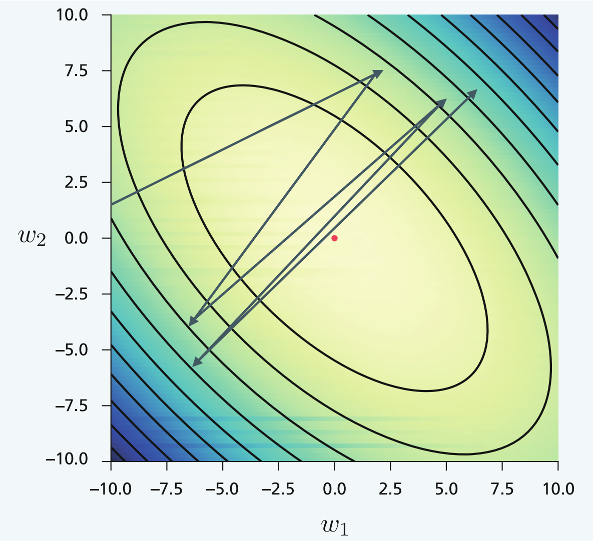


Fig. 3.27 Minimizing the loss function $L(w)$ by gradient descent. If the learning rate λ is too large, the computed parameter vector oscillates around the minimum and does not converge

for the training examples of the minibatch, so the computational effort decreases dramatically. The next gradient step is then computed using a different minibatch, and so on. Once all minibatches have been used, one epoch has been completed, i.e. all training examples of the training set have been used once. After that, the algorithm forms new random minibatches from the training set, for which the algorithm is repeated.

Figure 3.28 shows a sequence of training steps with minibatches for our example. It becomes clear that the training steps are strongly overlaid by random influences, since the minibatch gradient can deviate substantially from the "correct" gradient with all training data. Nevertheless, the minibatch gradient is correct on average and usually also leads to the minimum.

The minibatch gradient calculation has three advantages:

- The calculation of the gradient requires orders of magnitude less computational effort.
- The method reaches the vicinity of the minimum value much faster overall.
- The gradient deviates randomly from the actual gradient, since only a small subset of the training examples was used for its computation. Thus, the gradient is affected by random influences, but is correct on average. As a result, the method is often able to jump out of local sinks, i.e. to leave local minima.

For these reasons, minibatch gradient descent is used almost everywhere today. This method has made training with millions of training examples possible in the first place. Because of the random variations of the gradient, it is also called stochastic gradient descent.

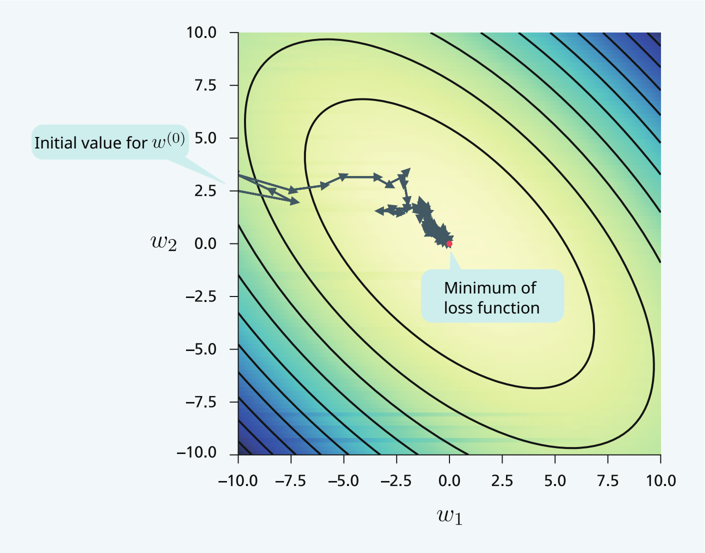


Fig. 3.28 When during stochastic gradient descent the negative gradient is calculated only for a minibatch, it deviates randomly from the "correct" negative gradient for all training data. As a result, the minimization path is overlaid by strong random fluctuations. Nevertheless, the stochastic gradient descent method comes close to the optimum

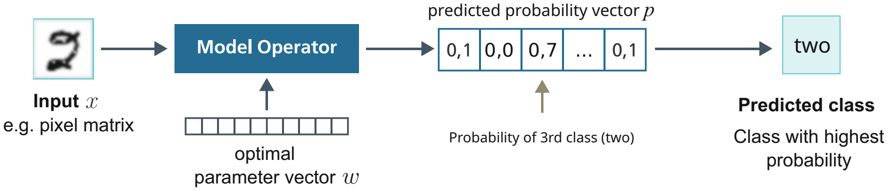


Fig. 3.29 Applying the logistic regression model to new data. Here, the parameter vector optimized by training is used and a prediction is performed, which calculates the probability vector for the classes. The class with the highest probability is chosen as the output

3.6.7 Applying the Model to New Data

Once the parameter vector w of the model has been adjusted by optimization to maximize the predicted probability of the observed classes y for the training data, the model can be applied to new inputs. This is illustrated in Fig. 3.29. Here, only one model prediction is performed with the input. The class with the highest class probability is chosen as the output.

3.6.8 Checking the Accuracy on the Test Set

To obtain an estimate of the quality of the model, one can calculate the accuracy of the classification. This accuracy is the number of correctly classified sample data divided by the number of all classified sample data.

The goal is to determine an accuracy that is also transferable to new data. The model should therefore correctly assign new inputs that were not available during training. This ability is called generalization and is the actual success criterion of a Machine Learning procedure. Therefore, the accuracy must not be determined on the training data, since the model was optimized there in such a way that as many training examples as possible are already correctly classified.

Therefore, one has to calculate the accuracy on data that was not used for training. It is now common practice to randomly split the available annotated data into two subsets (Fig. 3.30): the training set and the test set. Often one chooses the test set to include about 20% of the data samples. Then, on the one hand, the decrease of accuracy in training due to a smaller training set is not very large, and on the other hand, the estimate of accuracy is sufficiently precise. The elements of the training set are called training examples and the elements of the test set are called test examples.

Figure 3.31 shows the accuracy on the training and test set of the MNIST data (Sect. 3.2.3) after each epoch of optimization. In the beginning, the accuracy increases rapidly and grows oscillatorily higher and higher to about 0.924, i.e., out of 100 digits, slightly more than 7 are misclassified. The oscillations are caused by the minibatch gradient descent, which leads to random deviations in the gradients. As expected, the accuracy on the test set is significantly lower than on the training set. The method requires under 10 seconds for the calculations on a laptop.

3.6.9 Precision and Recall for Classes of Different Size

In some situations, accuracy may be a measure that is misleading. Suppose that in our application there are only two classes and the second class applies to only 1% of the test examples. Then a model achieves 99% accuracy if it assigns all test examples to the first class. Despite the high accuracy, the classification model is obviously unusable.

Therefore, two additional quality measures have been defined for each class: the precision of class k and the recall of class k. The F-value of class k is the harmonic mean between precision and recall. The exact definitions can be found in Fig. 3.32.

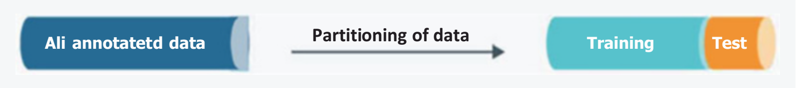


Fig. 3.30 Decomposition of annotated data into training data and test data

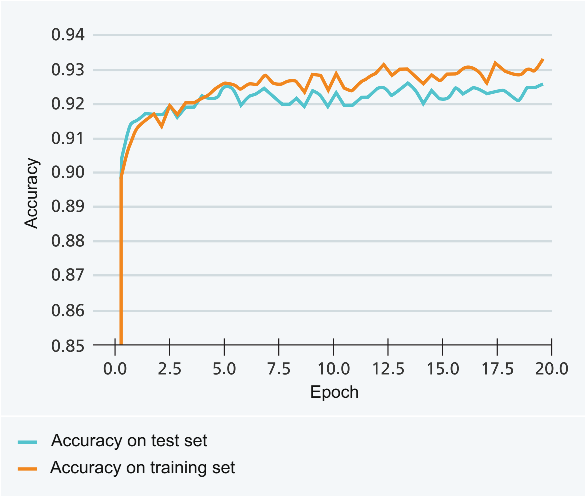


Fig. 3.31 Accuracy on the training and test sets for the epochs of stochastic optimization for the logistic regression model and MNIST data

$$\begin{aligned}
\text{accuracy} &= \frac{\text{number of correctly classified test examples}}{\text{number of all test examples}} \\
\text{precision}_k &= \frac{\text{number of all test examples correctly assigned to class } k}{\text{number of all test examples assigned to class } k} \\
\text{recall}_k &= \frac{\text{number of all test examples correctly assigned to class } k}{\text{number of all test examples belonging to class } k} \\
F\text{-value}_k &= 2 * \frac{\text{precision}_k * \text{recall}_k}{\text{precision}_k + \text{recall}_k}
\end{aligned}$$

Fig. 3.32 Common performance measures for classification. Accuracy considers all classes together, while precision_k, recall_k, and $F\text{-value}_k$ refer to the k-th class

If, for example, the precision of a class is 0.90, then 90% of the test examples that were assigned to the class are also correctly assigned. However, it is not yet known which part of the test examples of the class was also “found” by the classifier, i.e. was assigned to the class. This percentage is just the recall. How high the precision and the recall should be, depends on the application. If as many potential training examples of the class as possible are to be identified, the recall should be high. If, on the other hand, the assignment to the class should be correct with a high degree of certainty, the precision should be high. For many classifiers there is the possibility to influence the relative weight of precision and recall.

3.7 Summary

Machine Learning involves fitting a model to data in such a way that it can be used to extract relevant information. In supervised learning, one annotates the data directly with the answer sought, while in unsupervised learning, the model captures relationships within the data. Reinforcement learning deals with temporal sequences (e.g., games), where the final result is available only after a series of time steps and actions.

Learning proceeds according to a general scheme. First, the model performs a prediction with the inputs. Then, the loss function compares the predictions with the outputs in the training data. Next, the optimization modifies the model parameters to reduce the loss. This is almost always done by calculating the gradient that points in the direction of the steepest increase of the loss. The loss is reduced by many small steps in the negative gradient direction until there is no more change. In this process, the individual gradient is usually calculated with only a few training examples, which dramatically reduces the computational effort and avoids local minima.

The quality of the model is determined by computing performance measures such as accuracy or F-value. These measures must always be calculated on a test set, i.e. examples that have not been used for training.

The term Artificial Intelligence has undergone a change of meaning in the last decade. Initially, AI emulated abstract human cognitive abilities, particularly for logical "reasoning". However, progress in these research fields has been limited. In recent years, many areas have been subsumed under AI that were previously considered Machine Learning and data mining, e.g., speech recognition, image classification, or even game strategy optimization. These techniques were not developed to imitate human reasoning, but to solve a practical problem. There is a danger that by calling these techniques "AI," they become too focused on imitating the mental capabilities of humans and disregard other possible solutions (Jordan 2018).

References

Abraham, R., Marsden, J. E., & Ratiu, T. (2012). *Manifolds, tensor analysis, and applications* (vol. 75). Springer.

Efron, B., & Hastie, T. (2016). *Computer age statistical inference* (vol. 5). Cambridge University Press.

Jordan, M. (2018). Artificial intelligence — The revolution hasn't happened yet. Mediu. 19042018. https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7

Kaggle. (2017). Kaggle 2017 survey results. https://www.kaggle.com/amberthomas/kaggle-2017-survey-results

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE, 86*(11), 2278–2324.

Smith, C. (2020). Computers already learn from us. But can they teach themselves? *NewYork Times* 12-04-2020.

Chapter 4
Deep Learning Can Recognize Complex Relationships

Abstract For more complex problems, simple linear models are insufficient. A way out is offered by models with several nonlinear layers (operators), which can represent arbitrary "curved" relationships between inputs and outputs. This chapter describes the properties of such deep neural networks and shows how to find the optimal parameters using the backpropagation method. It then discusses the problem of overfitting and how it can be solved using regularization methods. Finally, an overview of the different types of deep neural networks is given and methods for finding a network structure are outlined.

Logistic regression models work very well in many cases. The Kaggle Data Science Survey, a survey among professional computer scientists, has found that logistic regression is by far the most commonly used analysis method (Kaggle, 2017). Nevertheless, it has been shown that logistic regression cannot find a satisfactory solution in a number of cases.

The logistic regression model can also be considered as a simple neural network. In fact, a simplified variant, the perceptron, was suggested as a first artificial neural network (Rosenblatt, 1958). It had an input vector and a single output variable. The components of the input vector and the output variable were also referred to as "neurons". The connections between the artificial neurons were weighted, meaning that they were not all equally strong. Thus, an input neuron makes an individual contribution to the excitation or inhibition of the output neuron. A simple learning algorithm was also presented that changed the weights each time errors occurred.

4.1 The XOR Problem Involves Interactions Between Features

If in a logistic regression model, an input value, e.g. x_1, increases, the output score always changes in one direction. The XOR problem is a classification problem with a two-dimensional input vector $x = (x_1, x_2)$ where the output probability must

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_4

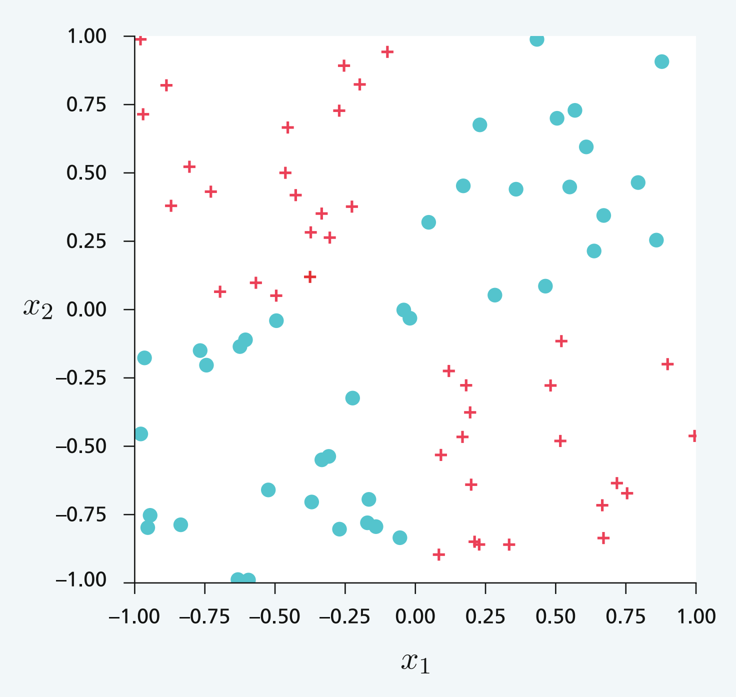


Fig. 4.1 The XOR dataset contains observations on two input variables x_1, x_2, which are assigned to two classes • and +. The location of the class areas replicate the XOR function: it has the value 0 (i.e. +), if exactly one of the variables x_1, x_2 is greater than 0. Otherwise, it has the value 1 (i.e. •)

change in a complex way. Figure 4.1 shows a training set in which 200 training examples are each assigned to class • or +. The class is equal to • exactly when $x_1 * x_2 > 0$. This corresponds to the XOR function in logic.

- If $x_1 < 0$, the probability of class • decreases from 1 to 0 with increasing value of x_2.
- If $x_1 > 0$, the probability of the class • increases from 0 to 1 with increasing value of x_2.

If the classification depends on such a combined effect of the input features, this phenomenon is called a statistical interaction.

Figure 4.2 shows a logistic regression model trained with this data. The training examples are again marked by 0 or 1. The colors represent the probability of the classes computed by the model. The line indicates the probability 0.5. Apparently, the logistic regression model is not able to separate the observed training examples

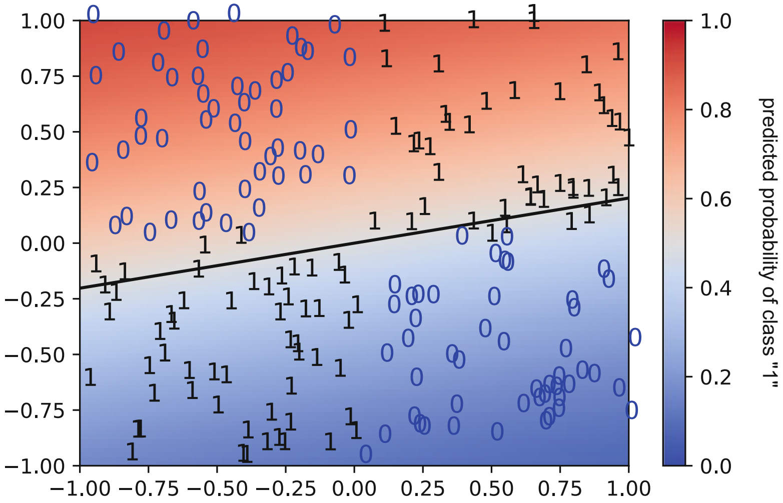


Fig. 4.2 The XOR data cannot be classified with a logistic model. The observed values are printed as 0 and 1. Red background indicates a high predicted probability of class 1 and blue background indicate a high predicted probability of class 0. On the right is a color scale for the background. The line is the contour line of probability 0.5

Marvin Minsky served in the Army from 1944 to 1945 and then studied mathematics at Harvard University. For his dissertation, he built the first neural network (SNARC) in hardware. In 1958, he went to MIT and founded the AI Lab there with John McCarthy. He developed a 7.4 Turing machine that could execute arbitrary programs. Together with Papert, he published the book Perceptrons, which illustrated weaknesses of single-layer perceptrons. Both developed the “Society of Minds Theory”, which aims to explain how intelligence can arise from the interaction of non-intelligent parts. In 1975 followed a theory of “frames”, which was widely adapted. In 1969 he received the Turing Award.

Fig. 4.3 Marvin Minsky, born August 9, 1927 in New York; † January 24, 2016 in Boston, Massachusetts, was one of the fathers of symbolic AI. Image credits in Appendix A.3

of different classes. As the line shows, the logistic regression model can only produce straight separating hyperplanes between classes. This is plausible because the logistic regression model essentially consists of an affine transformation, which always transforms straight lines into straight lines (Fig. 3.15). For classification problems that require curved separating hyperplanes, logistic regression does not provide a solution.

Minsky and Papert (1969) (Fig. 4.3) pointed out this problem and concluded that the neural networks known at that time could not solve relevant problems. This discovery was a shock in the early days of neural networks and brought research in this area to a virtual standstill. It was not until the mid-1980s that it became clear how to deal with this challenge. Figure 4.4 shows the history of machine and deep learning.

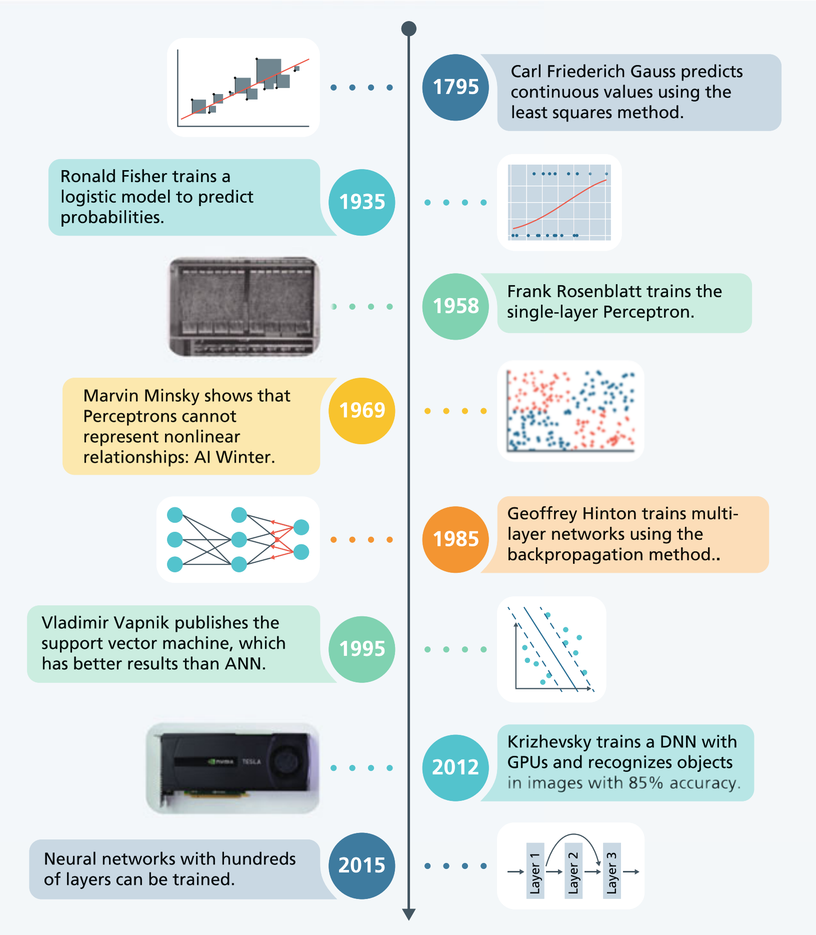

1795
Carl Friederich Gauss predicts continuous values using the least squares method.
Ronald Fisher trains a logistic model to predict probabilities.
1935
1958
Frank Rosenblatt trains the single-layer Perceptron.
Marvin Minsky shows that Perceptrons cannot represent nonlinear relationships: AI Winter.
1969
1985
Geoffrey Hinton trains multi-layer networks using the backpropagation method..
Vladimir Vapnik publishes the support vector machine, which has better results than ANN.
1995
2012
Krizhevsky trains a DNN with GPUs and recognizes objects in images with 85% accuracy.
Neural networks with hundreds of layers can be trained.
2015
Layer 1
Layer 2
Layer 3

Fig. 4.4 History of machine and deep learning. After initial euphoria, interest in neural networks declined sharply in 1969. After an intermittent upswing, the level of attention dropped again. Only with the technical developments of the last years DNN experienced a stormy renaissance. Image credits in Appendix A.3

4.2 Nonlinearities Create Curved Separating Planes

Rumelhart et al. (1985) suggested the use of additional layers within the neural network. Each of these layers had to contain a nonlinear function to represent curved surfaces. In the following table and Fig. 4.5 the most important of these so-called activation functions are shown:

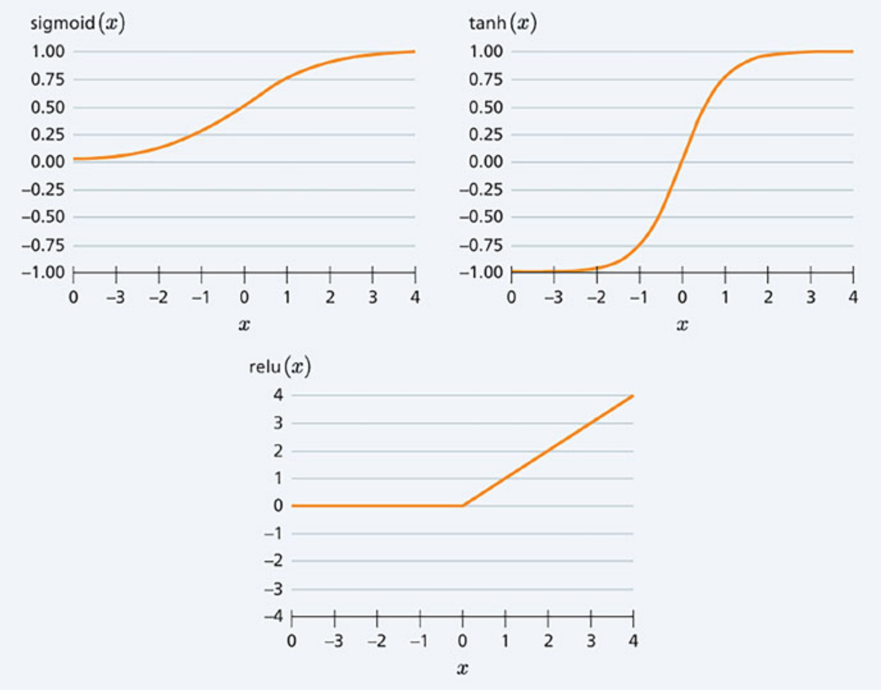


Fig. 4.5 Popular nonlinear activation functions for neural networks: sigmoid function, tangent hyperbolic, and rectified linear unit (ReLU) function

Name	Function $f(x)$	Derivative $\frac{\partial f}{\partial x}(x)$
Sigmoid	$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$	$\text{sigmoid}(x)(1 - \text{sigmoid}(x))$
Hyperbolic tangent	$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$(1 + \tanh(x))(1 - \tanh(x))$
Rectified linear unit	$\text{ReLU}(x) = \max(0, x)$	1 if $x > 0$ and 0 otherwise.

They are applied to a vector or tensor by separately transforming each component. The rectified linear function has a special property, since it has a "corner" where no derivative exists. In practice, it has been found that one can use the gradient descent method even if single corners are contained in the neural network, because one almost never hits this corner.

In Fig. 4.6 the structure of a neural network with two layers is shown. In the first layer, an affine transformation $u_1 = W_1 * x + b_1$ takes place, whose result vector is then subjected to a nonlinear transformation (e.g. $h = \tanh(u_1)$). The result of this transformation is a vector h whose values are not available in the training data. Therefore, it is called a hidden vector. The length of this vector is user-defined. Such a layer is also called a fully connected layer, because between each component of the

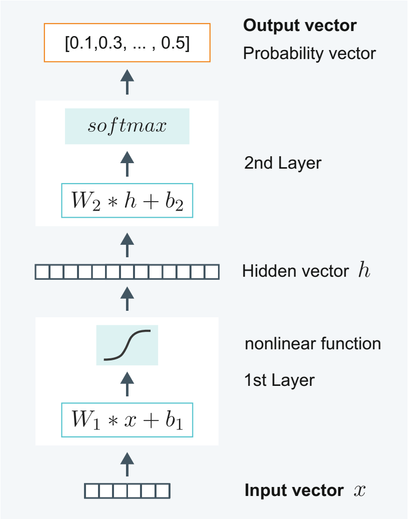


Fig. 4.6 Neural network with two layers. Importantly, the nonlinear function in the first layer improves the representation capability

input vector and each component of the output vector there is a weighted connection with an associated parameter value.

The second layer of the neural network consists of a logistic regression model with another affine transformation $u_2 = W_2 * h + b_2$ and a following softmax transformation $\text{softmax}(u_2)$. This produces the vector of class probabilities. Overall, there are now more unknown parameters $w = (W_1, b_1, W_2, b_2)$, which can be combined into the parameter vector w and modified by the optimization procedure.

Historically, the computation of the gradients for neural networks with multiple layers was a major problem. It was not until the mid-1970s that the previously developed backpropagation algorithm was used for this purpose (Werbos, 1974), which enabled efficient computation of the gradient. Today, there is no need to worry about determining the gradient; this is done automatically by the neural network program system. We applied a neural network with two layers and a hidden vector of length 4 to our XOR data. It has a total of 22 parameters. The result is shown in Fig. 4.7. The colors represent the probability of the classes and the contour lines have equal probability. Obviously, the classification task is solved much better by this model than by the logistic regression model. The accuracy on the test set reaches 98%. The error of 2% can be explained by the random differences between the training and test set.

A neural network with two layers can also be trained on the MNIST data (Sect. 3.2.3). The length of the hidden vector was set to 100. Figure 4.8 shows the accuracy on the training and testing set of MNIST data after each epoch. The accuracy on the training set increases steadily to 0.991, while the accuracy on the test set increases to 0.975, i.e., out of 100 digits, 2.25 are misclassified. The test accuracy is much higher than the corresponding accuracy of 0.924 of the logistic regression model. This means that nonlinear layers lead to better classification in

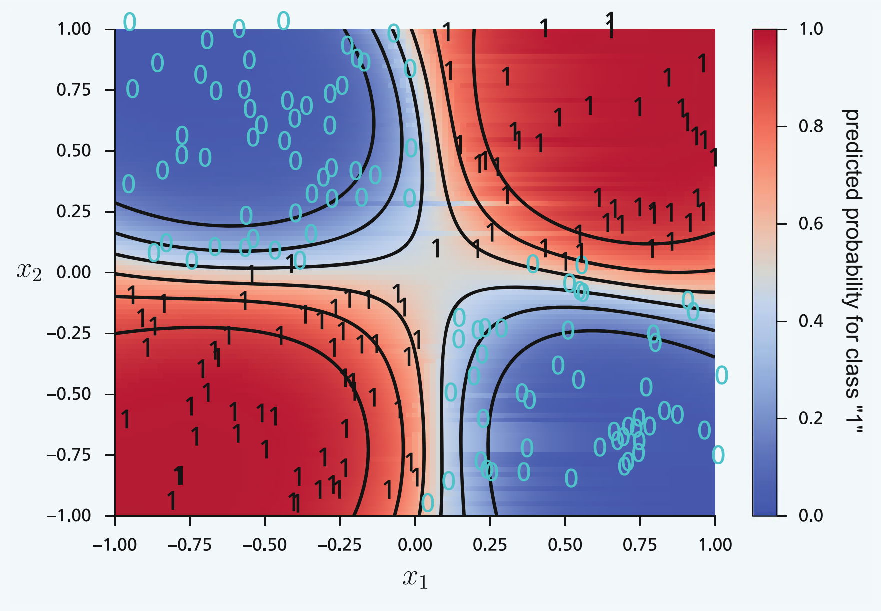


Fig. 4.7 Classification result for a neural network with two layers and a hidden vector of length 4 for the XOR data. The colors show the level of predicted probabilities. Red values show high probability of class 1 and blue values show high probability of class 0. On the right is a color scale for coding the values. The lines are contour lines of constant probability

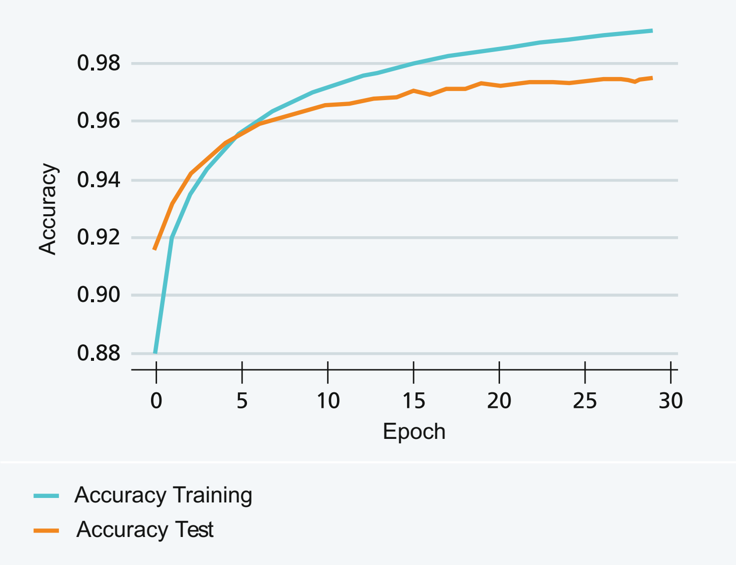


Fig. 4.8 Accuracy of a neural network with two layers on the training and test set of MNIST data

this application. As expected, the accuracy on the test set is significantly lower than on the training set. We will investigate how to further increase the accuracy on the MNIST data in the following sections.

The success of multilayer neural networks triggered a wave of euphoria. Rumelhart et al. (1985) suggested to replicate recognition processes in the brain by using neural networks with several layers. They founded connectionism, which assumes that brain activity can be simulated by networks of simple, connected units. According to this idea, neurons correspond to individual components in the hidden vectors, and synapses are comparable to the connections in the neural network, whose strength is expressed by the weight parameters.

Finally, it could even be formally proven (Hornik et al., 1989) that neural networks with two layers can represent and learn arbitrary functional relationships between inputs and outputs. Unfortunately, in real applications it is often the case that the length of the hidden vector must then become extremely large and the models are no longer trainable in practice. The next section discusses approaches to deal with this problem.

4.3 Deep Neural Networks Are Stacks of Nonlinear Layers

The experiences of the last years have shown that the recognition efficiency of neural networks can be increased by additional layers. Figure 4.9 shows a neural network with three layers. In the application, an input vector is successively transformed in the individual layers and finally an output is generated. Therefore, such networks are also called multilayer feedforward networks. The use of additional layers increases the flexibility and mapping capacity of the networks, but also the number of parameters, the amount of necessary data and the required computing effort.

Networks with many layers are referred to by the collective term deep neural network (DNN). More general network types are presented in detail in the following chapters.

Meanwhile, powerful DNNs have several hundred layers and many millions of parameter values. All these layers are hidden, i.e. the associated vectors are not observable in the data. In this context, the layers often no longer form a simple series, but there are also parallel connections and cross-connections. The details are discussed in the following chapters.

Deep neural networks are not programmed, they are **trained**. That means, nobody prescribes by hand what exactly a component of the hidden vectors has to represent. During the optimization the individual components are automatically assigned a functionality—based on the randomly chosen initial parameters—so that finally the accuracy of the overall system is as high as possible. With new random initial values, the functionality will evolve differently during optimization.

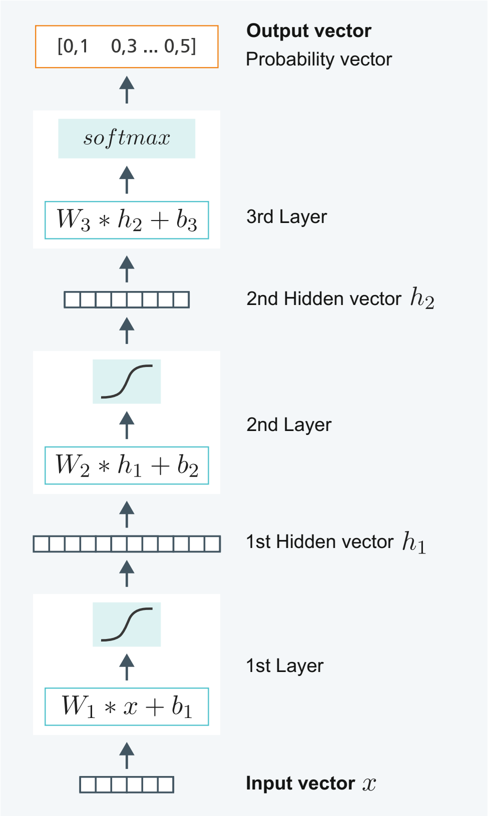


Fig. 4.9 Multilayer feedforward network with three layers. The input vector x is transformed into the hidden vectors h_1 and h_2 and finally into a probability vector that estimates the probabilities of each class

4.3.1 *Vectors and Tensors Represent the Transformed Contents*

The deep neural network in Fig. 4.9 transforms the input vector x into the hidden vectors h_1 and h_2. Therefore, h_1 and h_2 can be viewed as **representations** of the input vector (Fig. 4.10). The representations are chosen by the training algorithm in such a way that the final task, the classification of the input, can succeed as well as possible. Therefore, the representations will contain the essential information about the input that will enable classification.

The representations are thus a kind of language that the network uses to describe the data. As we have seen, neural networks can handle very many different inputs: Images, sounds, speech, videos, measurements, etc. All of these inputs are represented by vectors, matrices, or tensors. Within the DNN, further representations are successively generated during training, which finally allow to produce the desired output.

The automatic generation of improved representations of the inputs can be considered a characteristic feature of deep neural networks. In contrast to classical Machine Learning approaches, the user does not have to determine suitable features

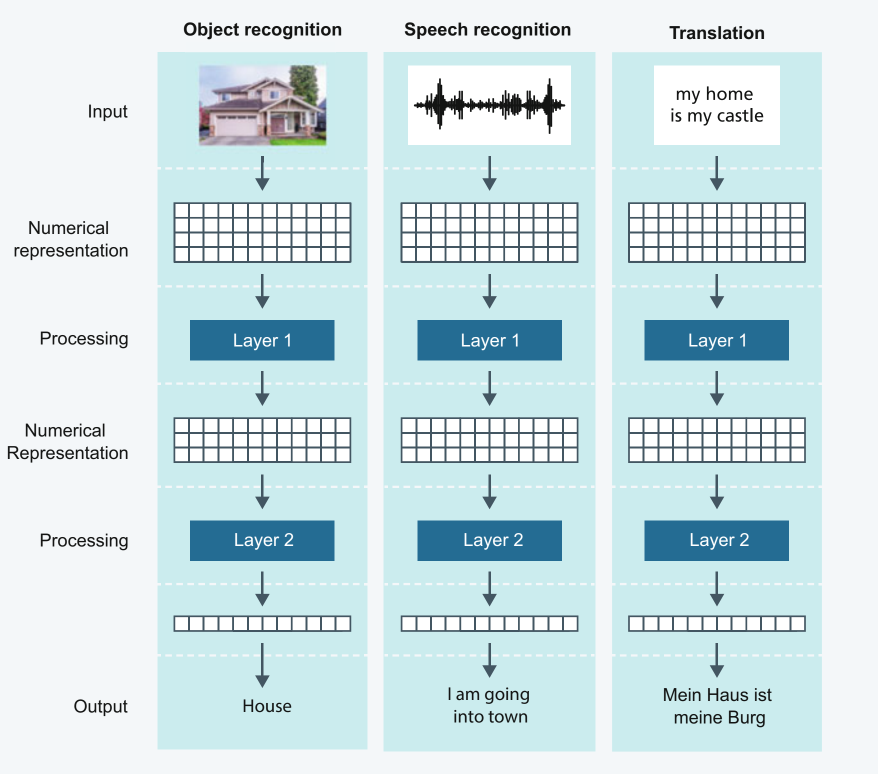


Fig. 4.10 The input of a DNN is translated into a numerical representation (vector, matrix, tensor). In each layer of the network, a new numerical representation of the input is generated so that eventually the goal of the network (e.g., classification of objects) can succeed as well as possible. In real-world applications, the structure of networks varies widely for different tasks. Image credits in Appendix A.3

and representations of the data, but deep learning automatically generates the best representations, which then allow the desired prediction or classification in an optimal way. The theoretical analysis of DNN has shown that the representational power of deep neural networks grows exponentially with depth: thus, each additional layer multiplies the number of relationships that can be represented (Eldan & Shamir, 2016). Furthermore, one can theoretically deduce that deeper neural networks are much better at identifying spatially rotated, shifted, or transformed objects than shallower networks (Bruna & Mallat, 2013).

It should be noted, however, that the DNN is not the only type of model that generates nonlinear surfaces of separation. Alternatively, almost all advanced Machine Learning approaches, such as nonlinear support vector machines, decision trees, random forests, k-nearest neighbors, etc., are capable of doing so. However, these approaches are usually not built from different layers, so hierarchies of more complex hidden features in each case cannot develop. Therefore, the accuracy of classical Machine Learning approaches is usually not as good as the accuracy of DNN for complex problems.

4.4 Training of DNN with the Backpropagation Approach

A DNN consists of a series of operators. In Fig. 4.9, each operator has a vector as input and produces an output vector, which is the input of the next operator. Instead of vectors, other models use scalars, matrices, or arbitrary tensors as input and output.

One difficulty with a series of operators is computing the gradient. The starting point is again the training set TrainSet $= \{(x^{(1}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of input-output pairs. As in Sect. 3.5.1, the parameters should be chosen to maximize the probability of the entire training set. It was not until the mid-80s that the previously developed backpropagation algorithm was used for this purpose (Werbos, 1974), which allowed efficient computation of the gradient. This method is based on the chain rule (Goodfellow et al., 2016, p. 199), which is suitable for calculating the gradient of nested functions.

Today's toolkits calculate the derivatives automatically once the model and loss function are specified. Nevertheless, it is important to understand which calculations are necessary to determine the derivative and what the chronological order of the calculations looks like.

As an example, we consider an extremely simple model with an input x of length 1, a hidden layer with vector $h_1 = f_1(w_1 x)$ of length 1, an output $\hat{y} = f_2(w_2 h_1)$ of length 1, and a loss function $loss = L(\hat{y}, y)$. The goal is to minimize the loss value $loss$ by changing the parameters w_1 and w_2. The model forms a nested function. The calculation of the derivatives for the parameters w_1 and w_2 according to the chain rule is shown in Fig. 4.11. The calculation has the following important properties:

- The derivatives of the individual functions $f_1(x)$, $f_2(h)$, and $L(\hat{y}, y)$ are calculated separately (highlighted in blue), e.g., using the derivatives from Sect. 4.2, and then linked together by simple operations (multiplication).
- The derivatives are themselves functions (highlighted in blue), e.g. $\frac{\partial f_2(u_2)}{\partial u_2}(u_2)$ which are evaluated for certain intermediate results (highlighted in yellow), e.g. $u_2 = w_2 f_1(w_1 x)$. These intermediate outcomes are the result of the forward calculation for the previous, inner functions.
- Common expressions appear in the derivatives of the different parameters w_1 and w_2, e.g. $\frac{\partial L(\hat{y}, y)}{\partial \hat{y}}(\hat{y}) * \frac{\partial f_2(u_2)}{\partial u_2}(u_2)$, which can be reused.

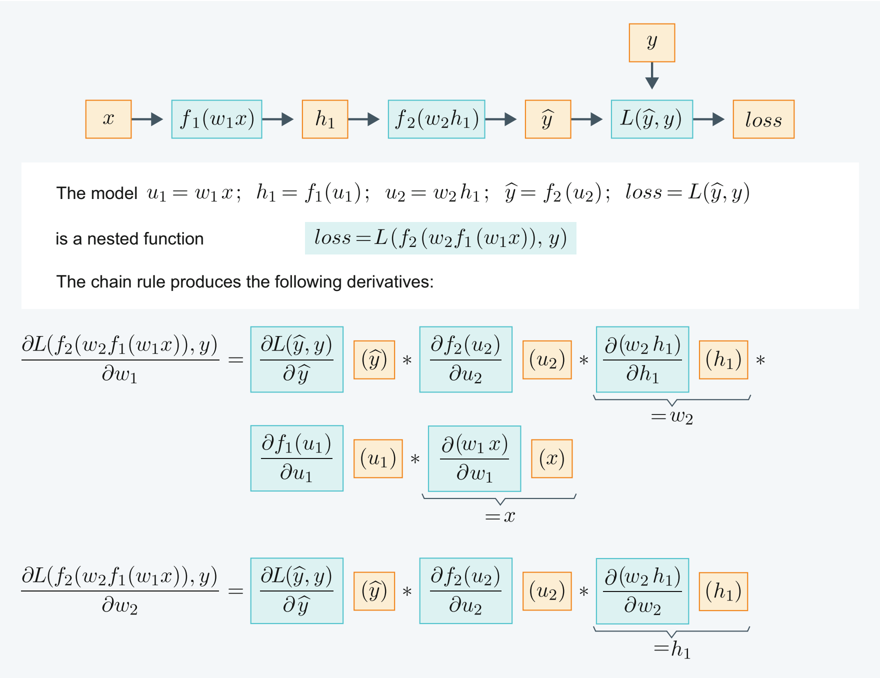

y
x
$f_1(w_1x)$
h_1
$f_2(w_2h_1)$
$\widehat{y}$
$L(\widehat{y},y)$
$loss$
The model $u_1=w_1x;\ h_1=f_1(u_1);\ u_2=w_2h_1;\ \widehat{y}=f_2(u_2);\ loss=L(\widehat{y},y)$
is a nested function $loss=L(f_2(w_2f_1(w_1x)),y)$
The chain rule produces the following derivatives:
$\frac{\partial L(f_2(w_2f_1(w_1x)),y)}{\partial w_1}=\frac{\partial L(\widehat{y},y)}{\partial\widehat{y}}(\widehat{y})*\frac{\partial f_2(u_2)}{\partial u_2}(u_2)*\underbrace{\frac{\partial(w_2h_1)}{\partial h_1}(h_1)}_{=w_2}*\frac{\partial f_1(u_1)}{\partial u_1}(u_1)*\underbrace{\frac{\partial(w_1x)}{\partial w_1}(x)}_{=x}$
$\frac{\partial L(f_2(w_2f_1(w_1x)),y)}{\partial w_2}=\frac{\partial L(\widehat{y},y)}{\partial\widehat{y}}(\widehat{y})*\frac{\partial f_2(u_2)}{\partial u_2}(u_2)*\underbrace{\frac{\partial(w_2h_1)}{\partial w_2}(h_1)}_{=h_1}$

Fig. 4.11 For a highly simplified model where input and hidden layers have length 1, the derivative is calculated. The chain rule shows that the derivative for a training example (x, y) is a product of the partial derivatives (highlighted in blue) of the individual functions. In each case, they are calculated for the predicted values of a forward prediction (highlighted in yellow). For the activation functions $f_1(u_1)$ and $f_2(u_2)$ the known derivatives can be used

These properties can be applied almost unchanged to functions (e.g. DNN), which have multidimensional tensors as input/output. Therefore, the backpropagation algorithm for computing derivatives for a training example (x, y) proceeds in the following steps:

1. **forward-propagation:** From the input x, a prediction is computed sequentially for each layer of the DNN and all intermediate results, e.g. $x, u_1, h_1, u_2, \hat{y}$ are stored.
2. **backpropagation:** For the layers of the DNN:
   (a) The derivative of the last layer $\frac{\partial L(\hat{y},y)}{\partial \hat{y}}(\hat{y})$, the penultimate layer $\frac{\partial f_2(u_2)}{\partial u_2}(u_2)$, $\ldots$, the first layer $\frac{\partial f_1(u_1)}{\partial u_1}(u_1)$ and $\frac{\partial w_1 x}{\partial w_1}(x)$, are evaluated for the previously calculated intermediate values.
   (b) The results are linked by simple operations (multiplication, addition).

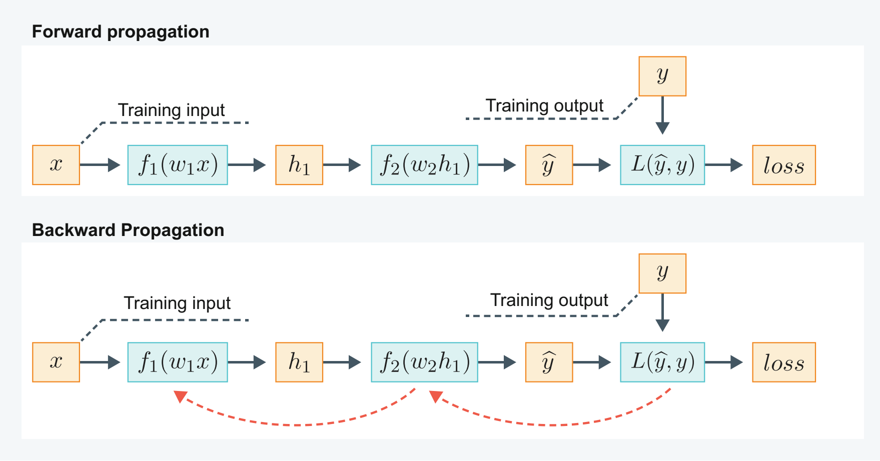


Fig. 4.12 Information flow in a DNN during the backpropagation process for a training example (x, y). Top: In forward propagation (black arrows), the input x is propagated through the DNN and an output $\hat{y}$ is predicted, and thus the associated loss is computed, with all intermediate results stored. Bottom: In backward propagation (dashed red arrows), the partial derivatives for the last, second to last, etc. layer are calculated employing the values of the intermediate results and the gradients of the unknown parameters are determined

After the DNN has been specified, the operations necessary to calculate the gradient can be determined automatically. This is realized in modern toolkits for programming DNN.

Figure 4.12 illustrates the information flow during training for a training example (x, y). In forward propagation (top), a prediction $\hat{y}$ is computed from the input x and the associated loss is determined. All intermediate results are stored. In backward propagation, the partial derivatives for the last, second to last, etc. layer are evaluated for the stored intermediate results and the respective partial derivatives of the unknown parameters are calculated.

The derivatives for different training examples are added up, since the loss function is a sum (Sect. 3.5.2) and the derivative of a sum is equal to the sum of the derivatives. In the original gradient descent procedure, the gradients of all training examples were added up and then the parameter vector was changed (Sect. 3.6.4). Today, the gradients for individual minibatches—randomly selected small subsets of the training examples—are usually summarized, and then the parameters are adjusted (Sect. 3.6.6). This can greatly reduce the computational cost. Geoffrey Hinton was a major promoter of the use of the backpropagation algorithm (Fig. 4.13).

The computational effort to determine the gradient for each training example is at least twice the effort required for a pure forward prediction. If one has a large number of training examples, this leads to a large computational cost. When you have calculated the gradient once for each element of the training set, this is called

Geoffrey Hinton studied psychology at Cambridge. His goal: "I wanted to understand how the mind worked". He realized "that psychologists didn't have a clue." In the 1980s, he promoted the use of neural networks along with the backpropagation algorithm to calculate gradients. "The brain sure as hell doesn't work by somebody programming in rules." In the years that followed, he unwaveringly continued to develop neural networks at the University of Toronto, and in 2012 he and his graduate students developed AlexNet, one of the first DNNs to greatly reduce image classification error rates. In 2018, he received the Turing Award, the highest honor for computer scientists, together with Yoshua Bengio and Yann LeCun.

Fig. 4.13 Geoffrey Hinton, born Dec. 6, 1947, is one of the fathers of deep neural networks. Image credits in Appendix A.3

an epoch. But often gradients have to be evaluated in hundreds of epochs, which results in an extreme computational overhead.

4.5 Toolkits Facilitate DNN Specification and Training

4.5.1 Parallel Computations Accelerate DNN Training

Training DNN with many millions of parameters often took several days and weeks. For this reason, possibilities to speed up the calculations by using parallel processors were investigated. There are a number of alternatives for this:

- Modern computers have a number (usually 4–32) of computational cores that can perform separate computations on the shared memory.
- A graphics processing unit (GPU) is a computer specialized in the generation of graphics with currently up to 4600 computing cores (Fig. 4.14). These GPUs were developed for video games, but can also be used for general calculations via a special programming language CUDA. GPUs are usually accessed via a normal computer.
- A cluster is a group of individual computers, possibly with GPUs, which are connected via a fast network. Compute clusters are also available from external providers via the Internet, i.e. in the "cloud".

Training is typically parallelized by dividing the available training data of a minibatch among the compute cores. Each compute core separately computes the derivatives for its data. The results are then aggregated to produce the derivative for the minibatch. Deep neural network toolkits have been developed that automatically distribute computations to parallel computers.

Fig. 4.14 The Nvidia Tesla 2075 graphics processing unit (GPU) with 448 cores has a computing power of up to 1.03 teraflops, or 1 million times 1 million flops (floating point operations per second). Image credits in Appendix A.3

4.5.2 *Toolkits Simplify Work with DNN*

When coding a DNN with a common programming language, there are many sources of errors. Special toolkits allow to easily specify, train and apply a DNN (Fig. 4.15). There are a number of these toolkits; an overview can be found in (DL-Software, 2019; Nguyen et al., 2019), of which CNTK, PyTorch (Pytorch, 2019), and TensorFlow (Abadi et al., 2016) are the most popular. Currently, TensorFlow is the most widely used toolkit. It was developed by the Google Brain team

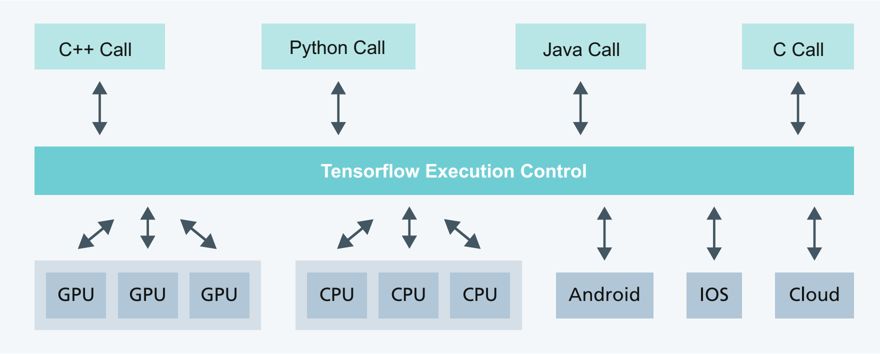


Fig. 4.15 Structure of the TensorFlow toolkit. The computations can be executed on CPUs, GPUs, cell phones and in the cloud. The call is possible from various programming languages

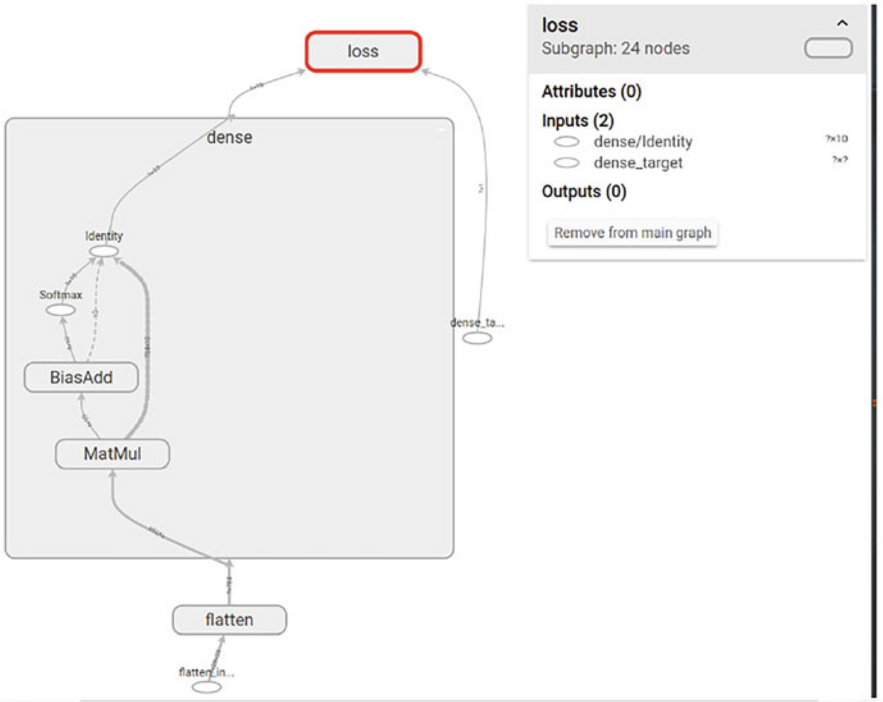


Fig. 4.16 TensorFlow generated representation of the data flow in a DNN

and released under the Apache 2.0 open source license. It provides the following functionality:

- Composition of the DNN from different operators, which can have arbitrary vectors, matrices and tensors as inputs and outputs. The operators do not all have to be executed one after the other, but parallel information flows are also allowed. Therefore, the operators form a directed graph (Fig. 4.16).
- The derivatives are calculated symbolically and inserted into the graph as additional operators.
- A number of optimization methods are available, in particular stochastic gradient descent (SGD) and its variants.
- The computations are automatically offloaded to the available parallel hardware and the transfer of data between the components is ensured. Figure 4.15 shows the architecture of the TensorFlow system.

The toolkit is available for different programming languages, preferably Python, but also Java, JavaScript, C++, etc. Keras (Chollet, 2019) is a simplified, user-friendly operating language for TensorFlow and other program systems. The goal is to specify the most common network types in a flexible and concise way. Figure 4.17 shows the Keras program code for a classification model with three layers.

```
x = Input((X_train.shape[1],))
hid1 = Dense(units = 128, kernel_initializer='he_normal', activation='tanh')(x)
hid2 = Dense(units = 64, kernel_initializer='he_normal', activation='tanh')(hid1)
y = Dense(units = 10, kernel_initializer='he_normal', activation='softmax')(hid2)
model = Model(x,y)
model.compile(loss='categorical_crossentropy', optimizer=SGD, metrics='accuracy')
```

Fig. 4.17 Keras program code for a neural classification network with three layers. The first line defines the dimension of the input. Lines 2–4 specify the properties of the layers. The last line specifies the loss function, the optimizer, and the evaluation metric

4.6 How to Improve the Network?

A DNN has a large number of hyperparameters (control variables), e.g., the number of layers, their interconnection, or the learning rate of the optimization procedure. When designing a model, it is a great challenge to determine the number of layers, the length of the hidden vectors, and the linkage pattern of the layers. Unfortunately, such hyperparameters cannot be determined by the normal optimization procedures.

Therefore, it is obvious to compare several DNN with different hyperparameters based on their accuracy on the test set. Then, the best DNN can be selected and used for the actual prediction task. However, there is a large problem with such an approach: the accuracy of the best DNN on the test set is systematically too high if one compares several variants of the DNN. Thus, one can no longer rely on the fact that applying the DNN to new data will result in the same average accuracy as on the test data. The bias is due to the fact that precisely the model variant with the best accuracy on the test data was selected.

4.6.1 Iterative Model Construction Using the Validation Set

Since the final test of the model with the test set has to determine a reliable accuracy for the future application, the test set cannot be used for structure selection and a different procedure is necessary. One has to select an additional, randomly determined "validation set" which, like the test set, must not be used during training. Figure 4.18 illustrates this random partitioning of the annotated data. On the validation set, one can then check which combination of hyperparameter values

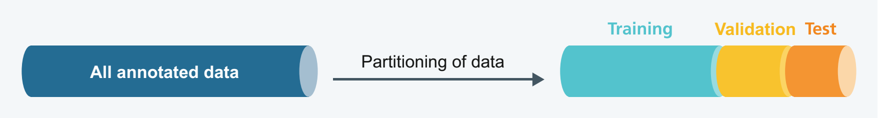


Fig. 4.18 Random partition of data into a training set, a validation set, and a test set

is the best. The accuracy of the selected model variant during application can then be estimated once (and not more often) on the test set at the end.

It should be noted, of course, that the estimation on the reduced training set cannot be as accurate as on the larger training set. There are theoretical studies which state that a quadrupling of the number of training data leads to a halving of the range of variation (standard deviation) of the model predictions. Therefore, it is always better to perform estimation on a larger set of training samples.

4.6.2 Underfitting and Overfitting Lead to Higher Errors

When checking the accuracy of a neural network on the validation set, there can be two types of problems:

In case of underfitting, the accuracy remains very low. This can happen, for example, because the model is not able to adapt to the training data. This is illustrated in Fig. 4.19 on the left. For example, this situation was the case with the logistic model for digit classification (MNIST data). Then one can increase the complexity of the model, for instance, by using additional layers or otherwise enlarging the number of model parameters. However, underfitting can also be caused by the fact that the model was not trained long enough or that the step size during optimization is too large.

The opposite of underfitting is overfitting. In this case, the accuracy on the training data becomes much higher than on the validation or test data. As shown in Fig. 4.19 on the right, the model then fits individual observations in the training set and learns the systematic relationships only insufficiently. It may even happen that the accuracy on the validation data drops again after a certain epoch, while the accuracy on the training data continues to increase. This scenario is shown in Fig. 4.20. The model then has too much complexity and too many parameters.

A simple technique to deal with overfitting is early stopping. This involves monitoring the accuracy on the validation data during an optimization run. If this accuracy has reached a maximum value and starts to decrease, the optimization is stopped and the model with the highest validation accuracy is used. This method

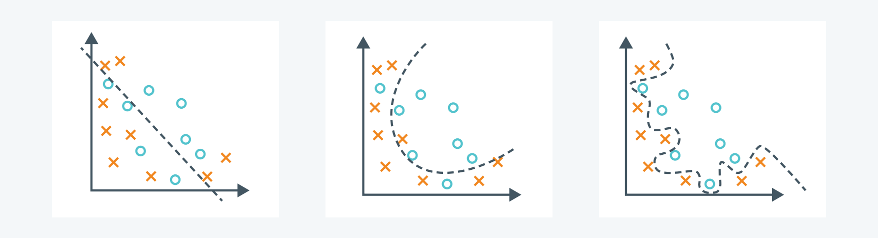


Fig. 4.19 The situation of underfitting (left), ideal fitting (center), and overfitting (right) in a classification problem

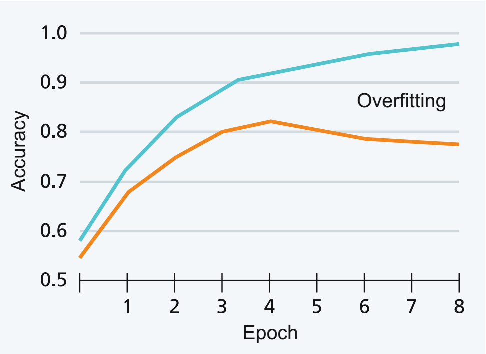


Fig. 4.20 Development of accuracy on the training data (blue) and the validation data (orange) when overfitting. A high difference between both accuracies is observed. In the further course of the training, the accuracy on the validation data can even drop

sometimes provides a usable model and reduces computation time. However, it is quite possible that there is still a better model with a different architecture for the problem.

The ideal behavior of the model lies between underfitting and overfitting. The model must be complex enough to capture the relevant relationships in the data, but should not be too complex and learn random variations of the training samples. The benchmark is always the generalization error on new data, i.e., the accuracy on the validation and test data.

4.6.3 An Example of Overfitting

To demonstrate the overfitting problem, we study an image classification task that is more difficult than MNIST digit classification. This notMNIST dataset consists of the images of letters A to J in a set of different character sets. The image size is 28 × 28 pixels in 10 classes, as in MNIST (Fig. 4.21).

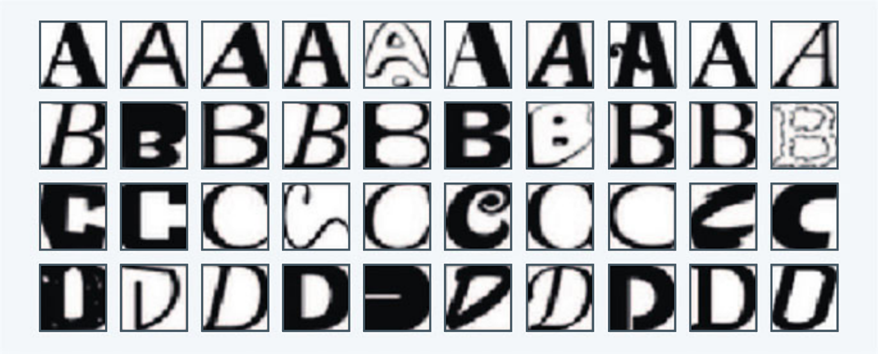


Fig. 4.21 Training examples of letters A-D from the notMNIST dataset. Image credits in Appendix A.3

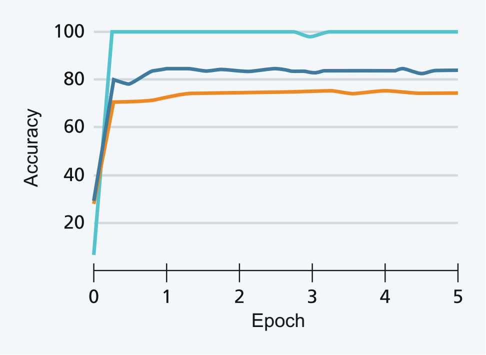


Fig. 4.22 The accuracy of a neural network with two layers on notMNIST. For training on 300 examples: Accuracy on training data (light blue), accuracy on validation data (orange). Training on all data (dark blue) results in higher accuracy

Training a 2-layer DNN with a hidden vector of length 1024 on notMNIST training data with ten thousand training examples, we obtain 84.1% accuracy on the validation set (dark blue in Fig. 4.22). If one trains the same network with only 300 training examples, the situation is completely different. The accuracy on the small training set goes towards 100% (light blue) and the accuracy on the validation data (orange) stagnates at 75.1% for this model. This is a clear case of overfitting. The network is able to accurately predict all examples in the small training set. Thus, it adapts to random variations and is no longer capable of capturing the characteristic properties of the letters. Consequently, the accuracy on the validation set is lower.

This shows that a lack of data can also lead to overfitting. Thus, if more data are available, they should be used. If N is the number of training examples, Shalev-Shwartz and Ben-David (2014) show that the prediction error is bounded by $\sqrt{(\text{number of parameters}) * \log(N)/N}$. This means that, approximately, one must increase the number of training examples by the same amount as the number of model parameters so that the prediction error does not increase.

4.6.4 Regularization Procedures Reduce Overfitting Errors

There are now a number of special procedures which can reduce or avoid overfitting. This process is called regularization. Most of these approaches constrain the complexity of the model:

- Penalizing high parameter values results in less variation (fluctuation) of the model function.
- Early termination of optimization avoids the training phase where the error on the validation data grows again.
- Dropout forces the neural network not to rely on single parameters.

- Batch normalization limits the range of variation of values in the hidden vectors.
- Reducing the number of model parameters by using fewer layers or shorter hidden vectors also reduces variation in outputs.

These regularization procedures are illustrated in what follows.

4.6.5 *Penalizing Large Parameter Values Reduces Abrupt Output Changes*

When penalizing large parameter values, the goal is to avoid high positive or negative values of the model parameters. This is because these large model parameter values lead to strong oscillations in the values of the output vectors, which can be observed during overfitting. This penalization is achieved by adding another term to the model's loss function that produces an additional loss contribution at high parameter values. For example, the log-likelihood loss function (Sect. 3.5.2) is modified as follows.

$$L(w) = -\log p(\text{TrainSet}|w) + \alpha * (w_1^2 + \cdots + w_n^2)$$

Thus, the sum of the square values of all n parameters is to be reduced, where $\alpha > 0$ is a weight factor. The optimization procedure will then tend to avoid the high parameter values and thus reduce abrupt oscillations. The procedure is also called weight decay or L2 regularization and is shown graphically in two dimensions in Fig. 4.23. This classical approach to regularization has good theoretical justification because the prediction error is bounded by the sum of squared weights (Neyshabur et al., 2015), and leads to success in many cases.

There is also a variant in which not the sum of the squares of the parameters is minimized, but the sum of the absolute values of the parameters. This type of regularization is called L1 regularization. It has as a side effect that many parameters

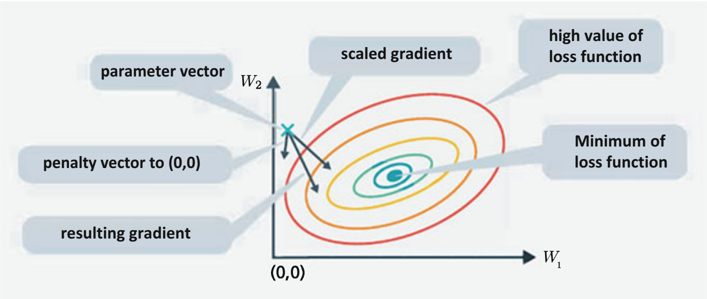


Fig. 4.23 Regularization of parameters by a penalty term added to the gradient during optimization. This "pulls" parameters with high positive or negative values towards (0, 0)

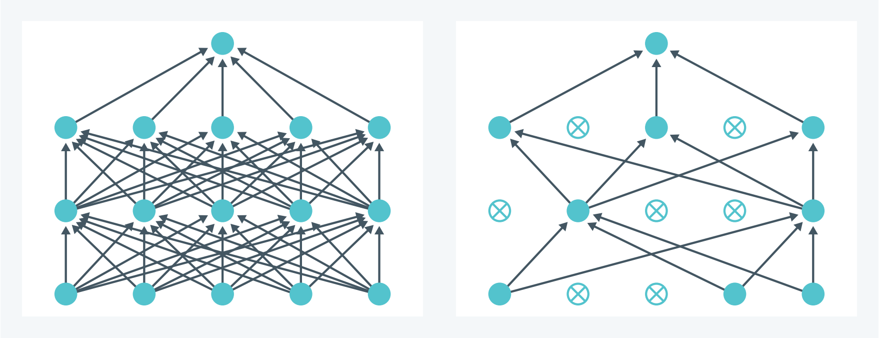


Fig. 4.24 Dropout randomly sets individual parameters of the full model (left) to zero. For a gradient step the model on the right is optimized. During the next gradient calculation, other randomly determined parameters are set to zero

are set to 0.0. However, single high parameter values can also occur, so that the regularization effect is usually not as good as with the L2-regularization.

4.6.6 Dropout Disables Parts of the Network

Dropout involves setting randomly selected parameters of the model to 0.0 during a training step. This is shown in Fig. 4.24. During the next minibatch for gradient calculation, other parameters are randomly selected and the associated connection strengths are set to 0.0. Thus, the neural network cannot "rely" on single parameters, but must use multiple connections to represent a context to ensure accuracy. This results in a strong regularization. In general, 50% of the parameters are set to 0.0. When a model trained with dropout is applied later, no connections are set to 0.0.

For example, we can use the dropout regularization to train the notMNIST model described previously on a subset of 300 training examples. The results are shown in Fig. 4.25. The phenomenon of overfitting disappears and training and test accuracy no longer diverge. The validation accuracy for the partial data model is much higher with dropout (83.8%) than without dropout (75.1%). Surprisingly, the validation accuracy is almost equal to the training accuracy (84.1%) of the model with the full 10,000 training examples without regularization. This shows that dropout regularization is an effective tool in reducing overfitting. Dropout is used for a large number of deep neural networks.

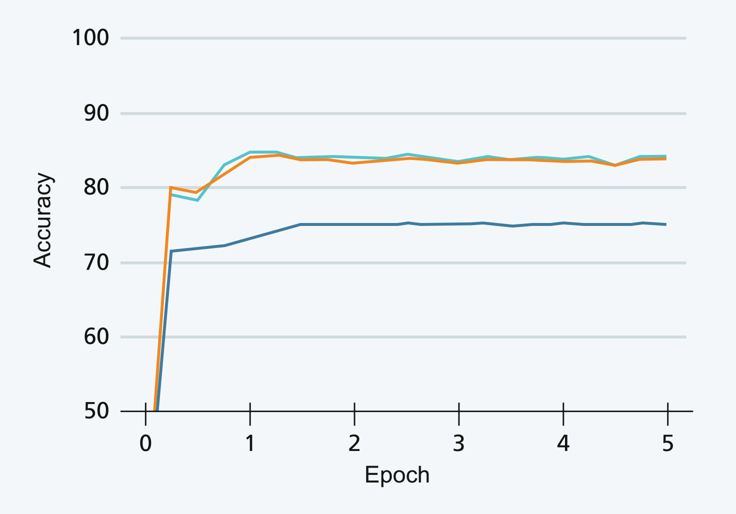


Fig. 4.25 Accuracy of a 2-layer neural network on the notMNIST validation data. Training on 300 examples with dropout regularization produces an accuracy (light blue) that is nearly the same as training on all data (orange) and much higher than without regularization (dark blue)

4.6.7 Batch Normalization Avoids Extreme Values of Hidden Vectors

Batch normalization aims at avoiding extreme values of the hidden vectors. An important reason for this are the nonlinear functions found in each layer of the neural network. If an input component of the hidden vector has a high value—e.g., 100.0—the nonlinear activation functions (e.g. $\tanh(x)$) are very flat for this value. Accordingly, the derivatives are close to 0.0 and it is very difficult to escape from this range of values by the gradient method.

Batch normalization is built into the network as a special layer and ensures that the components of the input vectors for the next layer for a minibatch have values around 0.0. This is done by normalizing the mean of each hidden value in a minibatch to 0.0 and its variance to 1.0. In Fig. 4.26, this is shown for two dimensions; however, the procedure is applicable to any number of dimensions. On the one hand, it has the effect that the derivatives do not become too large or too small. On the other hand, it has the effect that the network avoids extreme value fluctuations in the outputs. Thus, batch normalization can be used as a regularization method. Furthermore, the network is less dependent on a suitable initialization of the parameters and the convergence of the optimization is improved. The method has proven to be particularly useful for DNNs with very many layers.

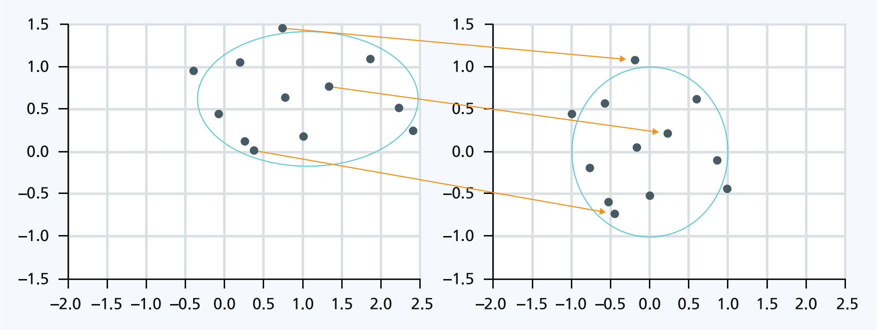


Fig. 4.26 The process of batch normalization of a hidden vector of length 2. For the training examples in a minibatch, the mean and standard deviation are calculated for each component of the hidden vector (on the left). Then, each component is transformed to have a mean value of 0.0 and a standard deviation of 1.0 (right), thus avoiding extreme values

4.6.8 Mathematical Proof: Stochastic Gradient Descent Finds Well Generalizing DNN

We have so far composed a DNN from layers of linear transformations and nonlinear activation functions according to heuristic considerations. We then used a simple optimization procedure, stochastic gradient descent, to optimize the parameters so that the loss function becomes minimal. At first, no one guarantees us that this procedure is able to reconstruct the true relationship between inputs and outputs. However, recent theoretical research could prove that DNN can indeed find the solution we are looking for. Hardt et al. (2016) and Bassily et al. (2018) have theoretically proven that stochastic gradient descent not only leads to an improvement in convergence properties, but also regularizes the DNN. Brutzkus and Globerson (2019) were able to formally derive for a two-layer DNN (under some assumptions) that training with the stochastic gradient descent achieves the global minimum, and that the DNN generalizes well. Thus, in retrospect, the suitability of trained DNN for approximating an arbitrary relationship between input and output data could be theoretically and formally demonstrated.

4.7 Different Applications Require Different Networks Structures

So far, we have considered the situation where a DNN consists of a number of layers and produces an output, e.g., a classification. However, for important applications there exist different types of networks, which are constructed in such a way to

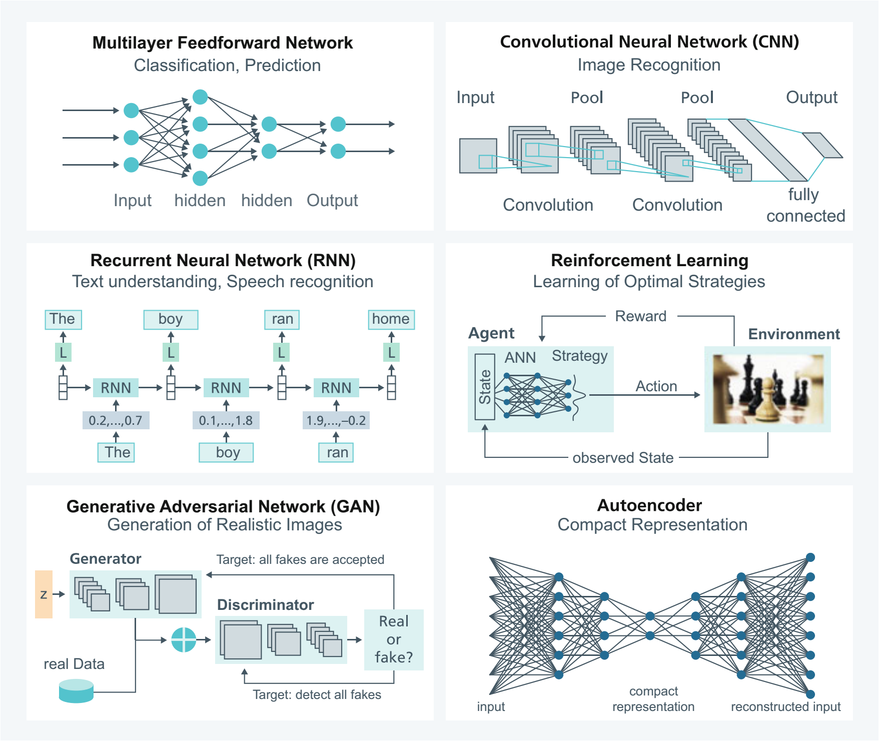


Fig. 4.27 Compilation of six different network architectures. The name of the architecture is printed above the graphic, followed by the main application area. Image credits in Appendix A.3

support the task through architectural features. The most important of these network types have been compiled in Fig. 4.27.

4.7.1 *Multilayer Feedforward Network*

This type of DNN receives a tensor as input and computes an output from it. If the network has to assign the input to classes, the output is usually a probability vector with the probabilities of each class. This network was described in Sect. 4.3.

However, the goal can also be to predict one or more continuous variables. This is called a regression problem and the resulting model is called a regression model. In this case it is necessary to evaluate the difference between the forecast and the observed quantity in the training data by a different loss function. Usually, one chooses the mean squared distance between the prediction and the observed values as the loss function (Fig. 3.19). With this change of the loss function, one can then train the model for the prediction of continuous quantities by optimization.

4.7.2 Convolutional Neural Network (CNN)

This type of network was designed specifically for the image recognition. It has been found that image recognition works well when it starts with the identification of small-scale features, such as corners, edges, and color patches. It is very important that these features are independent of the position on the image, i.e. they can appear anywhere. The recognition is done by a so-called convolution layer with a very small square input area (receptive field) of only a few pixels. This receptive field is "shifted" over the pixel image and can identify the same feature at different positions, e.g. an edge with a specific orientation.

Typically, a large number of alternative features are recognized, and the network autonomously constructs these features so that they are particularly meaningful, when taken together. In the higher layers, these features are combined (pooling) and more abstract features are detected by further convolutional layers. The top layer typically forms a logistic regression model for the classification of the image contents. These networks are described in Chap. 5.

4.7.3 Recurrent Neural Network (RNN)

An important type of input data are sequences, e.g. words of a written text or sound portions of spoken language. For simplicity, we will refer to the components of the sequence as words. An important property is that the meaning of the words in the sequence is often independent of their position, as long as the context is preserved. This means, for example, that the same statement can appear at the beginning or at the end of a longer text. Therefore, recurrent neural networks have been developed, which successively consider the words of a sequence. The information about the already processed words of the sequence is stored in a long hidden vector. The encoding of the content is automatically determined during training. RNN can be used to translate texts from one language to another, to generate new texts, or to transcribe a sound signal of a language into a text. Networks of this type are described in Chaps. 6 and 7.

4.7.4 Reinforcement Learning Network

An important type of problem arises in situations where the final outcome (e.g., gain or loss) is known only after a number of steps. In each step, the situation is described by a state and the agent has to select one of several possible actions. This is the case, for example, for game situations and robot control. Therefore, the agent needs to determine a policy that selects a good action for arbitrary given states. Reinforcement learning provides different approaches to compute such a policy by “replaying” possible move sequences very frequently and using the results as training data. Deep neural networks are used in this process to represent and optimize such strategies. These kinds of deep neural networks are described in Chap. 8.

4.7.5 Generative Adversarial Network (GAN)

Conventional DNNs are tasked with reconstructing the relationship between an input and an output. However, sometimes one wants to generate new data (hereafter called images) that are similar to the images in the training set. For this task generative adversarial networks have been developed. They consist of two subnetworks: A generator network generates new images, while the discriminator network tries to distinguish real images from the artificially generated ones. Both networks are trained simultaneously. In this process, the generator network tries to generate images that are accepted as real by the discriminator. The discriminator network tries to identify all generated images. After training, the artificially generated images are indistinguished from the real ones. This type of DNN is described in Sect. 9.1.

4.7.6 Autoencoder Networks Produce a Compressed Representation

Autoencoder networks are a variant of multilayer feedforward nets. Here, the input is also used as the output, so the network must reproduce the input, i.e., it does not require annotations. Importantly, an intermediate layer is significantly smaller than the input. It is called the information bottleneck. The autoencoder is therefore forced to compress the information of the input into this small hidden vector. Thus, it produces the most efficient encoding of the input vector. This DNN can be used to generate efficient representations of content or to generate new data (Tschannen et al., 2018).

4.7.7 Architectures for Specific Media and Application Areas

The architectures of DNN are strongly adapted to the media and application areas and determine the structure of the book (Fig. 4.28). Pictures are described in Chap. 5, written texts in Chap. 6, spoken language in Chap. 7, and action sequences processed by reinforcement learning in Chap. 8. Important methods for the media are printed in bold. In addition, there are some overlapping areas where one medium is transformed into another. This is indicated in the graphic by an arrow (→) and

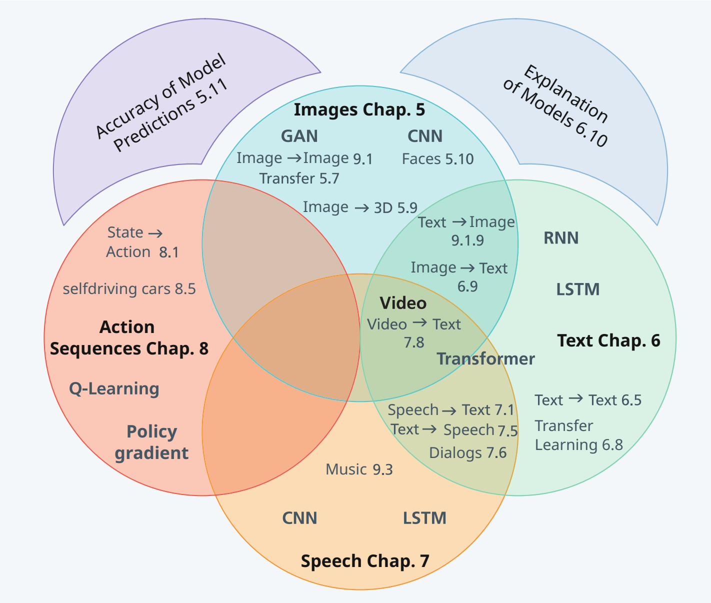


Fig. 4.28 The media and application domains considered in this book with their associated model architectures. Transformations between media are indicated by ⟶ and the corresponding section number

the section with the description of this transformation. There is, for example, the transformation speech → text in speech recognition or the transformation text → text in translation from one language to another. “Transfer Learning” states that models are trained in one application domain and transferred to a related application domain. At the top right and on the top left are two more general sections on model accuracy and model explanation.

4.8 The Design of a Deep Neural Network Is a Search Process

The previous section has made it clear that Machine Learning methods typically do not achieve 100% accuracy. Rather, it is a long process to find and train a model that has sufficient accuracy for the particular application. This process includes finding suitable data, testing different model variants, and evaluating the models in a meaningful way.

4.8.1 Selection of the Hyperparameters of the Network

We have seen that the accuracy of the DNN depends on a variety of design choices and control variables. Too many parameters (too much complexity) will result in overfitting, while too little flexibility (low representational power) of the DNN will lead to underfitting. Therefore, the task of the network designer is to choose the right combination of hyperparameters. He has to find a balance between too little and too much complexity.

Hyperparameters of the Network

- Number of layers and their connections
- Layer types
- Length of hidden vectors in every layer
- Type of activation functions
- Hyperparameters of regularization
- Type of loss function
- ...

Hyperparameters of Optimization

- Type of the optimization method
- Global learning rate
- Scheme to reduce the learning rate
- Type of accuracy measure on the validation set
- Intervals for saving the model
- Size of the minibatch

As the previous box shows, there are quite a few hyperparameters that need to be specified. Their values determine the accuracy of the model predictions on the validation and test set.

In general, the search for the best hyperparameters is done manually (Goodfellow et al., 2016, p. 416). Based on the available data and the research question, the first step will be to search the web for a network for a similar application. An invaluable resource for this is the GitHub website (Github, 2019), which contains a huge collection of executable neural network programs. Here, many solutions to problems are already provided together with sample data for TensorFlow, PyTorch, or another DNN toolkit. In addition, the authors provide information on available data, describe the quality of the results, and refer to related publications. This search will yield valuable information on the basic network architecture for the current problem and perhaps already provides the basic framework for the planned application.

Unfortunately, there is no magic formula to determine the best architecture. One has to find the best combination of hyperparameters in experiments. The search is usually done in stages, starting with relatively few layers and parameters. Then, the size and number of layers are gradually increased until the loss on the validation data stagnates or increases. Figure 4.29 describes this process, in which different model variants are successively generated and compared with respect to their accuracy on the validation data.

Often the result of a model run is partly obscured by random effects. For example, the initial values of the model parameters are usually chosen at random and the results on the test set depend on them. To determine the mean value of the results, one must then repeat these experiments several times with different starting values. Alternatively, one can fix the starting values for the random

Define a simple model with few layers and repeat the following steps:

- modify one or more hyperparameters: Number of layers, length of hidden vectors, learning rate, regularization parameters, ...
- train the model on the training data
- calculate the accuracy on the validation data
- if the model is better than the previous one, store the vector of hyperparameters

Fig. 4.29 Search for good combinations of hyperparameters

number generator to one value. This is especially important if the effect of many hyperparameters is investigated. Then it is recommended to perform a controlled experiment (Goodfellow et al., 2016, p. 415) to separate random effects from significant impacts. If this procedure is not followed, "significant" results cannot be replicated in control experiments. This phenomenon of a "replication crisis" unfortunately occurs in many current research publications (Cockburn et al., 2020).

4.8.2 The Standard Model Search Process Leads to Better Models

During the iterative search of a model, one can make many mistakes and forget important aspects. Therefore, a standard approach to this process has been developed. The Cross-Industry Standard Process for Data Mining (CRISP-DM) is a guideline (Shearer, 2000) developed jointly by many interested companies and research institutes. It should be used not only for DNN, but for any Machine Learning process, and includes the following steps (Fig. 4.30), which are illustrated by image classification as an example:

Business Understanding. This considers the business objective of the organization that needs to be achieved. This determines the goal of the analysis and the success criteria by which the entire analysis will be evaluated. This can be not only an optimal error value, but also criteria such as computation time, robustness or explainability.

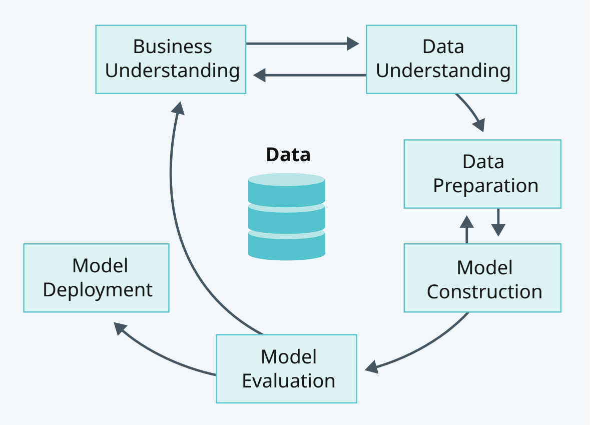
> In an image classification application, the analysis goal could be, for example, the detection of defective products based on photos.

Data Understanding. This includes finding and defining suitable data, examining it, and making an initial check for errors and outliers.

> The classification of images requires, for example, photos in a suitable resolution, which are then annotated as "defective" or "ok".

Data Preparation. This step usually requires a very large amount of work. It includes the actual collection and acquisition of data, consolidation and cleaning,

Fig. 4.30 Phases in the construction of a model according to the CRISP-DM scheme



including the detection of data errors. Then, appropriate training samples and features must be selected. Furthermore, the data must be randomly partitioned into a training, test and validation sets. Representativeness for the application must be maintained, i.e. the data must reflect the distribution of the training examples in the application. Often, data transformations are also necessary.

> When classifying images, for example, a training data set of annotated photos must be collected and the photos must be inspected for imaging errors. If necessary, the brightness and contrast of the photos must be changed.

Model Construction. This includes the selection of the model, taking into account the goal of the analysis and the characteristics of the data. It doesn't always have to be a deep neural network, because in many application cases, classical Machine Learning methods are more efficient and robust. Model construction is an iterative process in which the model is trained using the training data. A detailed description can be found in the next section.

> In our example application, we need to train different image classification models and compare the results. These methods are presented in Chap. 5.

Model Evaluation. This measures how well the model performs on validation data that were not used for training. The quality measures depend on the goal of the analysis and the type of model selected. Only when the model construction is completed, the test set may be used for a final evaluation. If fundamental problems arise during the evaluation, the goal of the analysis must be changed and the whole process must be started again.

> In the example application for image classification, it must be checked whether the achievable accuracy of the classification is sufficient. It is also necessary to verify if the process is robust against possible disturbances during operation (dirt, image errors, vibrations, etc.). If this is not the case, the process may have to be restarted.

Model Deployment. This includes applying the model to new data. Often, the model is also incorporated into the business processes and workflows. If the data changes over time, the model must be redefined and retrained.

> In our example application, we analyze whether the predicted quality of the image classification can actually be maintained.

4.8.3 Automatic Search of Model Architectures and Hyperparameters

It seems obvious to automate the search for the best architecture. This is a high-level optimization problem. This can be complicated because, for example, the introduction of another layer in the DNN increases the number of hyperparameters (length of the hidden vector, regularization hyperparameters, etc.). Moreover, these nested optimizations have high computational cost. A number of methods (Goodfellow et al., 2016, p. 420) have been suggested for this hyperparameter optimization problem (Fig. 4.31):

- Grid search performs an exhaustive calculation of the model performance on the validation data for all combinations of hyperparameter values. It is inefficient for more than two hyperparameters.
- Random search randomly chooses a combination of hyperparameter values each time. This can be more efficient than grid search, especially if the final model quality is determined by only a few hyperparameters, or if many hyperparameters are to be fitted.
- Bayesian optimization performs random modifications, but focuses on the potentially best combinations of hyperparameter values. Here, the performance of the previously evaluated hyperparameter values is described by a model,

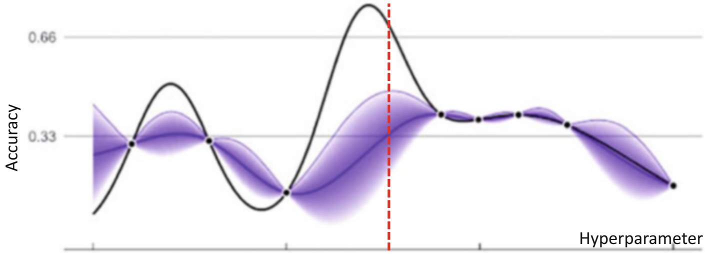


Fig. 4.31 The Bayesian optimization considers the previously evaluated function values (•) and estimates a mean value curve (blue) and fluctuation widths (purple) for the in-between values. The next hyperparameter value to be evaluated is selected where a high accuracy value is plausible (dashed red). The black line symbolizes the unknown actual accuracy. Image credits in Appendix A.3

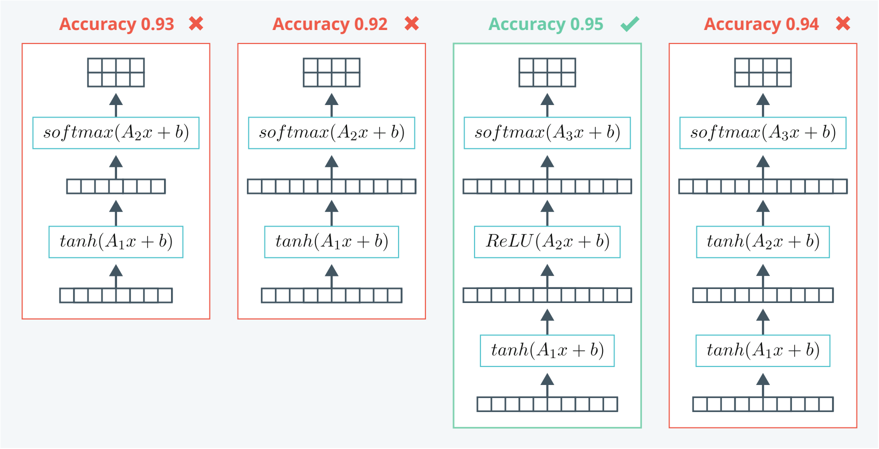


Fig. 4.32 AutoKeras is an automatic procedure that autonomously searches for the optimal network architecture and the values of the hyperparameters. During this process, the length of hidden vectors can be modified, new layers may be inserted, activation functions can be changed, etc. A performance measure, e.g. accuracy, is used as a selection criterion

e.g., a Gaussian process model. This model predicts not only the average expected performance values, but also the inaccuracy of the prediction. The next hyperparameter to be evaluated is now selected in a region where high performance values are possible, taking uncertainty into account. Thus, the model pursues a balance between the exploration of hyperparameters and their evaluation (concentration on the best value combinations). In practice, Bayesian optimization achieves better results than grid search and random search.

- One can also extrapolate Bayesian optimization to the case where not only continuous parameters are modified, but also discrete changes are made to the model structure, e.g. addition of an extra layer in a neural network (Jin et al., 2019). The authors have implemented this procedure and made it available under the name AutoKeras as a module of the Keras toolkit (Fig. 4.32).

Google uses these techniques extensively to develop deep neural networks and also provides a way for a user to optimize their neural network architecture (Google, 2019).

Previous methods for finding an optimal network architecture achieve good results, but require a very high computational effort (often thousands of GPU days). Here, normal parameter optimizations alternate with discrete changes, e.g. addition of additional layers or operators.

An alternative approach to hyperparameter optimization constructs a complex differentiable search problem, which can be solved entirely by gradient descent. Noy et al. (2020) additionally introduce a so-called “annealing”, which is well known from the optimization literature. Here, “temperature parameters” are used to modify the shape of the cost function in such a way that small weights can be “turned off”

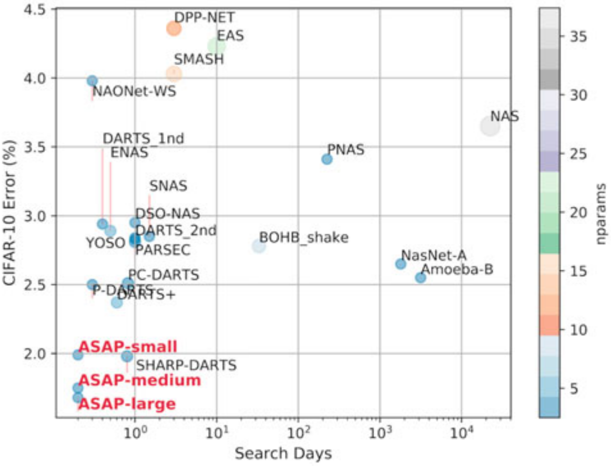


Fig. 4.33 Search duration and classification errors of different architecture searches for CIFAR-10 image classification. The costs of search are measured by GPU days. The size of the circles shows the memory requirements and the color encodes the number of parameters. The ASAP method requires dramatically less computation time and achieves better accuracy scores (Noy et al., 2020)

and entire network parts can be made inactive. Compared to previous approaches, their ASAP (Architecture Search, Anneal and Prune) method requires several orders of magnitude less computation effort and reduces the duration of architecture search from GPU-years to a few GPU-hours. Figure 4.33 shows the search duration and achieved accuracy for CIFAR-10 image classification with 10 classes. Here, ASAP can improve the classification error from 2.55% to 1.99% with a computational cost of only 0.2 GPU days.

4.9 Biological Neural Networks Work Differently

Although artificial neural networks were inspired by the biological processes that scientists began observing in the brain in the 1950s, they differ from their biological counterparts in many ways. Similarly, birds inspired modern airplanes and horses inspired locomotives and automobiles, but none of today's transportation vehicles resemble any of these living animals.

The brain of animals and humans is a biological neural network and consists of neurons, special body cells (Goodfellow et al., 2016, p. 13). Neurons can transmit an impulse to other neurons via a long extension, the axon, through a mixture of electrical and chemical mechanisms (Fig. 4.34). The coupling to other neurons through so-called synapses is weighted, i.e., of varying strength. When the weighted

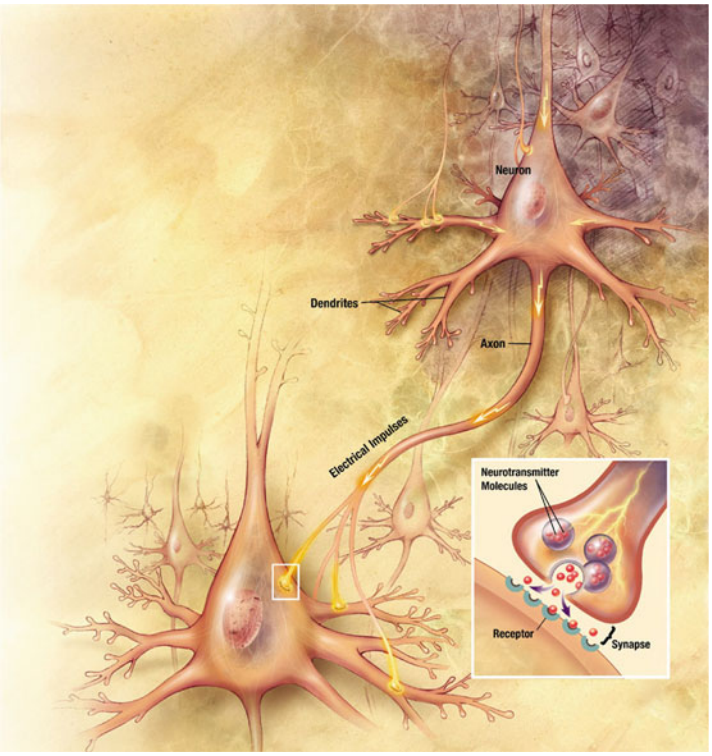


Fig. 4.34 A neuron sends a series of impulses via the axon. Both electrical and chemical transmission mechanisms are used to transfer the information to other neurons. The connections to other neurons (synapses) are weighted, meaning they have different "strengths." Importantly, the information on the axon is encoded by the frequency of impulses (frequency modulation). Image credits in Appendix A.3

sum of input signals to a neuron exceeds a certain threshold, the neuron "fires" and sends out impulses itself.

In the 1960s, Hubel and Wiesel found that the mammalian visual system is organized in layers. Signals from the retina in the eye are passed through different layers, extracting increasingly complex image features (Sect. 5.1.2). Simlar to this architecture, artificial neural networks have been developed, which consist of several layers containing linear transformations and subsequent nonlinearities. The nonlinearities are necessary for the networks to represent complicated relationships between inputs and outputs (Sect. 4.2). Backpropagation (Sect. 4.4) was suggested as a training algorithm.

From today's perspective, biological and artificial neural networks have a number of differences:

Connection pattern: Biological neurons are not strictly organized in layers. They have a small number of strongly connected neurons and many sparsely connected neurons.

Information transmission: the transmission of impulse along the axon of a neuron is frequency modulated, thus its information is encoded by the frequency of the impulses. This is a substantially different representation than the fixed values in the vectors and tensors of a deep neural network.

Learning: We still do not understand exactly how biological brains learn. Brain fibers grow and connect to other neurons, connections can be altered in their function, and synapses can be strengthened or weakened depending on their relevance. Neurons that are active together gradually form connections. Artificial neural networks, on the other hand, currently have mostly a fixed structure of connections. The backpropagation method for computing derivatives is biologically implausible, especially because synapses can transmit impulses in only one direction. However, more biologically realistic learning methods are now being discussed (Mesnard et al., 2019). Therefore, deep neural networks can only serve as a very limited model for information processing in the brain.

4.10 Summary and Trends

To solve more complex problems, the models must form nonlinear separating surfaces to handle prediction and classification tasks. This is only possible if additional layers with nonlinear functions are inserted into the neural network. Through training, these layers form increasingly complex representations of the inputs, which can then be used by the final layer to generate the desired outputs. There are also nonlinear classical Machine Learning methods, but their performance is almost always surpassed by DNN for large data.

In recent years, toolkits for DNN have been introduced, which allow a simple formulation of the common models and also automatically calculate the gradients and perform the optimization. Furthermore, they also enable the evaluation of the model and the execution of the computations on parallel hardware.

It is important to test for underfitting or overfitting of the models. Underfitting occurs when the model has too little representational power for the forecasting task. Overfitting happens when the model is too complex and adapts to random variations in the data. There are a number of regularization techniques to avoid overfitting, such as dropout or batch normalization.

In recent years, a number of important network types have emerged, including feedforward networks, convolutional neural networks (CNN), recurrent neural networks (RNN), reinforcement learning networks, generative adversarial networks (GAN), and autoencoders. These networks are discussed in the remaining chapters.

Because of the many possible hyperparameter values, the structure of DNN must be adapted to achieve a good prediction. This is done in a stepwise procedure, checking the performance of the model on a separate validation set. Automatic methods for searching network structures have recently been suggested, which are potentially much more efficient.

Trends

- There is a race between the deep learning platforms PyTorch and TensorFlow for the favor of the users. TensorFlow scores with a large number of implemented example models, and PyTorch is catching up with a more flexible language design.
- Automatic search of the model structure using AutoML techniques is becoming increasingly important. There are approaches where the model architecture is now found by gradient descent, which requires much less computational effort.
- Deep neural networks often have very many parameters. To reduce the associated memory requirements and computing effort (e.g., for smartphones), one can reduce the number of parameters. Frankle and Carbin (2019) use the initialization values of the full model for the reduced model and can thus often decrease the number of parameters to 10% without loss of accuracy.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep learning with differential privacy. In *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.* (pp. 308–318).

Bassily, R., Belkin, M., & Ma, S. (2018). On exponential convergence of SGD in non-convex over-parametrized learning. Preprint. ArXiv181102564. arXiv: `1811.02564`.

Bruna, J., & Mallat, S. (2013). Invariant scattering convolution networks . *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1872–1886.

Brutzkus, A., & Globerson, A. (2019). Why do larger models generalize better? A theoretical perspective via the XOR problem . In *Int. Conf. Mach. Learn.*, PMLR (pp. 822–830).

Chollet, F. (2019). *Keras*. https://keras.io/

Cockburn, A., Dragicevic, P., Besançon, L., & Gutwin, C. (2020). Threats of a replication crisis in empirical computer science. *Communications of the ACM*, *63*(8), 70–79.

DL-Software. (2019). *DL-Software. Comparison of deep-learning software*. https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software

Eldan, R., & Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conf. Learn. Theory* (pp. 907–940).

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Int. Conf. Learn. Represent.*

Github (2019). *Github*. https://github.com/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (Vol. 1). Cambridge: MIT Press. https://www.deeplearningbook.org/

Google (2019). *Cloud AutoML-Dokumentation*. Google Cloud. https://cloud.google.com/automl/docs?hl=de (visited on 01/18/2022).

Hardt, M., Recht, B., & Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *Int. Conf. Mach. Learn.*, PMLR (pp. 1225–1234).

Hornik, K., Stinchcombe, M., White, H., et al. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Jin, H., Song, Q., & Hu, X. (2019). Auto-Keras: An efficient neural architecture search system. In *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* (pp. 1946–1956).

Kaggle (2017). *Kaggle 2017 Survey Results*. https://www.kaggle.com/amberthomas/kaggle-2017-survey-results

Mesnard, T., Vignoud, G., Sacramento, J., Senn, W., & Bengio, Y. (2019). Ghost units yield biologically plausible backprop in deep neural networks. Preprint. ArXiv191108585. arXiv: `1911.08585`.

Minsky, M., & Papert, S. A. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.

Neyshabur, B., Salakhutdinov, R. R., & Srebro, N. (2015). Path-SGD: Path-normalized optimization in deep neural networks. In *Adv. Neural Inf. Process. Syst.* (pp. 2422–2430).

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P., & Hluchỳ, L. (2019). Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey. *Artificial Intelligence Review*, *52*, 77–124.

Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., & Zelnik, L. (2020). ASAP: Architecture search, anneal and prune. In *Int. Conf. Artif. Intell. Stat.*, PMLR (pp. 493–503).

Pytorch (2019). *PyTorch*. https://pytorch.org/

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Shearer, C. (2000). The CRISP-DM Model: The new blueprint for data mining. *Journal of Data Warehousing*, *5*(4), 13–22.

Tschannen, M., Bachem, O., & Lucic, M. (2018). Recent advances in autoencoder-based representation learning. Preprint. ArXiv181205069. arXiv: `1812.05069`.

Werbos, P. (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph Diss., Harvard University.

Chapter 5
Image Recognition with Deep Neural Networks

Abstract Image recognition is about finding automatic methods to identify objects and their arrangement in an image or photo. This includes classifying the image objects and determining their position in the image. The majority of DNNs for image processing are Convolutional Neural Networks (CNN). They use layers with small receptive fields (convolutions), which are shifted over the pixel matrix of the input image. They are capable of detecting local image features. In addition, pooling layers are used to aggregate features locally. Modern CNNs contain hundreds of these layers, which can successively recognize more complex image features. Some of them make fewer image classification errors than humans. Special variants have been developed to determine the position of objects in images with pixel precision. Finally, models for estimating the inaccuracy of image classifications are presented, and the influence of image distortions and intentional image manipulation on classification accuracy is discussed.

5.1 What Does Image Recognition Actually Mean?

Image recognition uses automatic methods to identify objects in an image or photo and label them with their name (class). In addition, image understanding aims at a more detailed description of the scene depicted in an image. This chapter mainly discusses image recognition, while aspects of image understanding are presented in Sects. 6.9 and 7.8.

5.1.1 Types of Object Recognition in Images

There are different types of image recognition tasks (Fig. 5.1). Object classification in images determines the name (class) of an object contained in the image, which is usually the most important or prominent object. If one is also interested in the position of the object in the image, this is the task of object localization in images. The position of an object is indicated by a bounding box (enveloping rectangle).

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_5

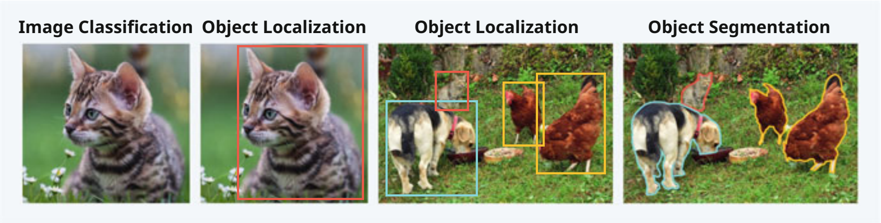


Fig. 5.1 Different tasks of image recognition: object classification assigns the "most important" object of the image to a class. Object localization marks the position of an object by a bounding box. Object segmentation identifies the pixels of an object. Image credits in Appendix A.3

In general, an image contains several objects, of which the most important are classified and located by bounding boxes during object localization. Because location by bounding boxes is relatively inaccurate, one often wants to determine the pixels of the image through which an object is represented. This task is called object segmentation.

There are many application areas for image classification. One obvious use is to organize users' personal photo collections, e.g., by classifying images according to content criteria or to assist in image search. These techniques can also be employed for large image databases, e.g. in the medical field, where image classification is used for the diagnosis of diseases. Another application is the search of images that are harmful to minors in social media. Finally, image recognition is a fundamental technology for self-driving cars

5.1.2 Inspirations from Biology

Greek philosophers already tried to fathom the process of human vision. Aristotle believed that light is reflected from objects, and when this reflected light hits the eye, "vision takes place" (Krämer 2016). Johannes Kepler discovered that light hits the lens of the eye. There it is refracted and bundled in such a way that an inverted, reduced image of the outside world appears on the retina, which is then further processed by the brain. But it was completely unclear how the recognition of objects in the brain could work.

Only Hubel and Wiesel were able to show how the visual system works by measurements on single neurons of cats and monkeys (Hubel & Wiesel 1962). They found that there are "simple" neurons in the V1 area of the brain that respond only to features located at a particular position on the retina. Such a feature might be, for example, a bar of light in a certain direction. This demonstrated that there are local detectors of edges in the brain that affect only small, circumscribed areas of image receptors in the retina. In addition, there are "complex" neurons that combine the output of local detectors in a small area. In areas V2, V4, etc. of the brain,

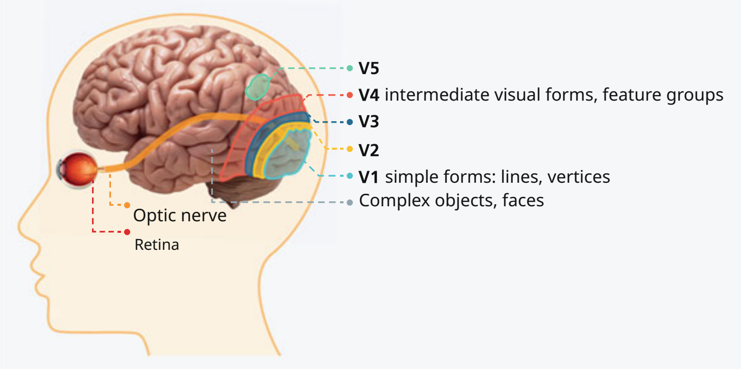

V5
V4 intermediate visual forms, feature groups
V3
V2
V1 simple forms: lines, vertices
Complex objects, faces
Optic nerve
Retina

Fig. 5.2 Image information from the eye is transmitted from the retina to area V1 in the visual cortex, where simple shapes, corners, and lines are recognized. In areas V2, V3, V4, and V5, intermediate visual shapes and feature groups are identified, and in nearby brain regions, complex objects and faces are detected. Image credits in Appendix A.3

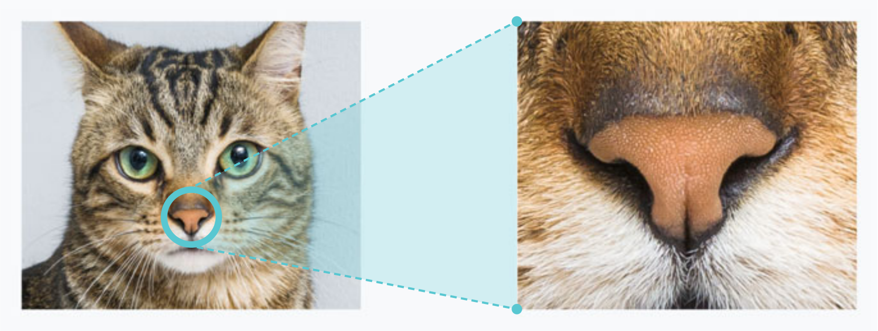


Fig. 5.3 The first step in recognizing a large object (cat) can be done by using small patterns (e.g. cat nose). Image credits in Appendix A.3

the information found in V1 is further processed and increasingly complex objects, such as faces, are recognized. This is illustrated in Fig. 5.2 and described in detail in Visual (2020). Hubel and Wiesel received the Nobel Prize for their research.

For example, the recognition of a cat in an image does not happen in one step, but small partial patterns in a small region of an image are recognized first, e.g. the nose of a cat (Fig. 5.3). The nose of another cat will look similar and also a small partial pattern can be identified with little effort. Since a cat's nose can appear almost anywhere in an image, it must be possible to shift the cat's nose detector over any subregion of the image. Crucially, the detectors for the subpatterns are not pre-programmed, but are automatically trained by visual impressions. Hubel and Wiesel were able to show, for example, that the visual cortex of a cat is irreversibly disrupted if a kitten is deprived of normal visual experience early in its life. Finally, through the interaction of many individual detectors, the cat is recognized as a whole and distinguished, for example, from a dog.

A first neural network incorporating these findings was presented in 1980 (Fukushima 1980). This Neocognitron contained layers that divided the input into many image areas and processed each of these image areas in the same way. Other layers performed pooling, i.e., local averaging over many small image areas. The layers were arranged several times on top of each other. But it was only at the turn of the millennium that powerful networks with this structure were successfully deployed, the Convolutional Neural Networks (LeCun et al. 1998). Figure 5.4 shows the history of automatic image recognition.

5.1.3 Why Is Image Recognition Difficult?

Object classification in images is about assigning the objects within an image to a given list of classes, e.g. dog, cat, truck, plane, etc. There are a number of effects that can make classification difficult (Fig. 5.5 on the left). Even a change in the distance to the object or the reflection or rotation of the object can greatly change the image. Differences in illumination modify or suppress contours of the object. Objects can change their shape and occlusions make parts of the object disappear. Finally, depending on the environment, it may be difficult to distinguish the object from the background. In addition, objects of different colors and shapes are often combined in one class, e.g. bridges in Fig. 5.5 on the right.

5.2 The Components of a Convolutional Neural Network

5.2.1 Convolutional Kernels Analyze Small Image Areas

A Convolutional Neural Network (CNN) is inspired by the mammalian visual system. It consists of one or more convolutional layers, each followed by a pooling layer. This sequence of layers can be repetitive. In image processing, the inputs are always at least two-dimensional, e.g., the pixel matrix of a grayscale image. The most important application of CNNs is object classification in images with the goal of automatically identifying different classes of objects (dog, cat, car, …) in these images. This identification of objects is learned using extensive training data, which include for each image the class names of the objects it contains.

It has been shown that the best results are obtained if the image is decomposed into small overlapping sub-areas and these are first analyzed separately. The kernel of a convolution layer is a small matrix of parameters. It typically has the dimensions 3×3 or 5×5. An input matrix together with a kernel is shown in Fig. 5.6. For

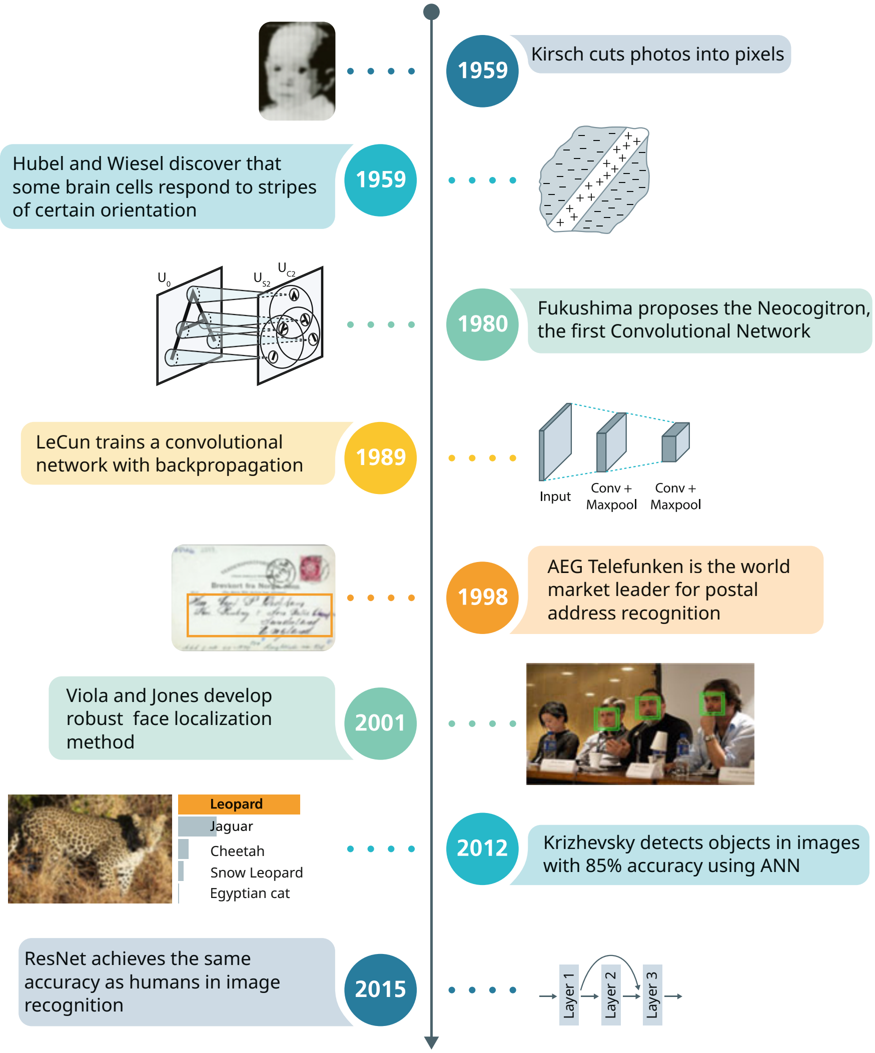

1959
Kirsch cuts photos into pixels
Hubel and Wiesel discover that some brain cells respond to stripes of certain orientation
1959
U_0
U_{S2}
U_{C2}
1980
Fukushima proposes the Neocogitron, the first Convolutional Network
LeCun trains a convolutional network with backpropagation
1989
Input
Conv + Maxpool
Conv + Maxpool
1998
AEG Telefunken is the world market leader for postal address recognition
Viola and Jones develop robust face localization method
2001
Leopard
Jaguar
Cheetah
Snow Leopard
Egyptian cat
2012
Krizhevsky detects objects in images with 85% accuracy using ANN
ResNet achieves the same accuracy as humans in image recognition
2015
Layer 1
Layer 2
Layer 3

Fig. 5.4 The history of automatic image recognition. Image credits in Appendix A.3

clarity, the input matrix and kernel have been filled with integers; usually these are real numbers.

At the beginning, the kernel is placed in the upper left corner of the input matrix (drawn in red in Fig. 5.6 on the left). Each value in the red partial matrix—the receptive field—is now multiplied by the corresponding value of the kernel and the result is added (see side calculation further right). The resulting value 2 is now stored in the upper left corner of the result matrix.

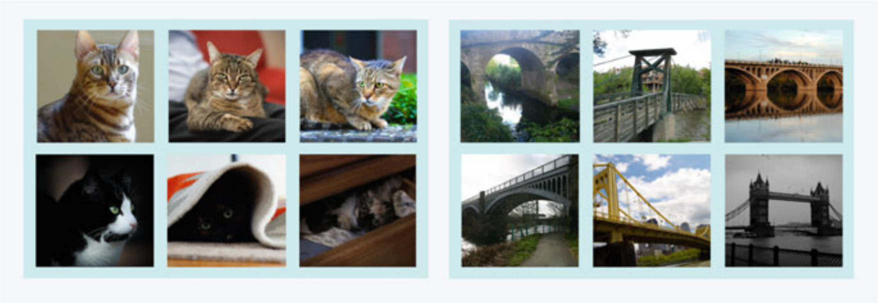


Fig. 5.5 Variations in the images of cats (on the left) that make object classification difficult: Change of perspective, changed illumination, change of shape, occlusion, and cluttered background. Within a class of objects there are may be strong differences, here for example in the class “bridge” (right). Image credits in Appendix A.3

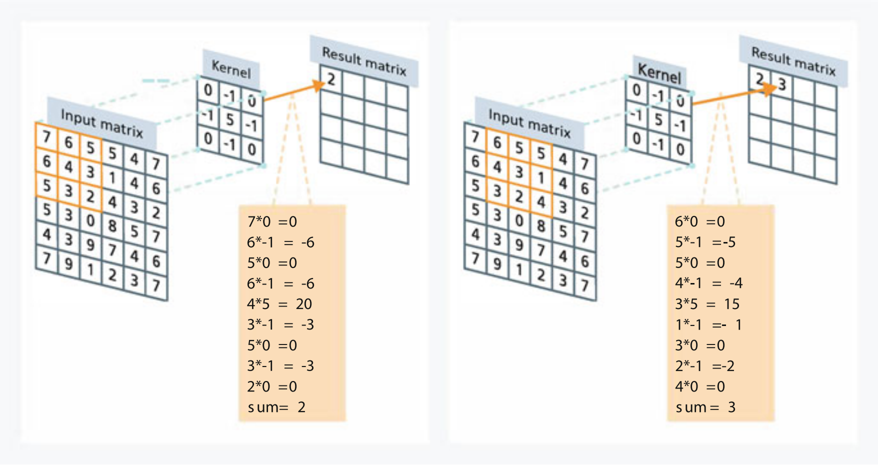


Fig. 5.6 First calculation step (on the left) and second calculation step (on the right) in the convolution layer, each for a shifted small area of the input matrix. Thereby, the kernel is successively “shifted” over the entire input matrix and the result matrix is filled

For the second calculation step of the convolution layer, the receptive field is shifted one position to the right in the input matrix and the calculation is repeated with the same kernel (Fig. 5.6 right). The result is stored one position further to the right in the result matrix. This process is repeated for all positions of the input matrix until the result matrix is filled. As you can see, the result matrix is slightly smaller than the input matrix. Usually, a number b is added to each value of the result matrix, which increases or decreases the result independently of the input values. This bias was omitted here for the sake of simplicity.

It is important to note that a value in the result matrix depends entirely on the pixel values in the particular receptive field. Furthermore, the parameter values of

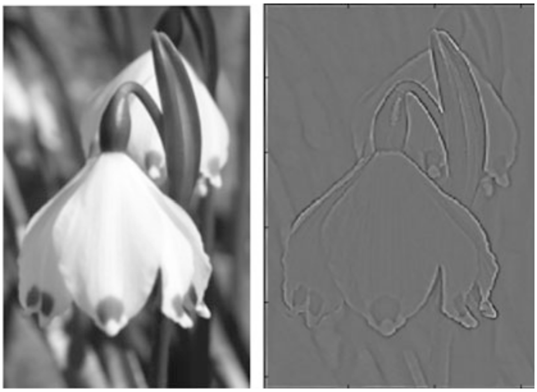


Fig. 5.7 Input image (on the left) and result values in grayscale representation for the Laplace kernel. Image credits in Appendix A.3

a kernel do not change when moving, so the features extracted by a kernel are translation invariant.

Depending on the values in the kernel matrix, different local features are extracted from the input matrix. The kernel matrix $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$, for example, is called discrete Laplace operator and produces high values at an edge, i.e. at a transition between light and dark. Figure 5.7 shows for an input matrix the result matrix for this kernel. It is also possible to move a kernel not only by 1 position, but also by 2 positions. If this stride is greater than 1, the size of the result matrix decreases accordingly.

The input matrix can also be written row by row as vector x. If one sets the values of the kernel to certain positions of a vector a_1, the scalar product $a_1 * x$ produces the first value of the vectorized result matrix. Figure 5.8 shows how to continue this for all values of the vectorized result matrix and how to form a matrix A from the a vectors. This computation produces the same result as the corresponding convolution layer. Thus, a convolution layer is an affine transformation $y = A*x+b$, where the transformation matrix A has very few nonzero values. Moreover, all nine parameters of the kernel appear in each row of A. Such multiple use of parameters is also called parameter sharing. This reduces the danger of overfitting.

5.2.2 *Different Kernels in a Convolution Layer Compute Many Features*

In a convolution layer, usually a number k of different kernels (e.g. $k = 128$) are used, generating k result matrices that are combined into a tensor (Fig. 5.9). This allows a large number of complementary features to be computed in parallel in one layer, which can be used to classify the image more accurately.

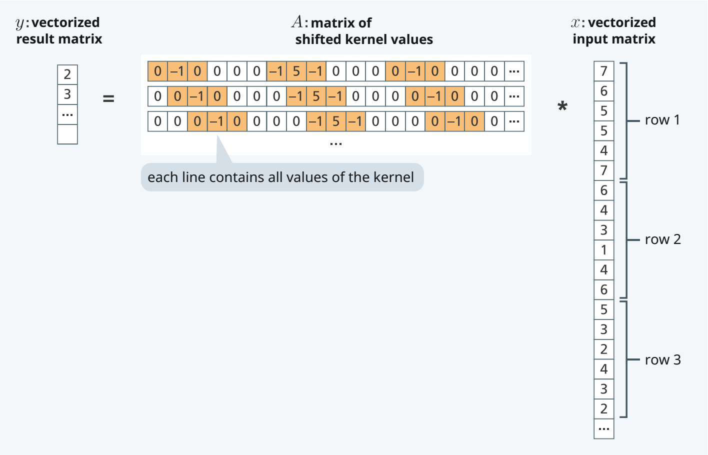

y: vectorized result matrix
A: matrix of shifted kernel values
x: vectorized input matrix
=
*
each line contains all values of the kernel
row 1
row 2
row 3

Fig. 5.8 You can arrange the input matrix from the convolution layer in Fig. 5.6 row by row as a vector (right). Further one can form a matrix A, in which in each row the elements of the kernel appear at different positions (middle). The multiplication with this matrix leads to the same result as the convolution. Thus, a convolution layer with a kernel corresponds to an affine transformation $y = A * x + b$

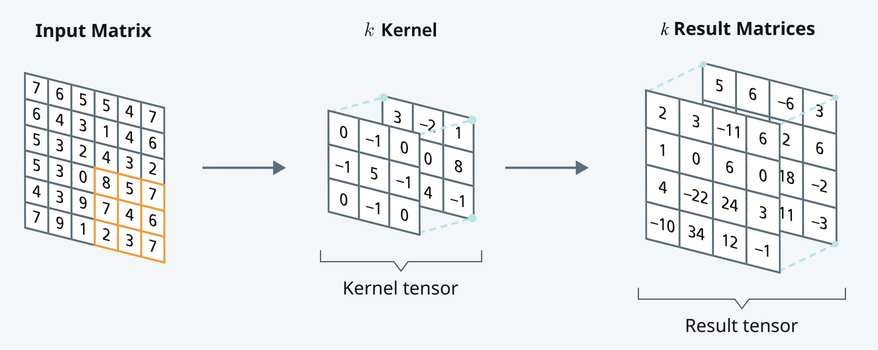


Fig. 5.9 A convolution layer contains k kernels and k result matrices, each of which are combined into tensors

The kernels of the CNN are small matrices of parameters. Thus, they are not predefined, but trained, i.e., automatically adjusted so that the resulting network can predict the desired final result (e.g., class name of the object contained in the image). The number of parameters of a kernel is very small, e.g. 9 parameters for a 3×3 kernel. Even if 128 different kernels are used, the number of parameters $(9 * 128)$ is still much smaller than for a fully connected layer. This would have, for example, a total of about 1 million parameters for an image with 1000 pixels.

The result matrices are then transformed with a nonlinear activation function. For CNNs this is usually the rectified linear unit (see ReLU, Fig. 4.5).

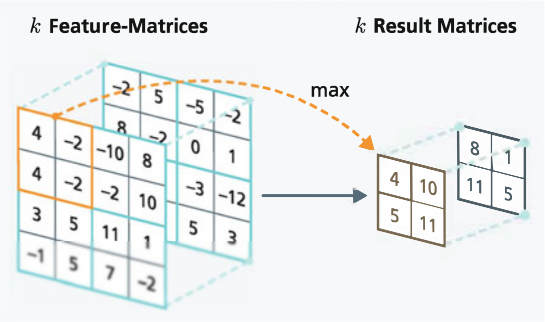


Fig. 5.10 In a pooling layer, the features of the result matrices of the convolution layer are combined locally. An aggregation function is calculated for small ranges, e.g. 2 × 2 fields. Usually, a maximum is computed separately for each result matrix

5.2.3 The Pooling Layer Selects Most Important Feature Values

If one has $k = 128$ kernels, the result matrices together contain much more feature values than the input pixel image. Therefore, the number of feature values must be reduced by a pooling layer. For this purpose, several neighboring features are combined. This reflects the fact that the exact position of a feature is not so important, the approximate position is sufficient.

In max-pooling, for each feature matrix, quadratic subareas, e.g. 2×2 squares, are aggregated by a number, the maximum, and stored in a result matrix (Fig. 5.10). The other values of the subarea are ignored. Which feature is chosen as the maximum depends, of course, on the particular input image. This aggregation can be repeated for each feature matrix and produce another result matrix. By so doing, e.g., for 2×2 squares, a data reduction to one quarter is obtained. However, a pooling operation can also be performed over all values of the result matrices at one position with a correspondingly higher data reduction. Overall, this approach has three important advantages:

- The image features become approximately invariant to small shifts in the input image, since max-pooling only maps the maxima of different input positions.
- If the first and second CNN layers each have a 5×5 kernel, the features of the second layer are affected by the pixels of a 9×9 pixel area. Thus, the receptive fields become larger, with pooling providing additional magnification. This allows larger object features to be captured.
- The computational effort and memory requirements are reduced.

Experience has shown that pooling by averaging leads to worse classification results than forming the maximum.

5.3 A Simple Convolutional Neural Network for Digit Recognition

For our MNIST digit recognition dataset (Sect. 3.2.3), a CNN was constructed (Keras 2019), whose layers are shown in Fig. 5.11. It uses two convolutional layers at the beginning with receptive fields of size 3 × 3 and 32 or 64 kernels, respectively, and a ReLU activation function. This results in an image region of size 24 × 24 with 64 features. A MaxPool layer computes the maximum in a receptive field of 2 × 2. This results in an image area of 12 × 12 with 64 features. On top of this, a dropout layer is added for regularization, which randomly drops 25% of the elements. The result is reshaped into a vector with 9216 features. This is followed by an affine transformation with ReLU activation and then another dropout layer that omits 50% of the elements. Finally, there is a logistic regression model with softmax activation, which gives as output the vector of probabilities for each class (i.e. 10 digits).

We have seen in Fig. 5.8 that a convolutional layer corresponds to an affine transformation in which the parameters of a kernel are repeated in every matrix row. This parameter sharing is the essential difference that must be considered when training a CNN. Otherwise, the training of such a network proceeds in principle as for a multilayer feedforward network (Sect. 3.6). The negative log-likelihood of a minibatch is again used as the loss function (Fig. 3.28). The model is trained with the stochastic gradient descent.

As Fig. 5.12 shows, the accuracy on the validation set increases rapidly during training and reaches a value of 99.13% after 10 epochs. Thus, the error (0.87%) is only about one third of the error (2.5%) of the two-layer neural network in Fig. 4.8. This demonstrates the effectiveness of convolutional layers for image classification.

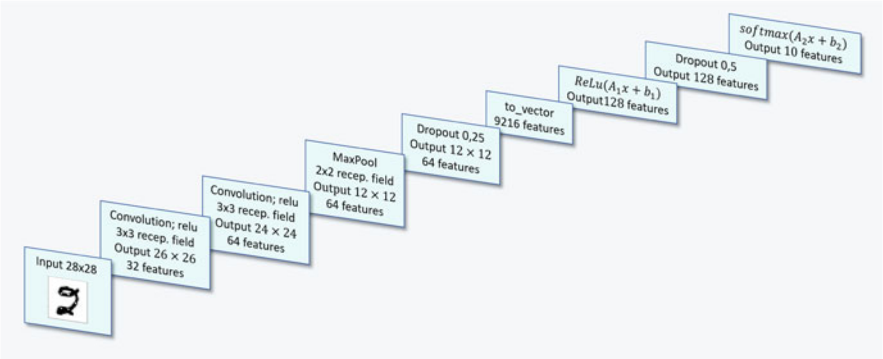


Fig. 5.11 Layers of the CNN for MNIST digit recognition. On the bottom left, the 28 × 28 pixel matrix is shown as input. The output of each layer is characterized by the result matrix size and the number of features in each position. Output of the last layer is the probability vector of possible digits

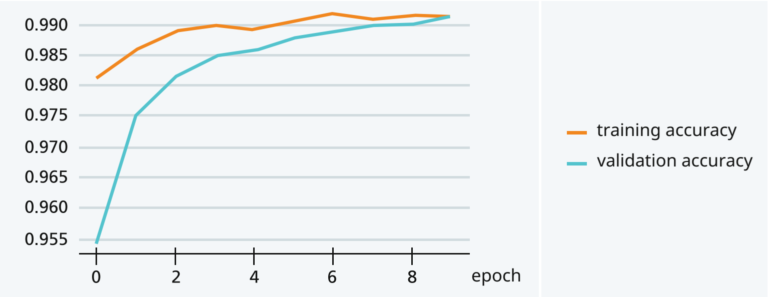


Fig. 5.12 Increase of the CNN accuracy with the number of epochs for the MNIST digit classification on the training and validation sets

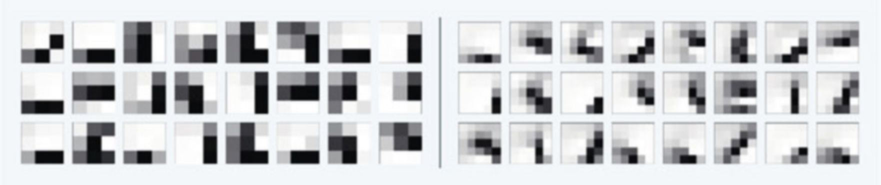


Fig. 5.13 For these inputs of the receptive field the kernels of the first layer (on the left) and the second layer (on the right) have the highest outputs after training with the MNIST data. Results are shown for only 24 kernels. These features are apparently particularly effective in classifying the digits

It is important to note that when the trained CNN is applied, the dropout probability is set to 0.0, i.e., no more weights are set to zero.

Figure 5.13 shows the inputs of the receptive field for which the kernels of the first and second layer reach their highest output values after training. Here, the kernels of the first layer have a receptive field of size 3×3 and generate 32 features. These are mostly horizontal or vertical edges. The second layer kernels have a joint receptive field of size 5×5 and generate 64 features after training, on which the final classification of digits is based. We observe vertical, horizontal, and diagonal lines and edges, as well as other types of subpatterns, which appear to be particularly informative for digit classification. Yann LeCun in Fig. 5.14 played a decisive role in the development of the CNN.

5.4 ImageNet Competition Boosts Method Development

Since 2010, there has been an annual competition (Sect. 2.1) where different research groups can apply their image recognition software to the ImageNet data (Russakovsky et al. 2015). There are 1000 object categories as well as a training set of about 1.2 million images and a validation set of fifty thousand images.

Yann LeCun
studied computer science in Paris. In 1987, he was a PostDoc of Geoffrey Hinton in Toronto. He then joined Bell Labs, where he developed learning methods for Convolutional Neural Networks (CNN). He suggested "optimal brain damage" as a regularization method and built probabilistic networks, which he applied to handwriting recognition and optical character recognition (OCR). The bank check reader he co-developed temporarily read over 10% of the checks in the United States. He designed stochastic versions of gradient methods. His LeNet model for digit recognition became prototype for all other CNN for image recognition. In 2013, he became the first director of Facebook AI Research. In 2019, he received the Turing Award together with Yoshua Bengio and Geoffrey Hinton.

Fig. 5.14 Yann LeCun * 08.07.1960 Soisy-sous-Montmorency, France. Image credits in Appendix A.3

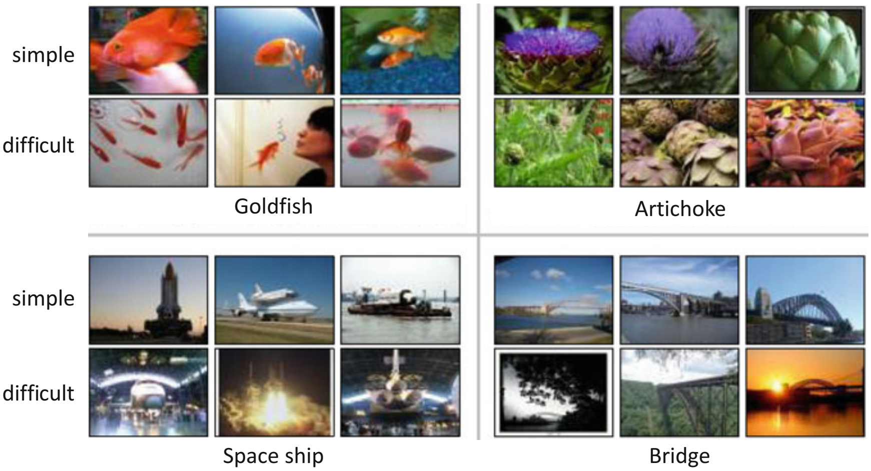


Fig. 5.15 Images in the ImageNet collection are annotated with 1000 different classes. Wu et al. (2018) have graded images of four classes as "easy" or "difficult" according to the assigned class probabilities. Image credits in Appendix A.3

Figure 5.15 shows sample images from four representative classes. Wu et al. (2018) have analyzed the images and classified them as "easy" or "difficult" according to their classification ambiguity. In the difficult images, for example, the objects are very small, have unusual colors, or are in a complex lighting situation. The classification is relatively fine; for example, there are 120 dog categories. Finally, there is a test set of one hundred thousand images unknown to the participants of the competition. For each category there exist at least 500 training images. Thus, the variation within the classes is well covered.

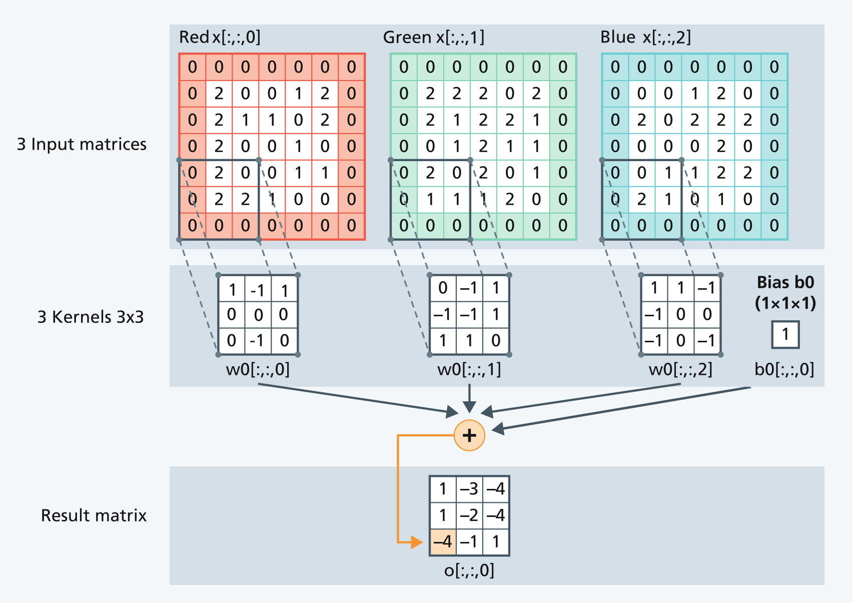


Fig. 5.16 Convolution layer to process the three intensity matrices for red, green, blue of a color image. In this example the kernels are shifted by two positions. The value 0 is assigned to the positions outside the input matrices

For cost reasons, not all objects in an image were classified manually. Therefore, the quality of image classification was evaluated using the top-5 error rate, which measures whether in the test set the annotated "true" object category is one of the top five predicted categories.

The images were reduced to a size of 256 × 256 pixels. Since these are color images, there are three RGB values for each pixel: the red, green and blue values, each of which were combined into a matrix. These three matrices form the input tensor for the image classification.

To relate the three color value matrices to each other, the first convolution layer of the CNN is modified. Three kernels are defined, which are applied separately to the input matrices. The result is summarized and stored in a result matrix. In Fig. 5.16 this calculation is shown. Note that for this model the kernels are shifted by two positions at a time (stride 2). In addition, the value 0 is assigned to the positions outside the input matrices (padding). Usually input matrices and kernels contain real numbers.

5.5 Advanced Convolutional Neural Networks

5.5.1 *AlexNet Successfully Uses GPUs for Training*

The year 2012 brought the breakthrough of DNN in the ImageNet classification competition. The AlexNet (Krizhevsky et al. 2012) uses a total of five CNN layers, three MaxPool layers and three fully connected layers with a total of 60 million parameters. For the first time, the rectified linear unit ReLU was used as an activation function and dropout for effective regularization. The network was trained on two GPUs (graphical processing units). AlexNet was able to reduce the top-5 error rate by 36% to 16.4% compared to the previous year.

The 2013 winners (Zeiler & Fergus 2014) used the AlexNet architecture and modified only some hyperparameters. The 2014 winners (Szegedy et al. 2015) constructed a CNN with 22 layers, called GoogLeNet. They introduced a module with multiple parallel convolutional layers and receptive fields of different sizes (inception module). This module was stacked several times on top of each other in the overall network. The best results of each year are shown in Fig. 5.17 (Li et al. 2018).

5.5.2 *ResNet Facilitates Optimization by Residual Connections*

Adding more convolutional layers to one of these networks did not result in any further improvement. As shown in Fig. 5.18, a CNN with 56 layers had a higher error on both the training and test sets than the CNN with 20 layers. It can be concluded

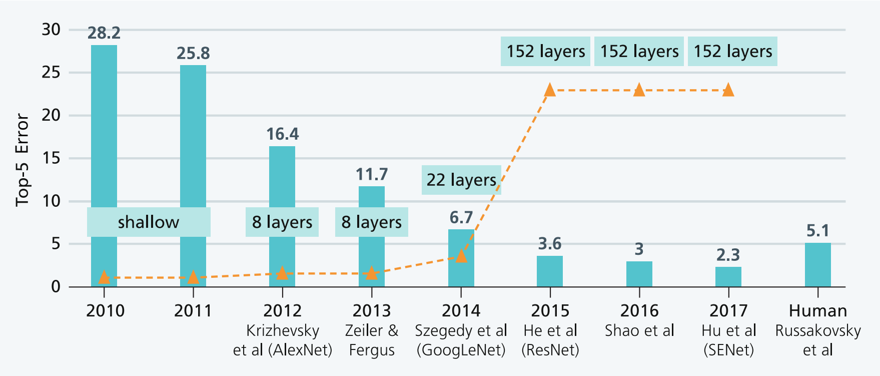


Fig. 5.17 Top-5 classification errors of the ImageNet classification contest. For each year, the winner's publication is given, the short name of the winning network (in parentheses), and the number of layers. On the far right is the classification error of human categorizers. Image credits in Appendix A.3

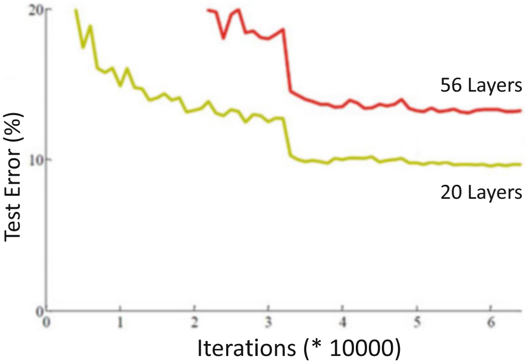


Fig. 5.18 Comparison of test errors on the CIFAR 10 dataset with 10 image classes for larger numbers of convolutional layers (He et al. 2016)

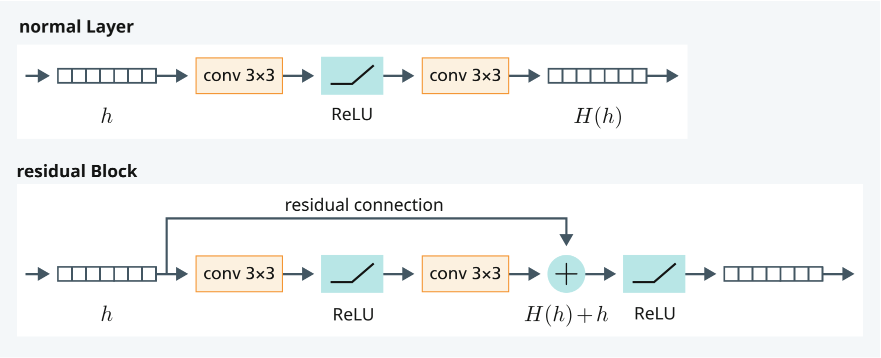


Fig. 5.19 Normal convolution layers (top) and corresponding residual block (bottom). The residual blocks can be stacked very often without disturbing the parameter optimization

that this is not an overfitting problem, but rather that the optimization procedure has difficulty finding the optimum when there are very many layers.

One possible solution is to copy the hidden vectors of the lower layers upward with a so-called residual connection (from Latin residuum “the rest”), so that the model can start with a few “active” layers during optimization and then use the additional layers for improvements as optimization proceeds (Fig. 5.19). Thus, one constructs a shortcut or bypass that allows activations to skip several convolutional layers. If the parameters of a convolution layer are close to zero, the hidden vectors are simply copied up through the bypass and the network has, in effect, fewer layers. This also facilitates the initial optimization. This network structure is called a residual block because in mathematics a remainder is also called a residual.

The winning network of 2015, ResNet (He et al. 2016), followed this scheme. The three color value matrices of the image served as input. Each residual block has two 3 × 3 convolutional layers. Periodically the number of kernels is doubled and

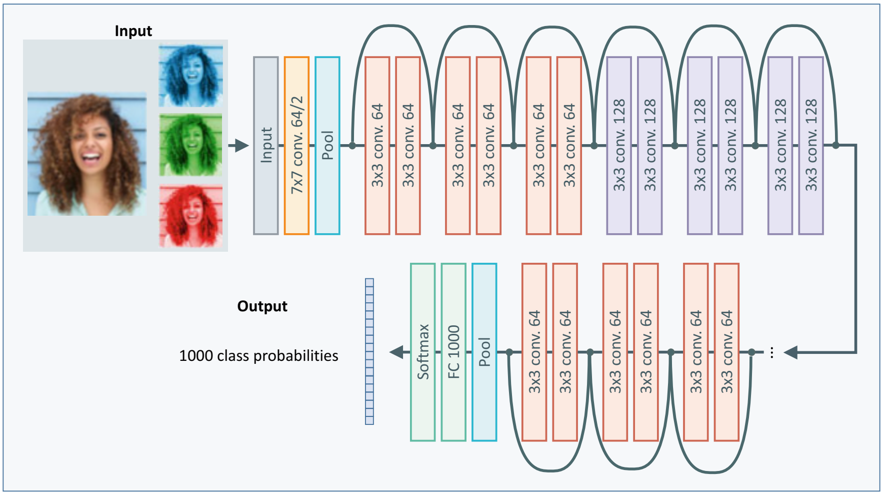


Fig. 5.20 ResNet with up to 152 layers and residual connections to bypass the convolutional layers. The layer structure is denoted as follows: 7 × 7 conv. 64/2 is a convolutional layer with a 7 × 7 receptive field, 64 result matrices and step size 2. Pool is a max-pooling layer, FC 1000 is a fully connected layer with an output vector of length 1000. Image credits in Appendix A.3

the step size of the following kernels is doubled, so that the total size of the result matrices remains the same. Batch normalization (Sect. 4.6.7) and L2-regularization (Sect. 4.6.5) are used to avoid overfitting. Such networks can be trained successfully even with very many layers. With additional layers, higher and higher accuracy on ImageNet was achieved. The ResNet (Fig. 5.20) with 152 layers achieved a top-5 error of 3.6% in the 2015 ImageNet competition. This is better than the 5.1% accuracy of human classifiers. The network has 60.2 million parameters (Zagoruyko & Komodakis 2016). This is about the same number as for AlexNet.

ResNet Requires Enormous Computing Power

The authors used stochastic gradient descent with a minibatch size of 256. The learning rate starts at 0.1 and is divided by 10 when the error stops decreasing. A prediction for one image requires about 10^{10} FLOPS (floating point operations) (He et al. 2016). These 10 billion FLOPS are slightly less than GoogLeNet requires with 22 layers. A total of 90 epochs is trained, each with 1.28 million images, which have been divided into 1000 classes. The training time required is relatively high. To train a ResNet network with 50 layers for 90 epochs on an Nvidia M40 GPU requires 14 days, which corresponds to about 10^{18} FLOPS. This can be greatly reduced by using many GPUs in parallel. Meanwhile, with 2048 GPU chips, (You et al. 2018) manage to train ImageNet in 14 minutes.

5.5.3 DenseNet Employs Additional Residual Connections

In DenseNet (Huang et al. 2017), the pattern of residual connections was further extended. Here, each layer receives inputs from all previous layers (Fig. 5.21). Thereby, the inputs are not added, but concatenated to avoid overlapping of features. However, this linkage no longer works if the feature resolution is changed by Pooling. Therefore, the layers are divided into different dense blocks. Each layer contains as suboperators a batch normalization, a ReLU nonlinearity, and a 3×3 convolutional layer. In total, DenseNet (Huang et al. 2017) has up to 250 individual layers. On ImageNet, the original DenseNet achieved a top-5 error of 6.15%. The authors note that DenseNet requires fewer parameters than ResNet to yield a given error.

5.5.4 Transformed Images Improve ResNeXt Training

Meanwhile, the top-5 error was reduced to 2.0% (Touvron et al. 2019). For this purpose, the ResNeXt-101 network was trained with 101 layers and 829 million parameters. The model uses input images of different pixel sizes. Data augmentation of the training images using various transformations, such as different image parts, horizontal flipping and color variations, proved to be particularly effective in increasing accuracy (Fig. 5.22).

Meanwhile, DNNs for image recognition are no longer constructed manually, but their architecture is determined by automatic methods. A recent development defines therefore a comprehensive differentiable search problem, which can be solved entirely by gradient descent (Noy et al. 2020). This method for hyperparameter optimization requires much less computation time than previous approaches to architectural optimization. Figure 4.33 shows the search time and achieved accuracy for the CIFAR-10 image classification with 10 classes. The classification error can be improved from 2.55% to 1.99% with a computational cost of only 0.2 GPU days.

5.6 Analysis of CNN Results

5.6.1 Individual Kernels Respond To Features of Different Types and Sizes

It is instructive to investigate to which image features the different kernels of the convolutional layers respond. Zeiler and Fergus (2014) first selected kernels of the respective convolutional layers of the CNN. They developed a back-projection technique to find the input patterns that caused the high activity of these kernels. Individual kernels were randomly selected in layers 1–5. Then, for the images of

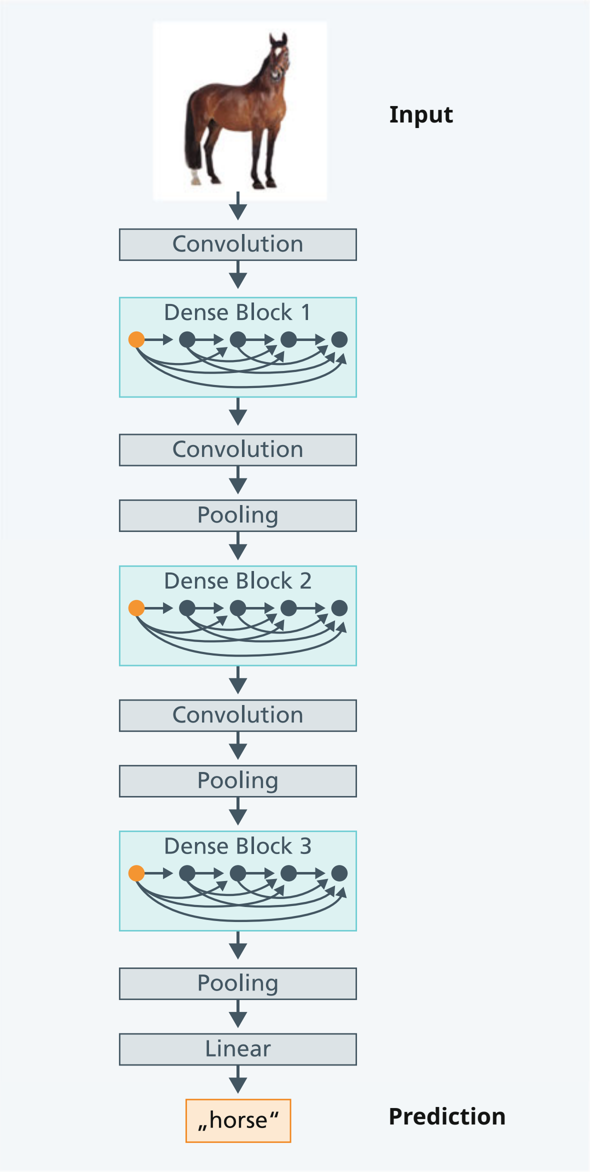


Fig. 5.21 DenseNet with three dense blocks. Within a dense block, each layer passes its output to all subsequent layers (Huang et al. 2017). Image credits in Appendix A.3

the validation set, which was not used in training, the highest output values of each kernel were determined and the corresponding input patterns were constructed by back-projection.

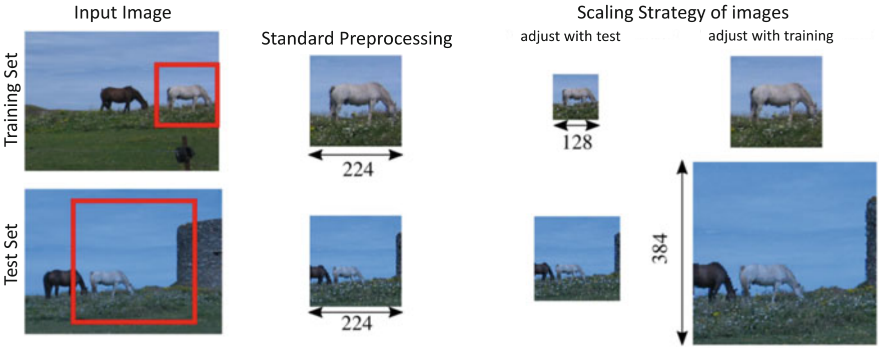


Fig. 5.22 The ResNeXt model uses modifications of the training data to achieve improved accuracies. For example, partial images (outlined in red) are selected and used for training. By default, the sections are normalized to one size. Alternatively, however, reduced or enlarged images are employed as training inputs (Touvron et al. 2019). Image credits in Appendix A.3

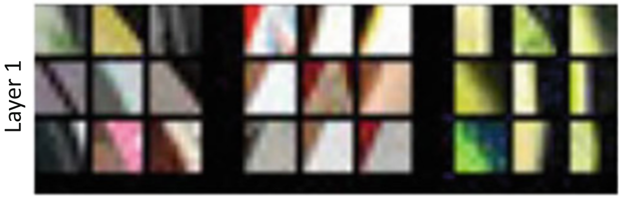


Fig. 5.23 Visualization of individual kernels of a trained CNN in layer 1. For each kernel, back-projected patterns from the validation set are shown, causing high output values of the kernels. Image credits in Appendix A.3

Figures 5.23 and 5.24 contain these back projections for the layers 1–5 of a variant of AlexNet. The complexity of the images increases with the depth of the layer. For layer 1, there are only edges in different directions for different colors and color areas. Layer 2 responds to corners and combinations of edges and colors. Layer 3 contains more complex shapes, such as circular structures and similar textures (small-scale patterns). Layer 4 already contains larger objects, such as dog faces and round volumes. Layer 5 shows complete objects in different perspectives, e.g. keyboard objects, but also grassy areas in the background. Interesting is the higher invariance of features in higher layers as well as the emphasis on discriminative components of the images (e.g. noses and eyes of dogs).

Szegedy et al. (2013) investigated the semantic meaning of individual components of the hidden vectors. It could be conjectured that the components of the hidden vector of the last layer $h = (h_1, \dots, h_k)$ form a unique basis, i.e., the components of this vector are readily interpretable in terms of content. Instead, the authors showed that random projections $q(h) = (q_1, \dots, q_k)$ of h are semantically indistinguishable from the coordinates $(h_1, \dots, h_k)$. Obviously, the individual components h_i or q_j have no special significance. On the contrary, only all components together contain the required information for a classification.

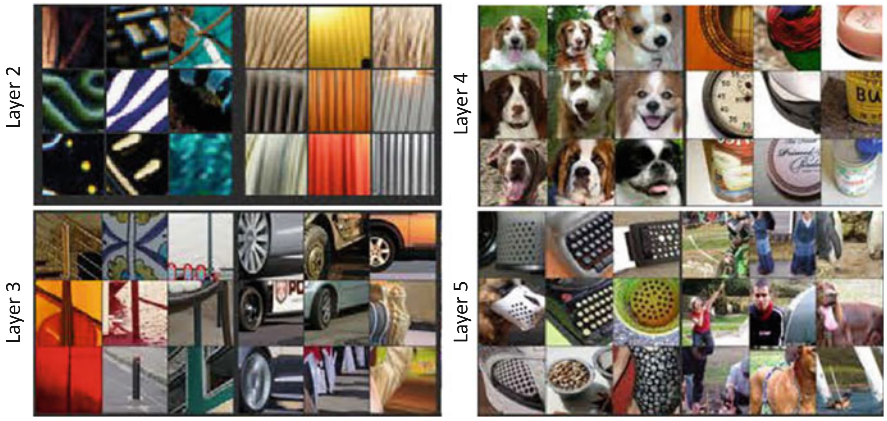


Fig. 5.24 Visualization of individual kernels of a trained CNN in layers 2–5. For each of the kernels, back-projected patterns from the validation set are shown, causing high output values of the kernels. Image credits in Appendix A.3

5.6.2 *Similar Images Correspond To Neighboring Hidden Vectors*

The computed hidden vectors can be considered as points in a high-dimensional space. Between these points, the distance can be calculated to determine closeness of vectors. Krizhevsky et al. (2012) have performed this for the input vector h of the final softmax layer of AlexNet (Sect. 5.5.1). This vector has length 4096 and encodes all features relevant for classification.

Figure 5.25 contains in the left column five images from the ImageNet test set for which the respective vector h was predicted. Columns 2–5 contain images from the ImageNet test set whose h vectors have the smallest distance to the h vectors of the left column. Thus, all these images were not used for training. It can be seen that the h-vectors characterize the images well and that they are suitable for similarity search. There are a number of retrieval methods for images, which are based on these hidden vectors computed by CNNs.

5.7 Transfer Learning Reduces the Need for Training Data

The ImageNet challenge (Deng et al. 2009) includes more than 1.2 million images for the 1000 categories (Sect. 2.1). There are hundreds to thousands of images for each class. Only after training with this large training dataset, DNNs were able to assign classes of image objects with high confidence for the first time. Similarly, also speech recognition or translation models require very large text datasets for

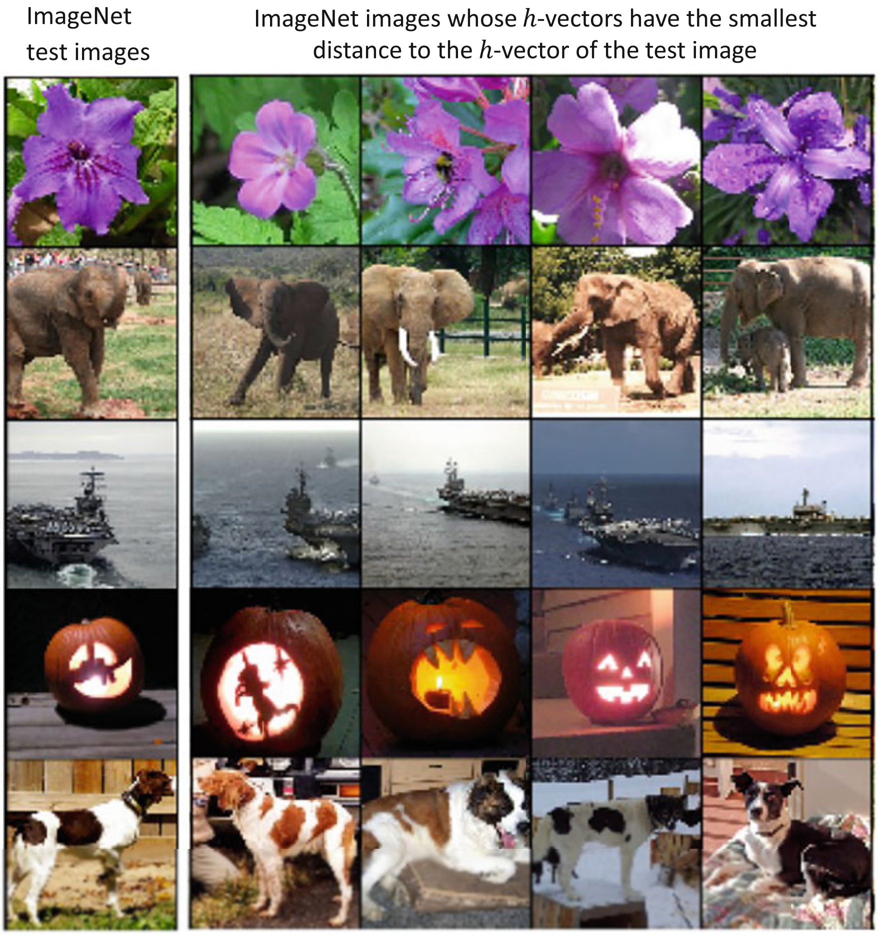

ImageNet test images
ImageNet images whose h-vectors have the smallest distance to the h-vector of the test image

Fig. 5.25 The hidden vector h of the last CNN layer can be used for similarity computation. The left column contains new images with hidden vectors h_i, for which similar images are searched. The right four columns contain found similar images with hidden vectors close to the search vector h_i. Image credits in Appendix A.3

training. An important criticism is (Marcus 2018) that, in contrast, people are able to recognize a new type of object even after seeing only a few examples.

This particular critique mentions a major problem with AI, and in practice is often a strong obstacle to its application. A counter-argument is the observation that humans also need a lot of time and observations during childhood until they can distinguish a large number of objects.

As an answer to this criticism, the concept of transfer learning was developed. First, a model with sufficient training data for a learning problem is trained. Then, the model is further trained for a slightly different problem, ideally requiring only a

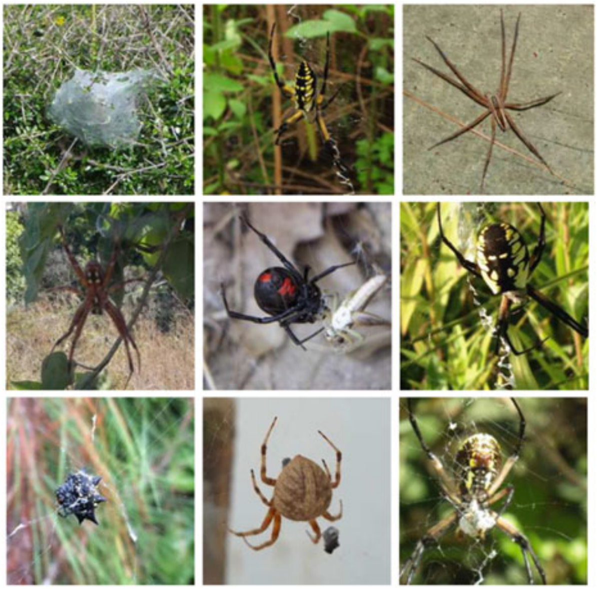


Fig. 5.26 Different training examples in the fine-granular image classification task iNaturals with over 5000 categories (Van Horn et al. 2018). Image credits in Appendix A.3

small amount of additional data. Using a model pre-trained with ImageNet, Cui et al. (2018) were able to achieve a top-5 error of 5.4% on the iNaturals dataset with over 5000 categories from specific domains (plants, birds, etc.) with little training effort (Fig. 5.26). Lee et al. (2020) show that it is also possible to achieve an accuracy of 80% when training with only five training examples of a class. Apparently, transfer learning approaches are suitable in many cases to reduce the number of training examples needed.

For example, it was possible to use transfer learning for the diagnosis of skin cancer. Here, a deep CNN was pre-trained on ImageNet. Through this, the network had already learned many features for image interpretation and classification. It was then adapted on a relatively small training set for classification of possible types of skin cancer (Fig. 5.27). This approach enabled the DNN to classify skin cancer with the same accuracy as experienced dermatologists (Sect. 2.1.1). The use of pre-trained CNNs for specific image recognition tasks is now standard.

It is possible to use semi-supervised learning to improve image classification results. This type of learning employs manually annotated but also non-annotated training examples. For example, Yalniz et al. (2019) train an initial “Teacher Model” on annotated training data (ImageNet). Using this Teacher Model, they classify a very large set (e.g., 1 billion) of unlabeled images. For each class, they select several thousand images that are classified with highest confidence and use them to train a “Student Model”. Finally, they fine-tune the Student Model with the original annotated dataset. For the ImageNet dataset with 1000 classes, Yalniz et al. (2019)

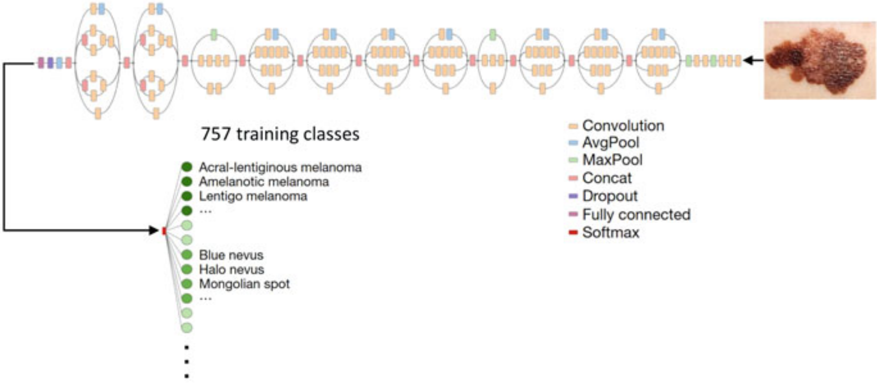


Fig. 5.27 Transfer learning in skin cancer detection. Here, a deep CNN (Google Inception v3) is used, which was pre-trained on ImageNet with 1.3 million images for 1000 classes. It was then trained on 129,450 images of skin abnormalities with 757 output classes. For an input image the probability for each of the classes is estimated (Esteva et al. 2017). Image credits in Appendix A.3

use a dataset of 1 billion images from a social media website that were marked with different hashtags. With their approach the authors are able to improve the top-1 accuracy for several of the best network architectures by more than 3% each.

5.8 Localization of Objects in an Image

5.8.1 Object Localization by Rectangles

Often, people want to know not only whether a car is in an image, but also where the car is in the image. In object localization (Fig. 5.1), the position of an object is indicated with a bounding box. If an image contains several objects, object localization has the task of determining the class of these objects together with their bounding boxes. A well-known application is autonomous driving. In order to be able to navigate safely, autonomous vehicles have to recognize other cars, pedestrians, bicycles, trees, roadside, etc. and localize them on the video image and consequently in the environment.

A method developed for this task is YOLO (Redmon et al. 2016). It divides an image into a regular grid. In Fig. 5.28 this is a 4×4 grid for simplicity, but usually a finer grid is employed. In each grid cell a different object can be detected. If k different object classes are possible, then the position of the center of each object (Bx, By) as well as the width and height (Bw, Bh) of the bounding box relative to the width/height of the grid cell are output. In Fig. 5.29, the center of the car is within grid cell "Grid 7". This allows the method to calculate the exact coordinates of the bounding box (yellow). Additionally, for each class the

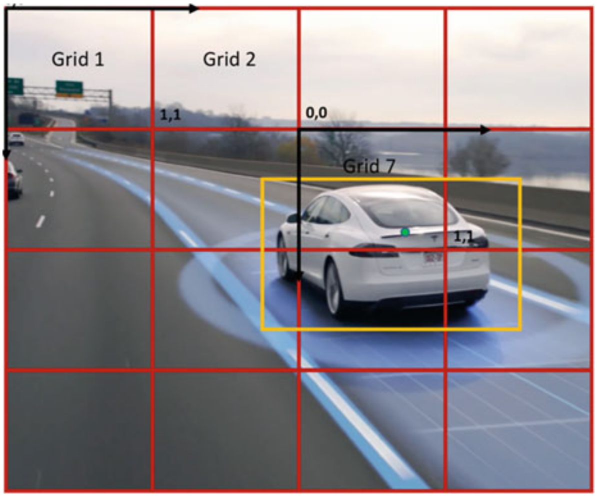


Fig. 5.28 The YOLO method for localizing objects in images divides the image into a grid of cells. If the center of an object is in a grid cell, the position of the center and the width and height of the bounding box are predicted. In addition, the probability for each object class is estimated for each grid cell. Image credits in Appendix A.3

probability is estimated, with which an object of the class is located in the grid cell. Furthermore, a probability is computed whether there is an object in the grid cell at all.

Overall, the method predicts for each cell the class of a contained object together with its bounding box. This is done with a single deep CNN, which generates the outputs for each grid cell in a single prediction step.

The disadvantage of this method is that only one object can be detected in each grid cell. For this purpose, variants have been developed in the meantime, e.g. Yolo9000 (Redmon & Farhadi 2017), which can distinguish 9000 object categories and is still able to annotate videos in real time. Figure 5.29 shows the bounding boxes found by Yolo9000 for two different images.

5.8.2 *Pixel-Precise Localization of Class Objects*

Image segmentation is an image recognition task in which the regions of each object in an image are marked more accurately than with bounding boxes. The goal of image segmentation is to classify each pixel of the image with the class of the object it represents (Fig. 5.30). When objects of individual classes (e.g. different dogs) are assigned to the image pixels, this is referred to as object segmentation. This means, for example, that the pixels of different dogs are displayed differently.

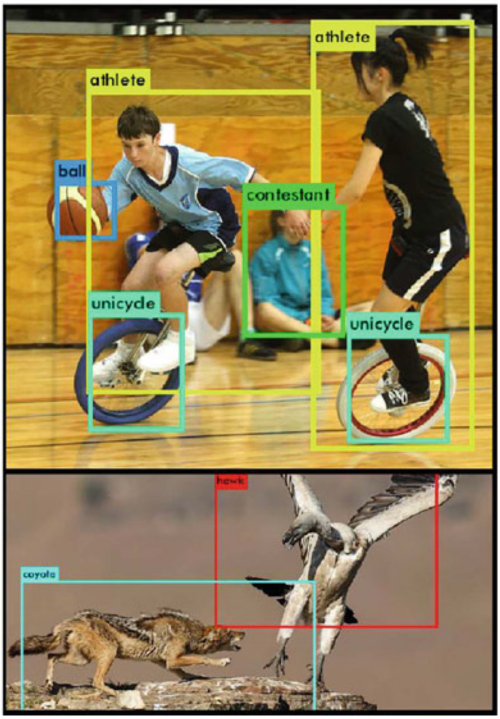


Fig. 5.29 Yolo9000 can create bounding boxes for 9000 different object classes. Image credits in Appendix A.3

In some applications, segmentations are particularly relevant. In case of self-driving cars, for example, it is important to reliably detect the position of roads, different people and different vehicles. In medical diagnostics of X-ray images, the position of individual organs and disease locations in an image must be precisely detected.

A naïve approach to a neural network for image segmentation could employ a model consisting of multiple convolutional layers, eventually computing a probability vector for each input pixel that assigns the probability of classes to each pixel (Li et al. 2017, p. 22). Since this approach requires too much computational effort, efforts are made to reduce the level of spatial detail of the intervening layers.

A common approach to image segmentation consists of an encoder and a decoder. The encoder reduces the spatial resolution of the input (Li et al. 2017, p. 24), e.g., by pooling or by convolutional layers with a step size (stride) larger than 1. In doing so, one can train features that are highly efficient in discriminating between classes. Afterwards, one can increase the resolution again by the decoder and distribute the previously found classifications to the pixels of the input image.

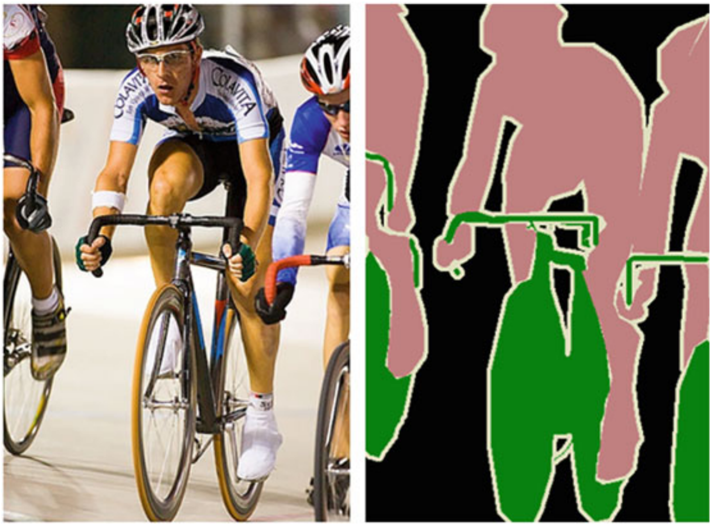


Fig. 5.30 Original image (on the left) and segmented image (on the right) with the object classes person, bike and background. Image credits in Appendix A.3

5.8.3 *Max-Unpooling Assigns Values To an Enlarged Field*

For increasing the resolution (upsampling) different approaches have been developed. In max-unpooling, the effect of a previous max-pooling layer is reversed (see Fig. 5.31). Max-pooling not only extracts the value of the maximum, e.g., for each 2×2 field, but also stores the position of the maximum for each field (Zeiler & Fergus 2014). In max-unpooling, which takes place a few layers further, the result of

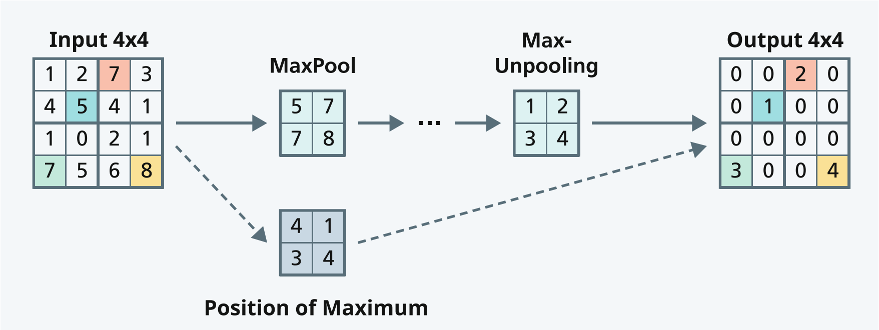


Fig. 5.31 Increasing the resolution of a CNN by max-unpooling. Pooling (on the left) calculates both the maximum and the position of the maximum for receptive fields of size 2×2. In unpooling, the position of the maximum is used to place the unpooling value in the “correct” input position where the maximum previously occurred

the network calculations is transferred to a higher resolution. The calculated value is entered into the position where the maximum previously occurred in the max-pooling layer. Thus, the localization of the features has been transferred from the pooling layer to the unpooling layer and can be used for pixel-precise class labeling (Noh et al. 2015).

5.8.4 U-Net Detects Objects and Then Finds the Associated Pixels

Ronneberger et al. (2015) have developed the U-Net for object segmentation, which first recognizes objects of a class in an image and then identifies the associated pixels. The network consists of a contracting subnetwork, which relates the image parts to each other, and a subsequent symmetric expanding subnetwork, which computes the precise location of the class information.

The U-Net architecture is shown in Fig. 5.32. The input size is 572 × 572 pixels. To this input, multiple 3 × 3 convolutional layers are applied in the contracting subnetwork followed by a rectified linear function (ReLU) and 2 × 2 max-pooling

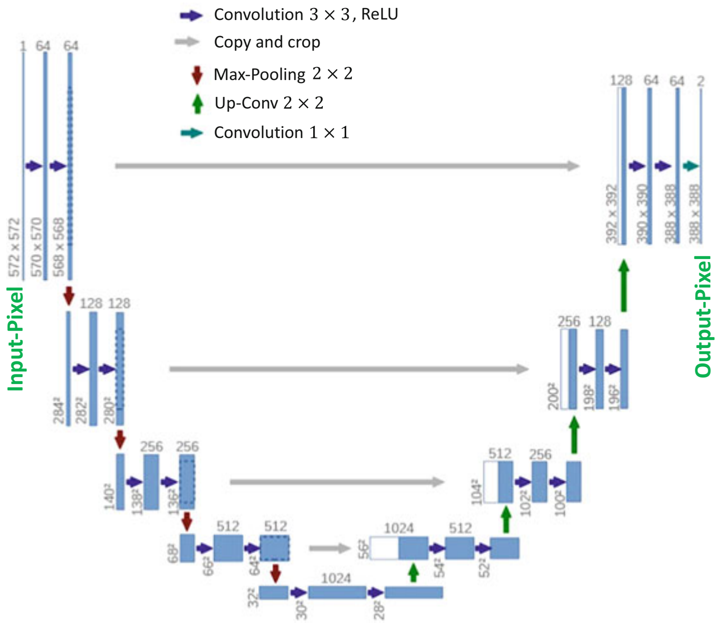


Fig. 5.32 Architecture of the U-Net with the contracting subnetwork (on the left) and the expanding subnetwork (on the right). Each blue box corresponds to a feature matrix. The number of features is above the box, and the resolution is on the left. White boxes represent the copied values from the contracting subnetwork. Image credits in Appendix A.3

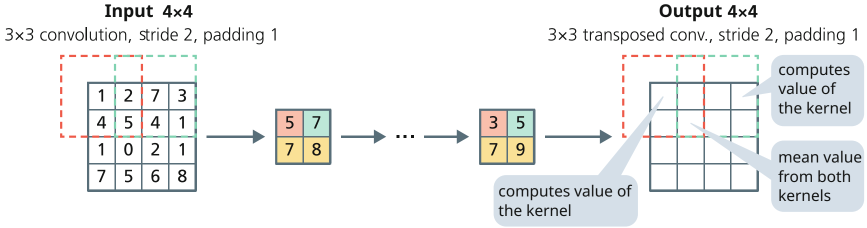

Input 4×4
3×3 convolution, stride 2, padding 1
1 2 7 3
4 5 4 1
1 0 2 1
7 5 6 8
5 7
7 8
…
3 5
7 9
Output 4×4
3×3 transposed conv., stride 2, padding 1
computes value of the kernel
mean value from both kernels
computes value of the kernel

Fig. 5.33 The contracting layer consists of a 3 × 3 convolution layer with step size 2 and padding 1 causing a reduction of the input field (left). The corresponding expanding transposed convolution enlarges the input field and maps each input value to a 3 × 3 field. The parameters of the kernel are learnable and in the overlapping area the output values are summed

with step size 2 to reduce the input size. At each step of the contracting network, the number of features (kernels) is simultaneously doubled. The output of the contracting subnetwork has a field size of 28 × 28 and 1024 features.

The expanding subnetwork consists of transposed convolutions, which increase the field size. For each previous convolution layer of the contracting subnetwork an associated transposed-convolution is defined. For example, the convolution layer has a trainable 3 × 3 kernel with step size 2 and projects one step beyond the input field (padding 1). The kernel is shifted by 2 positions at a time so that the output field is only half the size of the input field. The associated transposed convolution maps each input to a 3 × 3 field and uses a trainable 3 × 3 kernel, as shown in Fig. 5.33. In the areas where the kernels overlap, the computed values are averaged.

From the corresponding layer of the contracting network, the features there are copied and, after appropriate trimming at the edge, appended to the result of the transposed convolution. Two 3 × 3 convolution layers and a rectified linear function (ReLU) follow. This procedure is repeated three times, yielding a 388 × 388 output field with 64 features. Finally, a 1 × 1 convolution is used to weight the features and compute the class output for each pixel.

The network is trained with examples, which consist of the original image and its segmentation, as in Fig. 5.30. U-Net is very popular and has been extended by a number of authors (Jordan 2018). In addition, BatchNorm layers are inserted for regularization (Tutani 2017).

Zhu et al. (2019) apply segmentation techniques to videos of street scenes (Fig. 5.34). They exploit the fact that the labels of pixels between different video frames usually do not change. They predict the next frames and the segmentation associated with them from the previous video frames. Hereby, they can increase the accuracy of segmentation and achieve the best results (83.5% in a special metric for image segmentation) on the Cityscapes dataset.

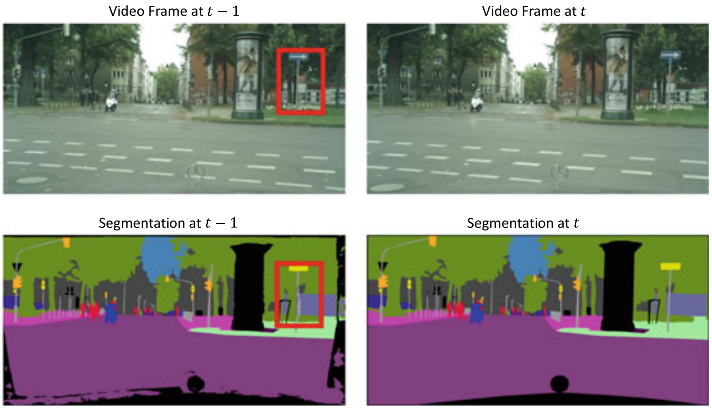


Fig. 5.34 Image segmentation for the Cityscapes dataset for successive video frames. In the top row are the original images and below is the predicted segmentation. Remarkable is the accurate detection and prediction of the motorcycle and the person crossing the street (Zhu et al. 2019). Image credits in Appendix A.3

5.9 3D Reconstruction of a Scene

The goal of Eslami et al. (2018) is to describe pictorial scenes without having to manually annotate images. For this purpose, they define the generative query network, which learns the description of scenes using only sensor input. This DNN uses photos from different viewpoints as input images of a scene and creates an internal representation. This representation is used to predict the appearance of the scene from previously unobserved viewpoints. Thus, it can create representations for scenes without human annotations and generate new looks from them.

In their experiments, Eslami et al. (2018) train a DNN that extrapolates an embedding from input images describing different aspects (left columns in Fig. 5.35). For a new view point, another model component then predicts a scene view (4th column in Fig. 5.35) that matches well with the real view (5th column in Fig. 5.35). The generated predictions respect the laws of perspective, occlusion, illumination and shadow. Only in scene 4 the model cannot predict the green sphere because it is occluded on the input images.

5.10 Human Faces Can Be Matched with High Accuracy

An important application of image classification is face recognition. Figure 5.36 shows the difficulty of recognizing different people. On the one hand, the photos can have very different lighting conditions. On the other hand, intentional facial changes

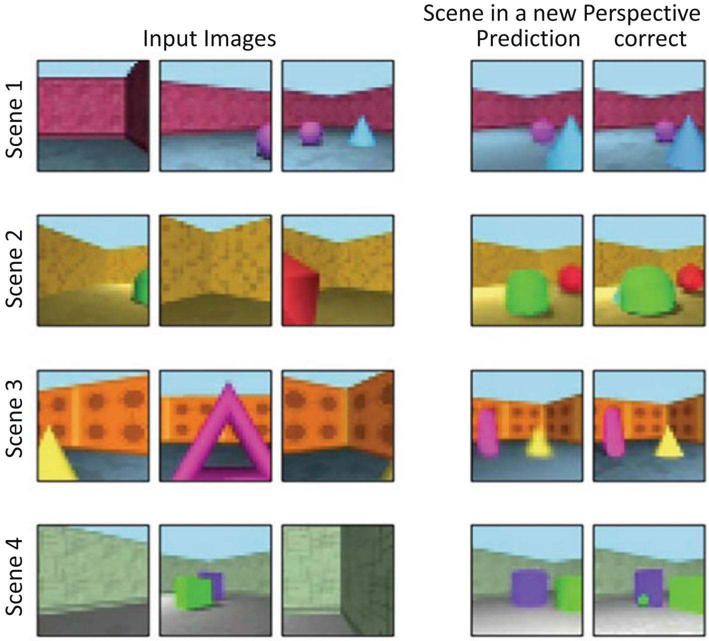


Fig. 5.35 Training images for individual views of the same scene (on the left), predicted image for a new view direction (4th column), and true image for the new view direction. Image credits in Appendix A.3

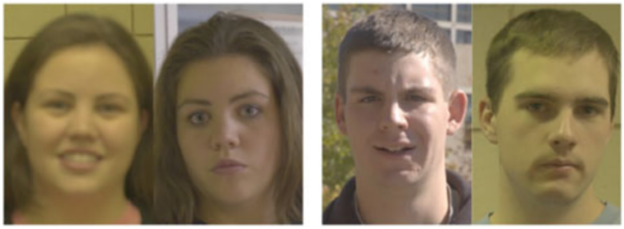


Fig. 5.36 Two examples of face recognition. The pair of images on the left shows the same person, and the pair of images on the right shows different people. Image credits in Appendix A.3

such as makeup, glasses, or hairstyles can make identification difficult. Wang and Deng (2018) give a comprehensive overview of face recognition.

In a first step, faces are initially localized in an image using a robust technique and transformed to the same size (Ranjan et al. 2018). Then, a deep CNN, e.g. ResNet, is trained with this dataset of faces (Fig. 5.37). A well-known training dataset for face recognition is “Labeled Faces in the Wild”. It consists of the faces of famous people, which may be used without privacy problems. Most of the images show the people from the front in a wide variation of lighting conditions. To reduce the effort of collecting photos, different methods of “data augmentation” are used

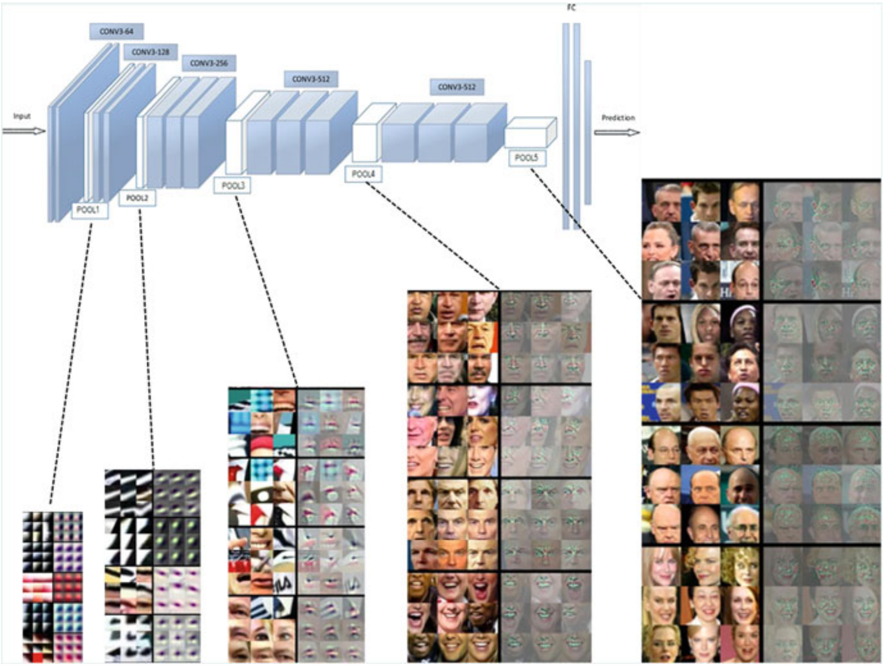


Fig. 5.37 Architecture of a DNN for face recognition. As with ResNet, features of increasing size and complexity are extracted in different layers. In the process, larger face attributes are formed from small-scale lower-level features. The output is a hidden vector representing the features of the face. When applied to an unknown face, its hidden vector is compared to the hidden vectors of other faces in a database, e.g., using Euclidean distance, and the best matching face is selected (Wang & Deng 2018). Image credits in Appendix A.3

to artificially create images in somewhat altered views. One variant of the data set includes 4 million images of 80,000 people.

Similar to Fig. 5.25 faces are represented by the hidden vector of the last CNN layer. A similarity measure is then computed between the hidden vector of the test image and the hidden vectors of the images of the known persons. Phillips et al. (2018) compared the performance of a face recognition method with a hidden vector of length 512 to the recognition performance of a set of human experts. The accuracy of the algorithm (0.96 AUC) was better than 73% of "examiners," which are the best available human experts and achieved an AUC of 0.93. The AUC value is calculated from precision and recall and should be as high as possible. A network based on ResNet with 100 layers achieves an accuracy of 99.83% for the Labeled Faces in the Wild dataset (Wang & Deng 2018, p. 6). That is, out of 1000 individuals from the test dataset, less than 2 are incorrectly matched.

Many potential applications, e.g., searching for people on a wanted list, require high accuracy with a very low false alarm rate, e.g., 10^{-6}. This is still a major challenge even with extremely large training data. Moreover, when applied to nonwhite people, the accuracy of detection usually drops sharply from over 99% to 80–90% (Wang & Deng 2018, p. 14), as these persons are usually underrepresented in the training data. This has recently led to increased public discussions about the reliability of face recognition methods.

5.11 Assessing the Accuracy of Model Predictions

DNNs are used today in many relevant applications where often serious decisions are made, e.g., for the diagnosis of tumors (Fig. 2.2) or in self-driving cars (Fig. 2.22). Here, an additional estimation is urgently needed whether the results of are DNN are correct and to what extent they are uncertain. Standard training techniques only provide a plausible prediction value for an input and do not provide any information about the accuracy of the prediction. Note the class probabilities provided by a classifier are not reliable. The techniques discussed here can be applied not only to image processing, but to any Machine Learning model.

5.11.1 Uncertainty of Model Predictions

There are several types of model forecast uncertainties (Gal 2016):

- Uncertainty due to **lack of training coverage**: A dog race recognition model is being trained with data from different dog races. It is given the task of classifying the image of a cat (Fig. 5.38). The resulting model prediction is uncertain because cats do not appear in the training data. Analogous effects result when parts of the population are represented in the training set with too few examples.

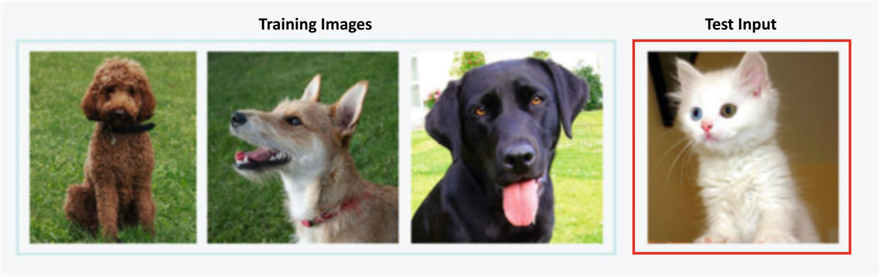


Fig. 5.38 A classification model for dogs was trained with images of dogs (on the left). Classification of a cat image (right) is uncertain because cats are not covered by the training set. Image credits in Appendix A.3

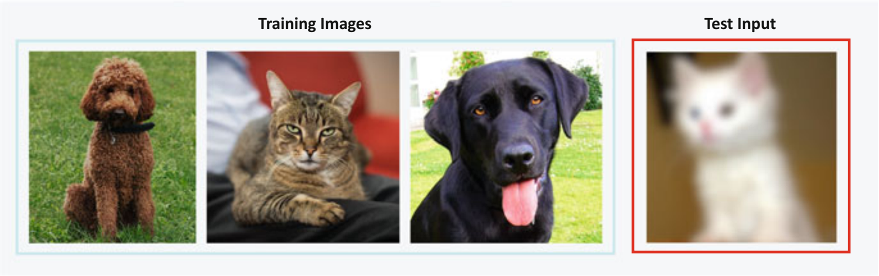


Fig. 5.39 A classification model for images was trained with images of dogs and cats (on the left). Classification of a blurred input image (right) is uncertain. Image credits in Appendix A.3

- Uncertainty due to **measurement errors**: A model is trained to discriminate dogs and cats using images of dogs and cats. It is given the task of classifying a blurred cat image (Fig. 5.39). The resulting model prediction is uncertain.
- **Model uncertainty**: Several models are trained with different numbers of layers and different lengths of hidden vectors (Fig. 5.40). The parameter estimates will also differ because of different initial values and other random influences (minibatches). All models have slightly different values of the loss function, so they are all plausible. The associated model predictions differ, so they are uncertain.

So far, for a particular model, a single "optimal" parameter vector $\hat{w}$ has been determined by using an optimization procedure. Obviously, however, parameter values near $\hat{w}$ are also plausible and could be used for prediction. There are two different approaches to considering neighboring values of the optimal parameter value to determine the extent of these uncertainties.

5.11.2 Bootstrap Generates a Set of Plausible Models

An important cause of model prediction uncertainty is that we never have all possible data available for training, but only a sample of the potential training data. The bootstrap (also called bagging) is an approach to estimate the effect of this training data selection. It uses a certain procedure to draw new random samples from the existing training data and train a set of models. It can be shown that the random samples are also plausible representations of reality and that therefore these models capture the variation in the training data (Hastie et al. 2017, p. 261ff). Running a forecast with each of these models yields a distribution of forecasts that represent model uncertainty.

The method has been tested many times and is quite reliable. It has the side effect that the average of the forecasts of these "ensembles" of models is more accurate than the forecast with a single model. For this reason, an ensemble of

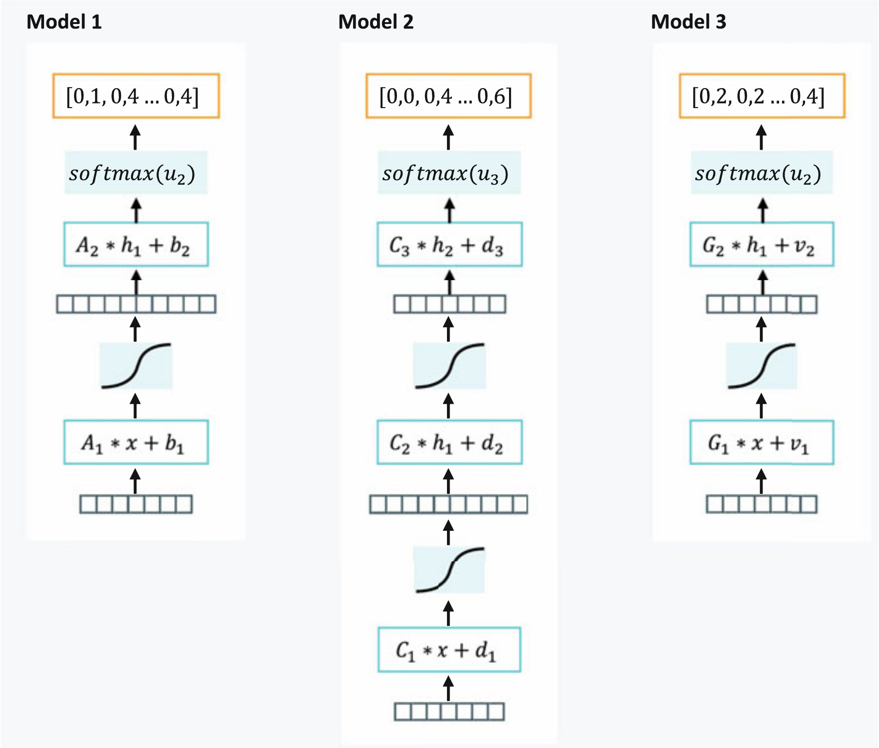


Fig. 5.40 Models with different model structure reach similar values of the loss function, but all can still be plausible. The associated model forecasts are different. Model uncertainty arises because it is not clear which of the models should be used for a forecast

different models often improves reliability (Fig. 5.40) and also allows the estimation of forecast uncertainty. The disadvantage of the method is that one must train a large number ($\geq$30) of models in order to make well-founded statements about the distribution. A bootstrap forecast interval is larger for inputs from areas for which little or no training data are available. This allows the method to estimate the additional uncertainty when only a few training examples are available close to an input.

5.11.3 Bayesian Neural Networks

Bayesian neural networks consider the model parameter w as an additional variable for which a probability distribution $p(w|data)$ can be derived (Goodfellow et al. 2016, p. 132). This posteriori probability can be interpreted as the probability (or density) that the particular parameter is that of the "correct model" that generated

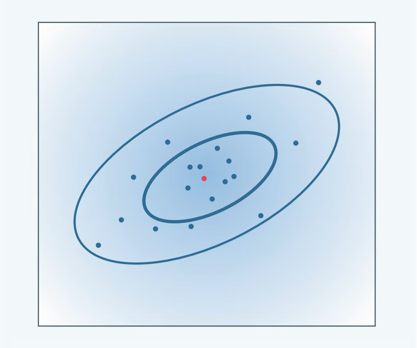


Fig. 5.41 Bayesian a posteriori distribution of model parameters w (points) to represent model uncertainty. Each point is a parameter of a possible model. The red point has the highest probability and is often the result $\hat{w}$ of the optimization

the data. In Fig. 5.41, this distribution is represented by lines of equal probability. The ellipse in the center contains a particularly large number of plausible model parameters. The red dot in the center is the result $\hat{w}$ of the optimization, which usually has the highest probability. The parameters in the neighborhood have a smaller probability, but must also be considered.

In practice, it is impossible to determine this distribution analytically. There are different methods to reconstruct it approximately. Markov-Chain Monte-Carlo methods (MCMC) produce a set of successive parameter vectors $w^{[1]}, w^{[2]}, \ldots$ that traverse this distribution and can be considered as a sample of the distribution. Thus, the result is a set of parameters representing the distribution of plausible parameters. Welling and Teh (2011) present a method that efficiently calculates this distribution using minibatch gradients. From this distribution, model predictions and their uncertainty can be estimated. The authors show that this procedure avoids overfitting.

Gal (2016) shows that a DNN trained with dropout (Sect. 4.6.4) for regularization is approximately a Bayesian neural network. Estimates of uncertainty can be obtained by computing predictions at different dropout masks. This does not cause any additional effort in the optimization. More recently, batch normalization (Sect. 4.6.7) is often used to regularize DNN. Teye et al. (2018) show that training a DNN with batch normalization is equivalent to approximate inference in a Bayesian neural network. Therefore, estimates of the forecast uncertainty can also be obtained for this regularization technique without much computational effort (Fig. 5.42). One can clearly see here that the forecast uncertainty increases sharply outside the area covered by data.

In Fig. 5.43, this technique is applied to a DNN for image segmentation of photographs. In the bottom row, the uncertainty estimates are shown. The uncertainty is highest at the edges of the object regions. The uncertainty estimate is more accurate on the right side because larger minibatches (36 instead of 10 training samples) were used there.

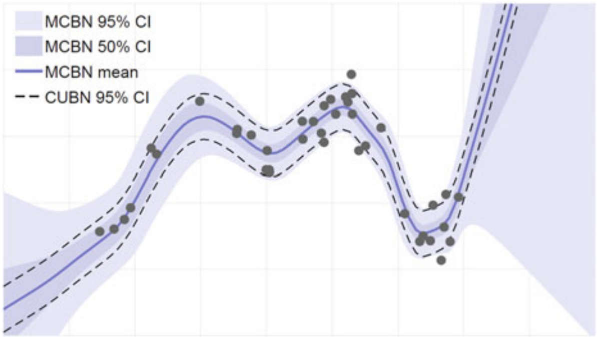


Fig. 5.42 Regression task in which the value of y is predicted from the values of x using a 3-layer model. The observed data are marked by dots. The batch normalization procedure MCBN was used to calculate the mean of the forecast (blue line) and the 50% and 95% forecast intervals (shaded). The dashed lines are forecast intervals of an inadequate technique. Image credits in Appendix A.3

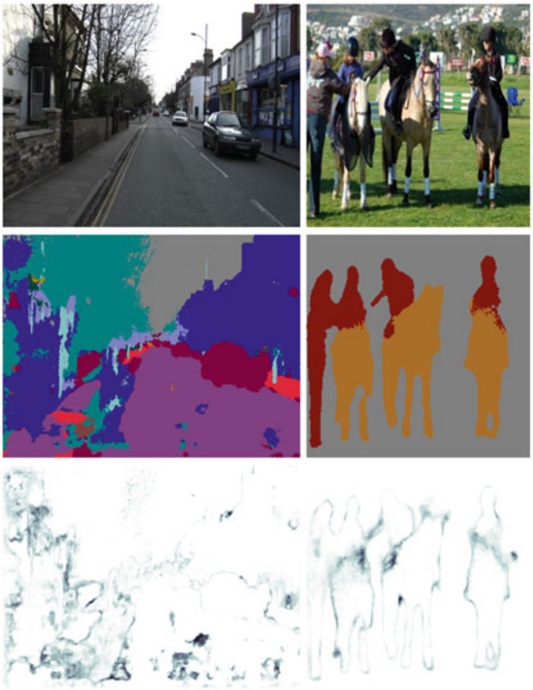


Fig. 5.43 Photos (top) are segmented into object regions by a Bayesian SegNET (middle). The bottom row shows the associated uncertainties in the segmentation. Image credits in Appendix A.3

5.12 Reliability of Image Recognition

5.12.1 Influence of Image Distortions

DNN are very successful in recognizing objects in the available image databases. The networks are optimized for recognition performance. Normally, no attention is paid to robustness of the recognition against modifications of images occurring in everyday life. However, this is extremely important, especially in safety-relevant application areas—e.g., self-driving cars.

In Fig. 5.44, a number of such modifications of the images can be seen, in which a human viewer can still easily recognize the searched object (Hendrycks & Dietterich 2019). Other types of transformation include shifts, rotations, tilts, and different scaling (enlargements and reductions). These were applied to ImageNet and some of them were combined and provided as benchmark data ImageNet-P and ImageNet-C (Hendrycks & Dietterich 2019).

To determine the susceptibility of image classification to such distortions, Hendrycks and Dietterich (2019) trained a number of different types of algorithms on ImageNet. The methods were then applied to the test samples with different distortions and the mean distortion error was computed (Fig. 5.45). AlexNet (Sect. 5.5.1) achieves an error of 43.5% on the unbiased data. This value doubles with the different biases to values around 80%. ResNet-50 (Sect. 5.5.2) has an error of 23.9% on the original data, which increases to values around 65% with

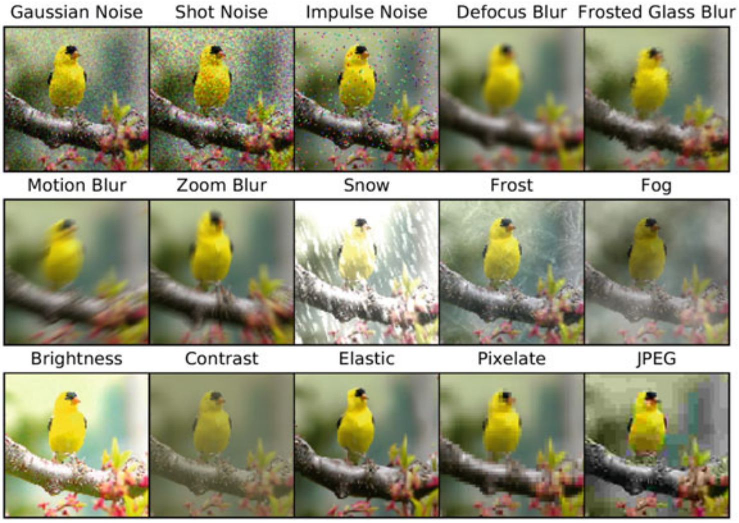
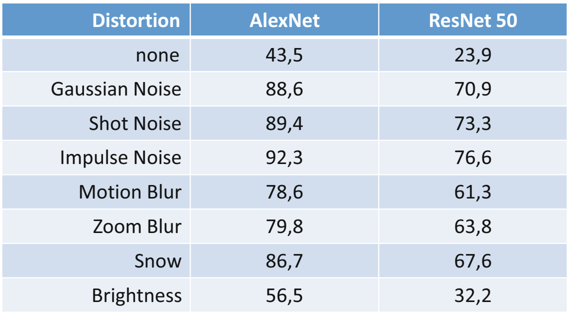


Fig. 5.44 Different algorithmically generated distortions of images in the Image-Net-C dataset. Each distortion has five intensity levels (Hendrycks & Dietterich 2019). Image credits in Appendix A.3



Distortion	AlexNet	ResNet 50
none	43,5	23,9
Gaussian Noise	88,6	70,9
Shot Noise	89,4	73,3
Impulse Noise	92,3	76,6
Motion Blur	78,6	61,3
Zoom Blur	79,8	63,8
Snow	86,7	67,6
Brightness	56,5	32,2

Fig. 5.45 Magnitude of image classification errors for two network architectures under different types of distortions on ImageNet-C benchmark (Hendrycks & Dietterich 2019)

distortions. Although the absolute error with distortions is smaller for ResNet-50 than for AlexNet, the relative increase from 23.9% to 65% is stronger.

This shows that there is still much to be done to make image recognition methods a useful tool for practical applications. One approach to this is to train the DNN on undistorted and additionally on distorted ImageNet data. However, the DNN then continues to have high error rates for unaccounted for types of distortions. Hossain et al. (2018) develop a DNN with the DCT-Net that is relatively robust against many types of distortions, with only small losses in accuracy on the unbiased data. While for the underlying VGG16 DNN the original data error increases from 7.4% to 67.5% with Gaussian noise, the original data error of the DCT-Net increases from 12.5% to 47.0% with Gaussian noise. Nevertheless, image classification degradation due to distortion remains an important problem.

However, one must also consider that when using a fully trained model, outliers can occur in the application data that the model cannot handle yielding unreliable results. Therefore an anomaly- or outlier-detector is required during model application. When using a one-class support vector machine, it is possible to interpret it as a neural network and thus make outlier decision explainable (Kauffmann et al. 2020). One can also use a generative adversarial network (GAN) (Sect. 9.1.2) to train a discriminator that identifies elements that deviate from the training set (Di Mattia et al. 2019). Such methods potentially have higher accuracy because the discriminator can use, for example, discriminative CNNs for images.

5.12.2 Targeted Construction of Misclassified Images

Training of DNN is only possible with sophisticated optimization methods, which can adjust millions of parameters in such a way that the DNN can extract the relevant properties of the training examples. Conversely it is also possible for a fully trained DNN to modify individual images by selectively manipulating the image pixels in such a way that the network assigns the image to the wrong category. The parameters w of the DNN are not changed. Such a procedure is called adversarial attack.

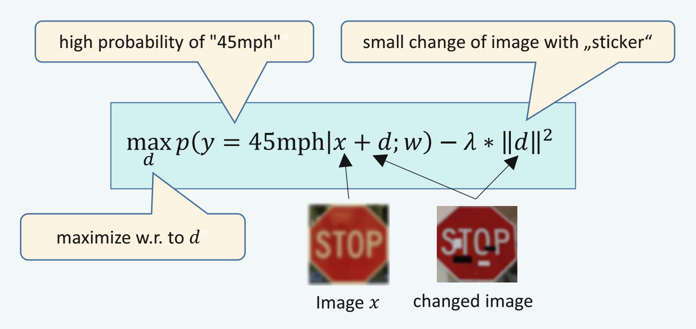

high probability of "45mph"
small change of image with „sticker"
$\max_d p(y = 45\text{mph}|x + d; w) - \lambda * ||d||^2$
maximize w.r.t. to d
STOP
Image x
STOP
changed image

Fig. 5.46 In an adversarial attack, an image x is modified by a small change d. d is chosen in such a way that the probability of the wrong class is high and the absolute values of d are small. λ is a weighting factor. For the traffic sign classification, only modifications to traffic signs were allowed. Image credits in Appendix A.3

Eykholt et al. (2018) applied this approach to traffic sign recognition. They used the LISA-CNN as the image recognition network, which has three convolutional layers and one fully connected layer. It has an accuracy of 91% on the test set. To remain realistic, the authors only allowed changes on the traffic signs themselves, such as stickers or dirt.

The goal was to modify stop signs in a way that they are classified by the image recognition network as a 45 mph speed limit. Figure 5.46 defines this optimization problem. The magnitude of the changes is measured by the area of the rectangles. It is ensured that the changes occur only on the traffic sign and are in the form of stickers.

Figure 5.47 shows some example traffic signs that were all classified by LISA-CNN as 45 mph speed limits. Here, the misclassification was largely independent of the viewpoint of the signs and the distance. Additional experiments also revealed that misclassification consistently occurred when video was recorded from a passing

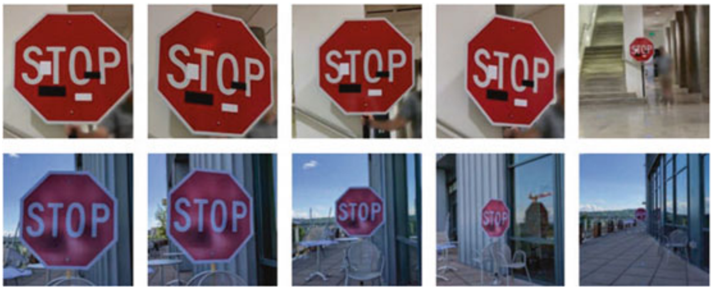


Fig. 5.47 This graph shows stop signs, all classified by LISA-CNN as 45 mph speed limit. In the top row are changes on a real traffic sign due to stickers calculated by the adversarial algorithm. In the bottom row are calculated changes that look like pollution. Image credits in Appendix A.3

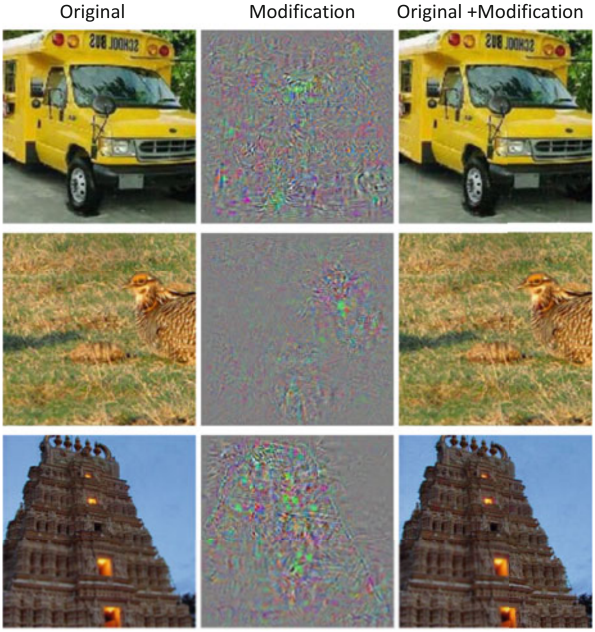


Fig. 5.48 The left column shows the original ImageNet images. The middle column displays the subtle modifications overlaid on the left image weighted . The right column contains sum of the original images and the modifications weighted with a factor < 0.01. All images in the right column were classified as "ostrich" by AlexNet. Image credits in Appendix A.3

car at the modified traffic signs. In contrast, the unmodified stop sign was correctly detected as cars passed by.

Most approaches of this type assume that the attacker has access to the trained classification model and the training data (white box attack). However, there are also successful methods that work without access to the training data and the trained model and even do not know the model architecture (black box attack). Then, a new classification model is first trained on similar data (Xiao et al. 2018). With this approximate model, modifications of the input images are generated, which again lead to misclassifications by the original model with a high success rate.

If all image pixels can be modified (Fig. 5.48), the distortions of the images necessary for an adversarial attack are so small (average 0.0065) that human observers ignore them as "noise" (Szegedy et al. 2013). However, the DNN extracts the relevant features and arrives at the misclassification. Apparently, the DNN often uses very small-scale details of the images for classification, whereas humans rather orient themselves on the contours of an object. Modifications of this kind could also be misused in practice, e.g. for deceiving a facial recognition system for access control.

Apparently, the problem are images where small variations can produce widely different output. Sokolic et al. (2016) present a procedure that reduces the magnitude of output changes for small input changes. Jakubovitz and Giryes (2018) show that this approach greatly increases the robustness of the network and minimally reduces the accuracy of the output.

5.13 Summary and Trends

Image recognition has the task of finding objects in images and interpreting images. In the simplest case, it only determines whether an object of a class (e.g. cat) is present in an image. More advanced methods classify multiple objects in the image and locate the positions of the objects.

The convolution layer used for this purpose was developed according to biological principles. It consists of a small receptive field (e.g. 3×3 pixel), which is linearly transformed with a parameter matrix. This receptive field is moved over the input matrix and computes a value for each position, which is stored in the corresponding feature or result matrix. Many different feature matrices are computed in each convolution layer. Pooling layers reduce the number of features, usually by forming a maximum in a small receptive field. In subsequent convolutional layers, these features are further transformed. Such a network is called a Convolutional Neural Network (CNN).

On the ImageNet dataset, such models have been trained to be good at classifying images by object class. Meanwhile, there are advanced networks with hundreds of layers that are better at classification than humans. The analysis of CNN computations shows that simple features are extracted in the lower layers, which become more complex in higher layers.

The localization of objects in images can be realized either with enclosing rectangles or by marking the pixels of an object. The methods are by now very good in their recognition performance. However, this is also necessary for sophisticated uses, such as self-driving cars. Today, faces can be recognized and assigned with a high level of accuracy. However, there are still too many misidentifications for mass surveillance of the population.

One problem is the sensitivity of the methods to image distortions. They can drastically reduce the recognition performance. Another potential danger are adversarial attacks, in which an intruder tries to manipulate the recognition result. Often, even a small modification of the input image causes the model to assign the image to a different class. However, the attacker usually needs to have access to the model to do this. There is still a great need for improved prevention measures in this area.

Trends

- When answering questions about pictures (Visual Question Answering), textual questions are asked about a picture, which then have to be answered in text form, e.g.: “What is the child sitting on?” It is expected that answering such questions will be the focus of development. This will likely involve an integrated approach using models for text, picture, and language.

(continued)

- Self-supervised learning (Sect. 3.1.2) has led to significant successes in the field of language processing (Sect. 9.2). These models predict the next word of the text from the previous words. It is to be expected that this approach will also be increasingly applied in the image and video domain.
- In industry, AI systems are increasingly used for visual quality control of medical devices, production of pharmaceuticals, food and in the automotive industry.

References

Cui, Y., Song, Y., Sun, C., Howard, A., & Belongie, S. (2018). Large scale fine-grained categorization and domain-specific transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4109–4118).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 248–255). IEEE.

Di Mattia, F., Galeone, P., De Simoni, M., & Ghelfi, E. (2019). A survey on GANs for anomaly detection. ArXiv Prepr. ArXiv190611632. arXiv: 1906.11632.

Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science, 360*(6394), 1204–1210.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115–118.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1625–1634).

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, 364*, 93–202.

Gal, Y. (2016). Uncertainty in deep learning. University of Cambridge 1.3.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* (Vol. 1). MIT Press. https://www.deeplearningbook.org/

Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.), corrected 12th printing. Springer Science & Business Media. https://web.stanford.edu/~hastie/Papers/ESLII.pdf

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Hossain, M. T., Teng, S. W., Zhang, D., Lim, S., & Lu, G. (2018). Distortion robust image classification with deep convolutional neural network based on discrete cosine transform. ArXiv Prepr. ArXiv181105819. arXiv: 1811.05819.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700–4708).

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology, 160*(1), 106.

Jakubovitz, D., & Giryes, R. (2018). Improving DNN robustness to adversarial attacks using Jacobian regularization. In *Proceedings of European Conference on Computer Vision ECCV* (pp. 514–529).

Jordan, J. (2018). *An overview of semantic image segmentation*. https://www.jeremyjordan.me/semantic-segmentation/

Kauffmann, J., Müller, K.-R., & Montavon, G. (2020). Towards explaining anomalies: A deep Taylor decomposition of one-class models. *Pattern Recognition, 101*, 107198.

Keras. (2019). *Keras CNN for MNIST*. https://keras.io/examples/vision/mnist_convnet/

Krämer, T. (2016). *Von Sehstrahlen Und Schwebenden Bildern*. https://www.dasgehirn.info/wahrnehmen/sehen/von-sehstrahlen-und-schwebenden-bildern

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics, 36*(4), 1234–1240.

Li, F.-F., Johnson, J., & Young, S. (2017). *Lecture 11: Detection and segmentation*. Stanford UNIV. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

Li, F.-F., Johnson, J., & Young, S. (2018). *CNNs for visual recognition. Lecture 2: Image classification pipeline*. Stanford UNIV. http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture02.pdf

Marcus, G. (2018). Deep learning: A critical appraisal. ArXiv Prepr. ArXiv180100631. arXiv: 1801.00631

Noh, H., Hong, S., & Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1520–1528).

Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., & Zelnik, L. (2020). Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics* PMLR (pp. 493–503).

Phillips, P. J., Yates, A. N., Hu, Y., Hahn, C. A., Noyes, E., Jackson, K., Cavazos, J. G., Jeckeln, G., Ranjan, R., Sankaranarayanan, S., et al. (2018). Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms. *Proceedings of the National Academy of Sciences, 115*(24), 6171–6176.

Ranjan, R., Sankaranarayanan, S., Bansal, A., Bodla, N., Chen, J.-C., Patel, V. M., Castillo, C. D., & Chellappa, R. (2018). Deep learning for understanding faces: Machines may be just as good, or better, than humans. *IEEE Signal Processing Magazine, 35*(1), 66–83.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7263–7271).

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention* (pp. 234–241). Springer.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision, 115*(3), 211–252.

Sokolic, J., Giryes, R., Sapiro, G., & Rodrigues, M. R. (2016). Margin preservation of deep neural networks. ArXiv Prepr. ArXiv160508254. arXiv: 1605.08254.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. ArXiv Prepr. ArXiv13126199. arXiv: 1312.6199.

Teye, M., Azizpour, H., & Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. ArXiv Prepr. ArXiv180206455. arXiv: 1802.06455.

Touvron, H., Vedaldi, A., Douze, M., & Jégou, H. (2019). Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems* (pp. 8252–8262).

Tutani, G. (2017, October 1). *Practical image segmentation with Unet*. Tuatini's blog. http://tuatini.me/practical-image-segmentation-with-unet/ (visited on 27 February 2022).

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., & Belongie, S. (2018). The iNaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8769–8778).

Visual. (2020). *Visual Cortex — Wikipedia*. https://en.wikipedia.org/wiki/Visual_cortex

Wang, M., & Deng, W. (2018). Deep face recognition: A survey. ArXiv Prepr. ArXiv180406655. arXiv: 1804.06655.

Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of 28th International Conference on Machine Learning ICML-11* (pp. 681–688).

Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., & Feris, R. (2018). Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8817–8826).

Xiao, C., Li, B., Zhu, J.-Y., He, W., Liu, M., & Song, D. (2018). Generating adversarial examples with adversarial networks. ArXiv Prepr. ArXiv180102610. arXiv: 1801.02610.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., & Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. ArXiv Prepr. ArXiv190500546. arXiv: 1905.00546.

You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., & Keutzer, K. (2018). Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing* (pp. 1–10).

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. ArXiv Prepr. ArXiv160507146. arXiv: 1605.07146.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision* (pp. 818–833). Springer.

Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., & Catanzaro, B. (2019). Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8856–8865).

Chapter 6

Capturing the Meaning of Written Text

Abstract The vast majority of information in our society is available as written text. Therefore, this chapter describes the extraction of knowledge from written text. In deep neural networks (DNN), words, sentences and documents are usually represented by embedding vectors. While simple embedding creation methods can only be used to approximate the meaning of words, recurrent neural networks (RNN) have the potential to capture the meaning of a sentence. The best known RNN, Long Short-Term Memory (LSTM), can be used as a language model. It predicts the next word in a sentence and can thus acquire the syntactic and semantic structure of a text. Among other things, it can be used to translate from one language to another. The BERT model calculates the "correlation" between the embeddings of all words of a text and derives context-sensitive embedding vectors that grasp much finer nuances of meaning. It is pre-trained on a large text dataset in an unsupervised manner and then adapted to specific tasks on a small labeled dataset. These models have now been shown to nearly match or exceed human performance for a wide variety of semantic tasks. The transformer model extends this approach to translation and generation of texts and other sequences. Further sections are devoted to the description of images by text and the explanation of predictions of deep neural networks.

Language is the most common medium of communication between humans (Fig. 6.1). Natural Language Processing (NLP) is about developing algorithms that automatically understand and produce natural human language. There is a very wide range of applications, such as search engines for web pages, answering of questions, machine translation from one language to another, recognition of spoken language, or conversational dialog. In this chapter we will look at written language processing, while the next Chap. 7 is devoted to the analysis of spoken language.

Written texts always consists of sequences, e.g. the sequence of words in a sentence or document. Text processing procedures must be able to evaluate such sequences. Another aspect is the number of different words that exist in a language. The "German Dictionary" of the Brothers Grimm contains a vocabulary of at least 320,000 words. In addition, German allows the creative formation of new words,

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_6

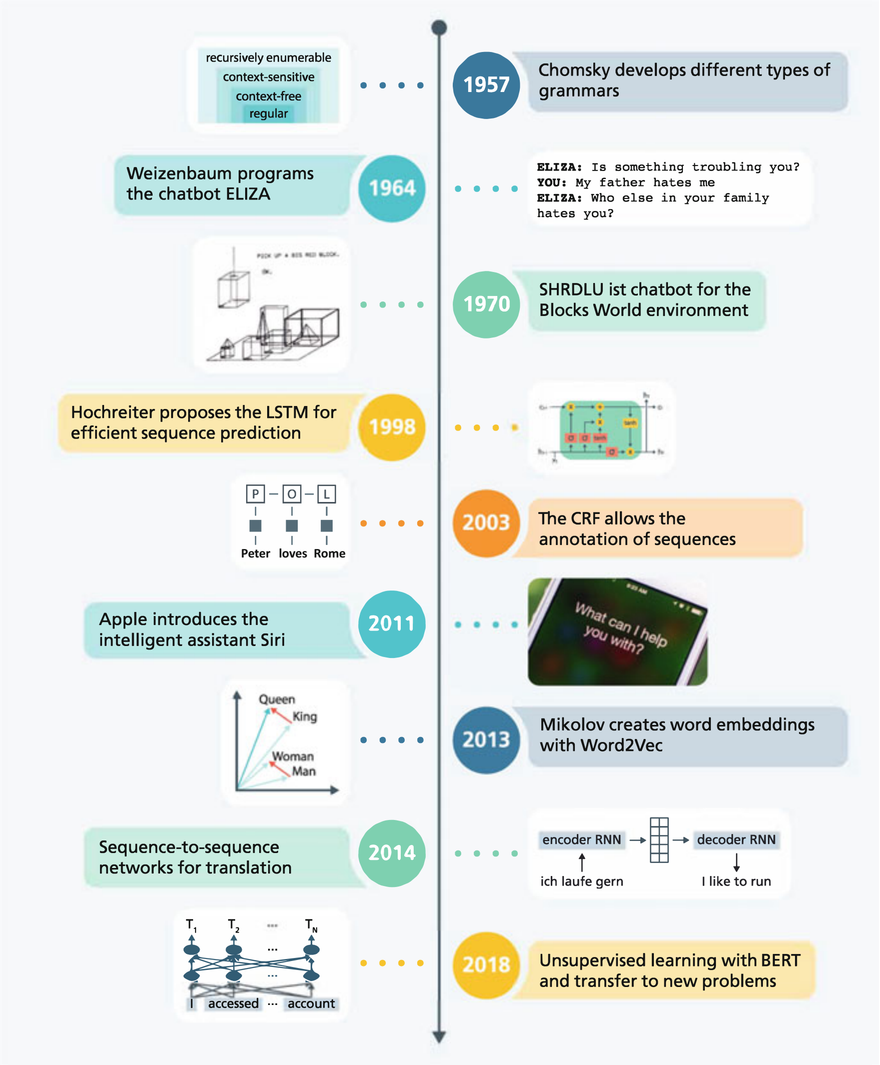
Fig. 6.1 With natural language, people can effortless communicate with each other. A wide variety of media and communication channels can be used. Image credits in Appendix A.3

e.g., as compounds. Taking into account so many features places completely new demands on the analysis methods and Machine Learning techniques used.

The development of automatic processing of texts has a long history (Fig. 6.2). Starting in the middle of the last century, Chomsky developed grammars to capture the formal grammatical structure of language. First chatbots for dialogues on small domains were developed, which used rules. Recurrent neural networks were suggested as special deep neural networks for sequences, and the LSTM developed by Hochreiter and Schmidhuber in 1998 could capture particularly long contexts.

An important idea in language processing was the representation of a word by a vector of numbers, called embedding. Here, words with similar meanings should also have similar embeddings. Mikolov et al. (2013) presented a simple method for computing embeddings. With the advent of graphics processing units (GPU), more and more complex problems could be solved with LSTMs, e.g., automatic translation in 2014. In this context, the attention mechanism was developed, which evaluated the “correlation” between words.

Recently, natural language processing approaches have been developed that learn the internal structure of language from large amounts of non-annotated text. They acquired a profound knowledge of grammar and contextual relations, including a limited knowledge of the world. These models, e.g., BERT, can be used for a number of tasks without specifically annotated training data. We will present these models in Sect. 6.6. The Transformer extends this idea to the translation of a text into another language (Sect. 6.8).


recursively enumerable
context-sensitive
context-free
regular
1957
Chomsky develops different types of grammars
Weizenbaum programs the chatbot ELIZA
1964
1970
SHRDLU ist chatbot for the Blocks World environment
Hochreiter proposes the LSTM for efficient sequence prediction
1998
P
O
L
Peter
loves
Rome
2003
The CRF allows the annotation of sequences
Apple introduces the intelligent assistant Siri
2011
What can I help you with?
Queen
King
Woman
Man
2013
Mikolov creates word embeddings with Word2Vec
Sequence-to-sequence networks for translation
2014
encoder RNN
decoder RNN
ich laufe gern
I like to run
T_1
T_2
...
T_N
I
accessed
...
account
2018
Unsupervised learning with BERT and transfer to new problems

```
ELIZA: Is something troubling you?
YOU: My father hates me
ELIZA: Who else in your family
hates you?
```

Fig. 6.2 The history of automatic processing of written natural language. Image credits in Appendix A.3

6.1 How to Represent the Meaning of Words by Vectors?

A symbol is a written or pictorial sign for an idea, an object, a process, etc. Classical natural language processing was based on the use of symbols for words and relations:

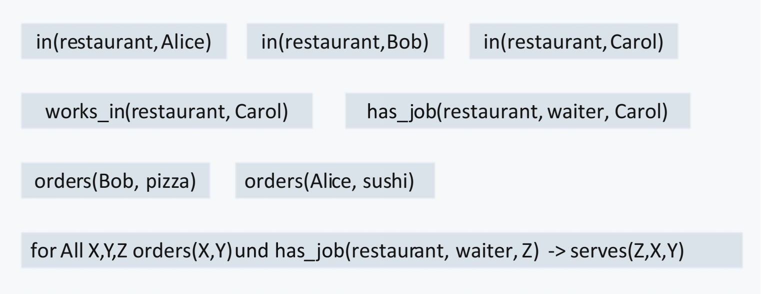

in(restaurant, Alice)
in(restaurant, Bob)
in(restaurant, Carol)
works_in(restaurant, Carol)
has_job(restaurant, waiter, Carol)
orders(Bob, pizza)
orders(Alice, sushi)
for All X,Y,Z orders(X,Y) und has_job(restaurant, waiter, Z) -> serves(Z,X,Y)

Fig. 6.3 In a symbolic knowledge base, knowledge is characterized by word symbols and these are connected by relations

- Knowledge is explicitly represented by symbols for words (concepts) and relations. A relation establishes a relationship between concepts. For example, some aspects of a restaurant can be described by the word and relation symbols in Fig. 6.3. The last statement describes the rule: if person X orders a dish Y and Z is a waitress in the restaurant, then Z serves the dish Y for person X.
- Results are computed by comparing word symbols and by linking rules to form a conclusion with the help of logic. For example, from the "knowledge base" in Fig. 6.3, one can conclude: *"serves(Carol, Bob, Pizza)"* and *"serves(Carol, Alice, Sushi)"*.
- Language processing aimed at transforming natural language texts into knowledge bases, and then use logic to draw inferences from them.

However, a fundamental problem with this approach is that the actual meaning of the words (e.g., *"in"*, *"works_in"*, *"restaurant"*, *"carol"*, ...) in the real world is not known. They are simply symbols that do not capture the richness of natural language use. Rather, it is necessary that the word symbols must be connected to concepts in the real world by "symbol grounding". Thus, for example, one must relate the symbol *"restaurant"* to real restaurants and their typical characteristics. It is therefore necessary to connect the meaning of alternative formulations in terms of content.

When working with knowledge bases as in Fig. 6.3, it turned out that one needs a lot of facts and rules (so-called world knowledge) to get a useful description of real application domains. For example, one has to consider that Carol only works within her shift in the restaurant. On the other hand, it turned out that additional rules and facts often lead to internal contradictions in the knowledge base. Therefore, the use of symbolic knowledge bases was not very successful.

In order to better describe the meaning of words, a new way of processing such information has been established in recent years:

- Knowledge (words, sentences) is implicitly represented by large vectors of real numbers. Words with similar meaning are mapped to vectors that have a small

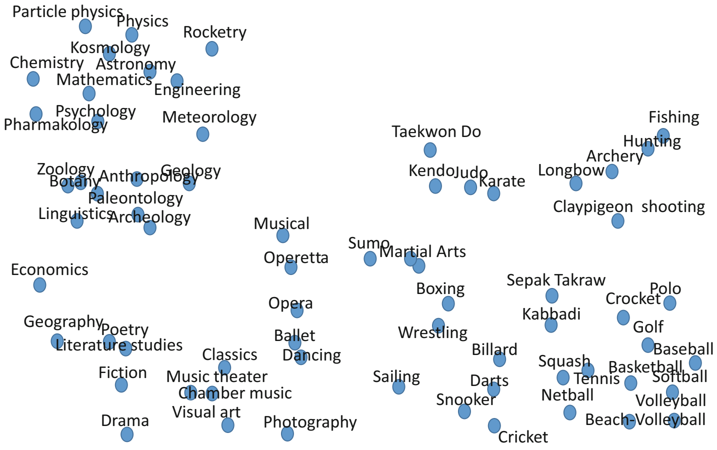


Fig. 6.4 Words can be represented by vectors, where the vectors of words with similar meaning have a small vector distance. This figure contains a projection of high-dimensional embedding vectors for words onto vectors of length 2 corresponding to a point in the 2-dimensional plane

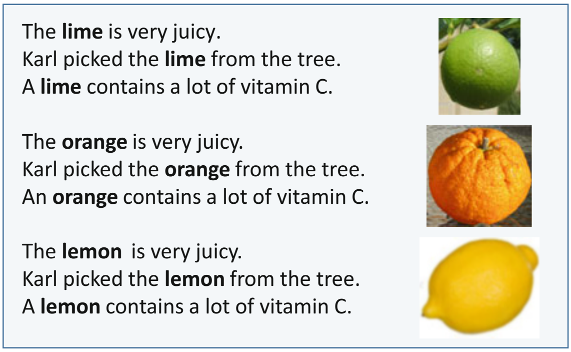
vector distance in the mathematical sense. The semantic content of words and sentences thus becomes comparable.

- These meaning vectors are derived by deep neural networks using the vectors as inputs or outputs. They are based on the idea that words with similar meaning mostly appear in similar textual environments.
- The necessary models can be trained with simple optimization methods, but the emergence of a result is often difficult to explain.

In Fig. 6.4 such high-dimensional representations of words have been projected onto two dimensions in such a way that the proximity of vectors is almost preserved. Only the proximity and distance of the terms are relevant, the vector components themselves have no direct meaning. On the right you can see sports terms, types of performing arts in the middle, and science terms on the left. If two terms are positioned close to each other, they are very similar in their meanings.

6.1.1 The Concept of Embedding Vectors

The spelling of words says almost nothing about their meaning. The words “couch” and “sofa” have completely different spellings, but express the same concept. Such words with the same or very similar meaning are called synonyms. Since natural language processing strives to capture the meaning of words, it seeks a representation for words that is nearly the same for similar words and different for semantically distinct words.



The **lime** is very juicy.
Karl picked the **lime** from the tree.
A **lime** contains a lot of vitamin C.

The **orange** is very juicy.
Karl picked the **orange** from the tree.
An **orange** contains a lot of vitamin C.

The **lemon** is very juicy.
Karl picked the **lemon** from the tree.
A **lemon** contains a lot of vitamin C.

Fig. 6.5 The terms lime, orange, and lemon can be interchanged in these three sentences. This is a clue to their semantic similarity. Image credits in Appendix A.3

One approach to the automatic determination of such a representation was developed long ago by Firth (1957). He stated that *"a word is characterized by the company it keeps"*. This means that the sense of a word is determined by its neighboring words. Figure 6.5 contains exemplary sentences in which the words lime, orange and lemon can be exchanged. This is an indication that these words are semantically similar.

We have seen in Sect. 4.3.1 that deep neural networks transform the input features into new vectors or tensors that form a more suitable representation of the inputs for the learning task. Therefore, it is obvious to try this with words as well, i.e., to use a vector as a representation of a word.

In the following, each word v is represented by a vector $emb(v)$ of real numbers, the embedding vector or embedding for short. The dimension k of the vector is chosen beforehand, e.g. $k = 100$. The goal is to specify for each word v the values of the corresponding vector $emb(v)$ in such a way that the vectors of similar terms are close to each other in the k-dimensional space . These embeddings are also called static embeddings, because a single unique embedding is assigned to each word.

In Fig. 6.6 this is symbolized in three dimensions for the terms *"Orange"* and *"Lemon"*. The term *"Apple"* is also a type of fruit, but different from *"Orange"*. The distance between the embeddings of *"Orange"* and *"Lemon"* is smaller than the distance between the embeddings of *"Orange"* and *"Apple"*. The direction of the connecting line between two terms, e.g. *"Orange"* and *"Orange_tree"*, can be interpreted in terms of content as a "direction of similarity" and symbolizes the relation between orange and its tree. Even further away are the embeddings of very different terms such as *"Car"*.

Overall, embeddings are a way to encode words so that they can be processed in DNN. All of the following neural models follow this pattern. Yoshua Bengio has laid important foundations for the derivation of embeddings by neural networks (Fig. 6.8).

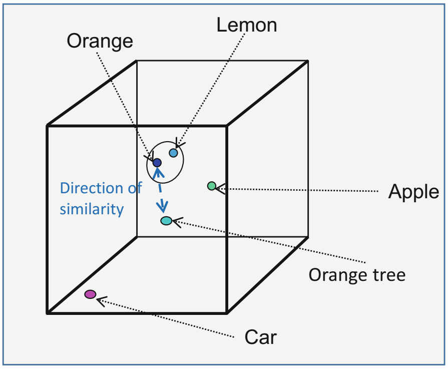


Fig. 6.6 Three-dimensional representation of the embeddings of words. Words that are similar in meaning are close together, while others are further apart, depending on the degree of similarity in meaning. The direction of the line connecting two embeddings says something about the relationship of the words and is a "direction of similarity"

6.1.2 Computation of Embedding Vectors with Word2vec

Word2Vec is an embedding network, a simple but very efficient approach to determine embeddings. The method uses a large training dataset of texts. For each word v in the vocabulary V of this training data, a separate embedding vector $emb(v)$ of length k is defined, which initially is filled with random real numbers between -1.0 and 1.0. Usually, an embedding length k between 100 and 1000 is chosen.

As a preparation, the set of words of a text collection, the vocabulary, must be determined. For this purpose, a list of the different words in the text is formed and sorted according to their frequency, counting words with different endings as different. The vocabulary can cover several hundred thousand words, with very many words being rare, i.e. appearing only once or twice. This observation is true for almost all languages. An embedding vector can only be determined reliably if the corresponding word occurs several times. Therefore, rare words with less than 5 occurrences are omitted from the vocabulary and replaces them in the text with a special placeholder, e.g. *"UNK"*. Nevertheless, a text collection will still contain many different words, e.g. $N = 50,000$.

According to Firth (1957), neighboring words are relevant for the meaning of a word. For the example sentence *"Karl loves sweet and fruity oranges"*, the 2-neighborhood of the word "sweet" consists of the words *"Karl"*, *"loves"*, *"and"*, and *"fruity"* (Fig. 6.7). Obviously, these words characterize the word orange in some way, e.g., *"fruity"* suggests that *"sweet"* describes the taste of a fruit.

The Word2vec model now predicts the probabilities of words $v^{(t+i)}$ in the neighborhood of the central word $v^{(t)}$ (e.g., *"sweet"*) by using a logistic regression model (Sect. 3.4):

$$p(v^{(t+i)}|v^{(t)}) = \text{softmax}(A * emb(v^{(t)}) + b)$$

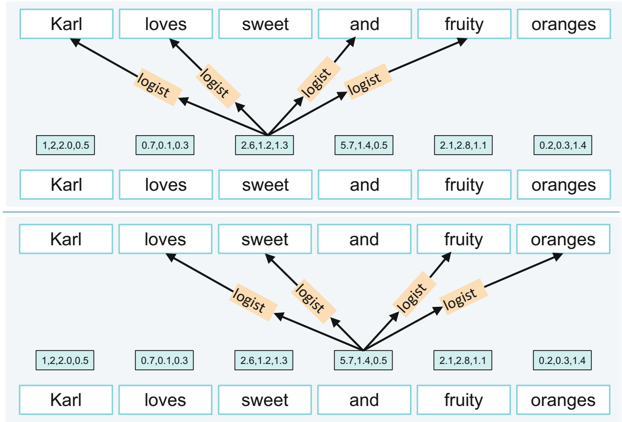


Fig. 6.7 The logistic regression model is applied to the embedding vector of the central word $v^{(t)}$ = "*sweet*" (top). This is used to predict the probabilities of the words in the neighborhood of "*sweet*". In the next step (bottom) the prediction of the neighboring words for the next word $v^{(t+1)}$ = "*and*" is performed

Yoshua Bengio studied computer science and was a PostDoc at MIT under Michael Jordan. In the 1990s, he combined DNN with probabilistic models that have subsequently been used for handwriting recognition. In 1993, he moved to the University of Montreal. He then suggested high-dimensional word embeddings, which are in use everywhere today. He was involved in the development of attention mechanisms, which play a major role in language translation. He was also engaged in the development of generative adversarial networks (GAN), which can be used, for example, to generate novel images from a set of training images. He assumes that in the future AI systems will be needed that can recognize causes and effects. In 2019, he received the Turing Prize, the most prestigious scientific award in computer science.

Fig. 6.8 Yoshua Bengio, *1964 in Paris, has been a major contributor to the development of neural networks and object representation. Image credits in Appendix A.3

Here $v^{(t)}$ denotes the word at position t. If one chooses $N = 50{,}000$ as vocabulary size, one obtains a classification model with a huge number of 50,000 classes, and the model calculates a probability for each possible neighbor word (Fig. 6.8).

Typically, one uses neighborhoods of size 5 to 10. This model contains as unknown parameters the matrix A and the vector b and for each word v the embedding vector $emb(v)$. The parameters A and b are the same for the prediction of all neighborhood words. The dimensions of these parameters are for $emb(v)$: $k \times N$, for $A : N \times k$, and for $b : N$. Choosing $k = 100$, the model has a total of more than 10 million free parameters, so it can potentially represent many properties of words. As in Sect. 3.5.2, the cost function requires that the probabilities of the predicted neighbor words become as high as possible (Mikolov et al. 2013).

Word2vec is an example of self-supervised learning (Sect. 3.1.2). Here, one component of the observed data is predicted from the remaining components. No explicit annotations are needed. The model ignores the exact order of the words, only the neighborhood of the words is considered. Nevertheless, the model can extract a lot of information about the inner structure of a text and use it for the intended applications. This principle is later used for many applications in language processing.

6.1.3 Softmax Function Approximation Reduces Computation

The parameters of the model are optimized as usual with the help of the stochastic gradient descent (SGD). The parameters are changed so that the probability of words appearing in the vicinity of the central word becomes as high as possible. Note that usually many different words are observed in the neighborhood, so the probabilities cannot be high. However, the probability of words never appearing in the neighborhood of a central word should become very small.

There is a difficulty, which is caused by the high number of alternative words. The affine transformation of the logistic model

$$u^{(t)} = A * emb\left(v^{(t)}\right) + b$$

generates a value $u^{(t)}$ for **each** of the N, e.g. $N = 50{,}000$, possible words. For example, to compute the probability of the word $v^{(t-2)} =$ *“Karl”* in the neighborhood by the softmax function, $N = 50{,}000$ terms must be computed and added in the denominator of the fraction, which requires a very large computational effort. As an alternative, one can calculate an approximate value for the softmax function, for which first a sample S of a few (5 to 20) words is randomly selected, and the sum over the associated terms in the denominator is determined.

$$\frac{\overset{\text{Index of Karl}}{\exp\left(u^{(t)}_{i(Karl)}\right)}}{\underbrace{\exp\left(u^{(t)}_{1}\right)+\cdots+\exp\left(u^{(t)}_{50000}\right)}_{\text{Sum of scores for all words}}} \approx \frac{\exp\left(u^{(t)}_{i(Karl)}\right)}{\exp\left(u^{(t)}_{i(Karl)}\right)+\underbrace{\sum\nolimits_{m\in S}\exp\left(u^{(t)}_{m}\right)}_{\text{Sum of scores for a random sample of all words}}}$$

This procedure is called “Noise Contrastive Estimation” Dyer (2014) and is commonly used for approximating the softmax function for many classes. The computational cost is orders of magnitude lower. The additional inaccuracy in the gradient determination is not severe, since a stochastic gradient descent is used for the optimization (Sect. 3.6.6), which itself already yields inaccurate gradients.

6.2 Properties of Embeddings Vectors

We tested the Word2vec method for the German Wikipedia. This currently comprises about 2.2 million articles with 971 million words and requires 6.2 GB of memory. A vocabulary of about 2 million words was formed and the length of the embeddings was set to 100. Training the model required about 6 hours. Importantly, the method does not use any default values and works for arbitrary languages and alphabets.

As a result of the training, an embedding vector is obtained for each word of the vocabulary. The components of this vector have no meaning per se, but are to be interpreted only in comparison with the embeddings of other words.

6.2.1 Nearest Neighbors of Embeddings Have Similar Meanings

As a first evaluation, one can take the associated embedding for a word and find the nearest neighbors in the 100-dimensional space. As distance measure one chooses the cosine distance, which calculates the angle between two vectors $u = (u_1, \ldots, u_k)$ and $s = (s_1, \ldots, s_k)$. For this, both vectors are first normalized to length 1 (i.e. divided by their Euclidean length). Then, the scalar product

$$d(u, s) = \sum_i \tilde{u}_i * \tilde{s}_i$$

of the normalized vectors $\tilde{u}$ and $\tilde{s}$ is the angle between the vectors u and s. The possibly different lengths of the vectors do not matter because of the normalization.

Figure 6.9 shows the nearest neighbors of the words in the first line in descending order. According to this, for example, the embedding of "trumps" has the smallest distance to the embedding of "trump". This states that the words "trumps" and "trump" occur in very similar environments. Therefore, the embedding of former president "obama" also has a small distance to the embedding of "trump". Note that "mccain" is also close, although McCain was never president. Thus, the similarity of the words does not always match the usual terminology.

Figure 6.10 was created using t-SNE (van der Maaten & Hinton 2008), a method for visualizing high-dimensional data. It maps each high-dimensional point into a point in two-dimensional space, so that points adjacent in high-dimensional space are also close to each other in low-dimensional space. Correspondingly, points

crime	**explosion**	**trump**	**stupid**	**iran**	**evacuation**
violence	accident	trumps	dumb	irans	bombardment
suicide	eruption	obama	nasty	irak	interdiction
homicide	wreckage	barack	ignorant	iranis	invasion
burglary	outbreak	obamas	fat	teheran	extermination
police	barrage	reagan	ugly	afghanistan	bombing
corruption	fog	mccain	ratchet	turkmenistan	encirclement
theft	explosions	bush	dirty	pakistan	replenishment
narcotics	iceberg	clinton	simple	iranshar	inspection
fraud	earthquake	clintons	lame	iranische	redeployment

Fig. 6.9 For the embedding vector of the words in the top row, the closest embedding vectors were calculated and the corresponding words were printed. The embedding vectors of the words listed in the lower rows are further away, so these words are more dissimilar

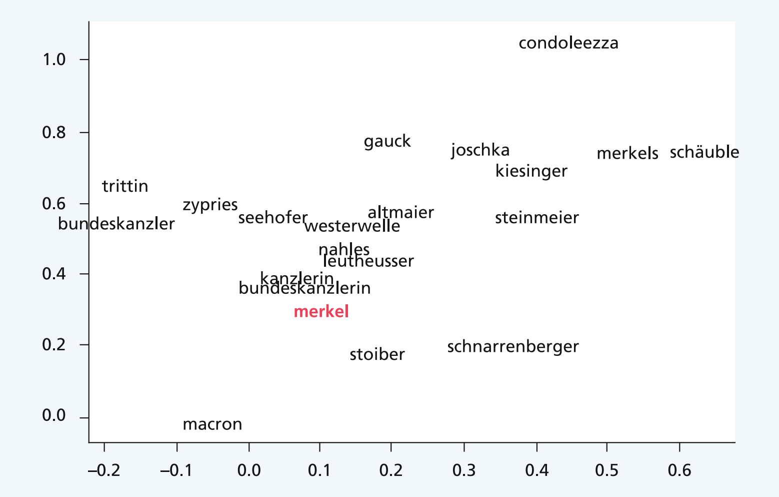


Fig. 6.10 The embeddings of the twenty nearest neighbors of “merkel” projected to two dimensions. The embeddings of length 100 were estimated by the Word2vec model trained with the German Wikipedia

that are farther apart in high-dimensional space also have a high distance in low-dimensional space.

In Fig. 6.10 the embeddings of the 20 nearest neighbors of “merkel” are symbolized in two-dimensional space. Here, e.g., “macron” and “condoleezza” are very far from each other, because the persons in question were mentioned in mostly other sentences (at different times) together with “merkel”. Here one must consider that the mutual distance relations are much more detailed in the 100-dimensional space.

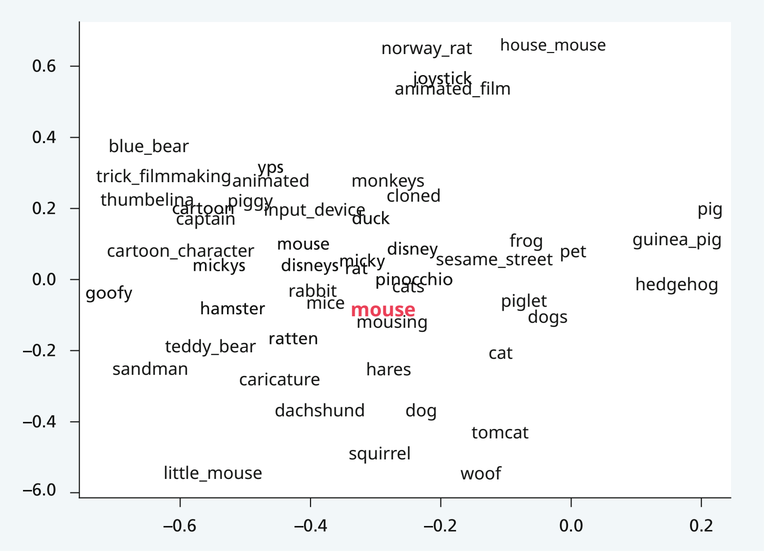
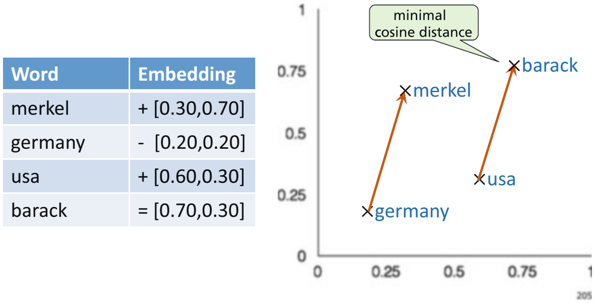


Fig. 6.11 Two-dimensional projection of the embedding vectors of the 50 nearest neighbors of the word “mouse”. The embeddings of length 100 were estimated by the Word2vec model trained with the German Wikipedia. All words were translated from German

In Fig. 6.11 the embeddings of the 50 nearest neighbors of “mouse” are shown. On the left upper corner are mostly terms from the cartoon area, on the right lower corner the names of small animals and in the middle also terms like “input device” and “joystick”. From the examples shown, it is clear that one can learn the meaning of words in the text by using the embeddings. Of course, which terms appear as neighbors depends on the collection of documents being processed. A collection of medical articles will form completely different conceptual fields than those that found in Wikipedia.

6.2.2 Differences Between Embeddings Express Relations

The embeddings have the surprising characteristic that one can interpret distance vectors between embeddings in terms of content. In Fig. 6.12 such a relation is shown in two dimensions. Subtracting the embedding of “germany” from the embedding of “merkel” yields a difference vector (indicated by an arrow in the figure) that captures the relationship between “merkel” and “germany”. Adding this difference vector to “usa” yields a new embedding vector. For this embedding vector, you can search for the word with the closest embedding and obtain “barack”. Thus, one can interpret the difference vector as the relationship between a country and its head of government. Such a relation between concepts is also called analogy,



Word	Embedding
merkel	+ [0.30,0.70]
germany	- [0.20,0.20]
usa	+ [0.60,0.30]
barack	= [0.70,0.30]

Fig. 6.12 Subtracting from the embedding of "merkel" the embedding of "germany" yields a difference vector, indicated by an arrow. Adding this difference vector to the embedding of "usa" generates a new embedding, and "barack" is the word whose embedding is closest

berlin - germany + russia	**dollar - usa + russia**	**merkel - germany + usa**	**merkel - germany + russia**	**red - rose + grass**	**small - woman + man**
moscow	ruble	barack	putin	green	large
petersburg	juganskneftegas	obama	putins	yellow	prusse
kolokolnikov	rubling	reagan	medwedews	blue	big
tschurikov	surgutneftegas	obamas	medwedew	turf	kleinendorf
yekaterinovsky	neftegas	clinton	tschubarow	grass_tufts	slim
sokolnikov	billions	bush	russlan	medium_red	gansen
leningrad	billion	condoleezza	poroschenko	ground_colored	larynx
khvostov	millionov	trumps	kossygin	grass_tuft	east_prussia
yekaterinodar	beresowsky	merkels	juschtschenko	upright	pommerania

Fig. 6.13 Analogies can be calculated with embeddings. The title row contains the words whose embeddings were subtracted and added. The columns contain, in descending order, the words whose embeddings are closest to the result

where the relation between two concepts (here "usa" and "barack") is explained by the relation between two other concepts ("germany" and "merkel").

In Fig. 6.13 there are further analogies computed with embeddings. The headings show which embeddings were subtracted and added. Below are listed the nearest neighbor words whose embeddings have the smallest distances (in descending order) to the resulting embedding. The analogies often apply to the relationship country—capital, country—currency, country—head of government, etc. In the last column, one sees an example in which a possible prejudice in language use is revealed. While a "woman" is assigned the size attribute "small", a "man" is assigned the size attribute "large".

Such analogies can only be extracted for relations that occur frequently in the text. For testing purposes, a set of 18,000 such relations was set up and tested to see how often they were reproduced. The Word2vec method was able to recover 69% of these analogies (Levy et al. 2015).

Embeddings for words can be used to represent the meaning of words. Therefore, embeddings are also employed in web search engines to retrieve words that are similar in meaning rather than just matching strings (Diaz et al. 2016). This leads to a significant improvement of search results.

6.2.3 *FastText Uses N-Grams of Letters*

One problem of the Word2vec method is that each word gets its own embedding and very rare words are ignored. For this reason, a variant was proposed which considers not only words but also parts of words (Bojanowski et al. 2017). This allows the procedure to use syllables and endings as inputs as well and to compose rare word parts from its components Fig. 6.14.

As shown in Fig. 6.15, the word "anarchy" is represented by the whole word and sequences of letters. Such sequences of *n* letters are also called n-grams. Usually all overlapping 3-grams to 6-grams are used.

For each part of this representation a separate embedding is used. As in Word2vec, a model is used which aims to describe the relation of a central word with its neighborhood. Input to the model is the sum of the embeddings of the neighboring words and their letter sequences. With this input, the central word and its letter sequences are to be predicted with a logistic model (Fig. 6.14).

During training, both the parameters of the logistic model and the embeddings are modified so that the prediction becomes as good as possible. In Fig. 6.16, results of FastText are shown. For compound words, the most important letter sequences are shown, which receive the greatest weight in representing the compound word.

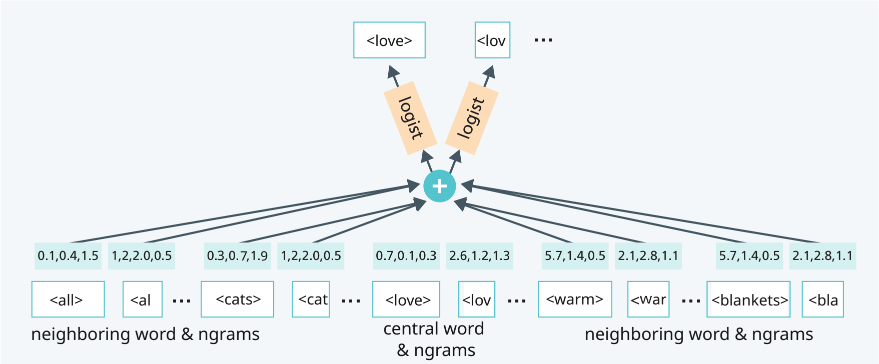
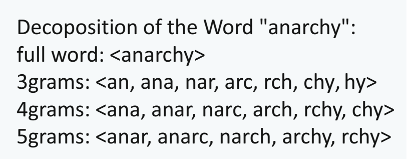
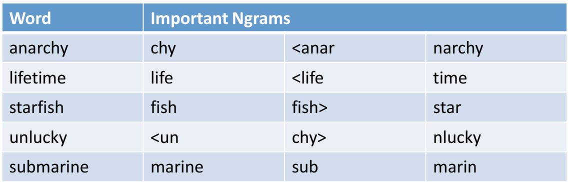


Fig. 6.14 Training the FastText model with the text "all cats love warm blankets . . . ". Each word is additionally represented by its letter sequences (n-grams). For a central word ("love"), a neighborhood of words is considered (here two neighbors). The sum of all embeddings of the letter sequences of the neighboring words is formed. This predicts the central word and its n-grams



Decoposition of the Word "anarchy":
full word: <anarchy>
3grams: <an, ana, nar, arc, rch, chy, hy>
4grams: <ana, anar, narc, arch, rchy, chy>
5grams: <anar, anarc, narch, archy, rchy>

Fig. 6.15 Decomposition of a word into sequences of letters (n-grams). $<$ and $>$ indicate the beginning and the end of the word



Word	Important Ngrams		
anarchy	chy	<anar	narchy
lifetime	life	<life	time
starfish	fish	fish>	star
unlucky	<un	chy>	nlucky
submarine	marine	sub	marin

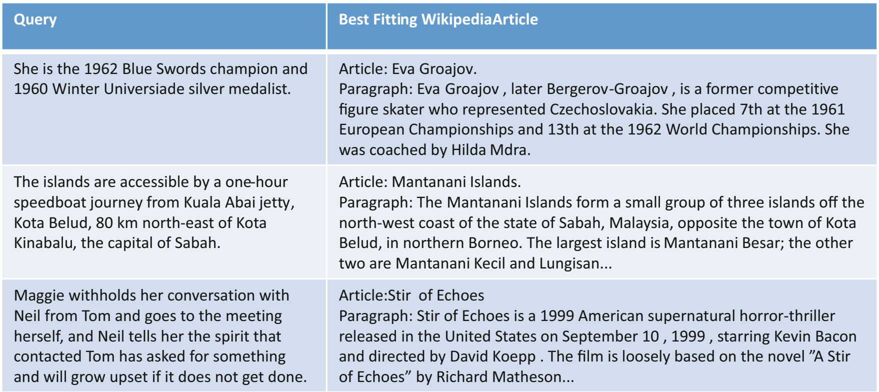
Fig. 6.16 For each word on the left, the n-grams are shown that result in the most significant changes in the FastText representation when omitted

6.2.4 StarSpace Creates Embeddings for Other Objects

So far, we have encountered procedures that describe words by embeddings, which evaluate the neighborhood of different pairs of words. This procedure can be transferred from pairs of words to other types of pairs of objects, e.g., to the neighborhood between words and classes of documents, between a web query and the response documents, between a piece of music and the listeners, between the people of a relationship graph (e.g., network of Facebook friends), etc. All these relationships can be considered using the StarSpace method (Wu et al. 2018). It works according to the same scheme as the FastText algorithm. The obtained embeddings can then be used, for example, for classifying texts into very many object classes, for finding suitable documents in a search engine, for recommending suitable pieces of music, for embedding sentences or documents, and so on. Figure 6.17 shows the results of a search for Wikipedia articles. Those articles were selected that best matched a search query in terms of StarSpace embeddings.

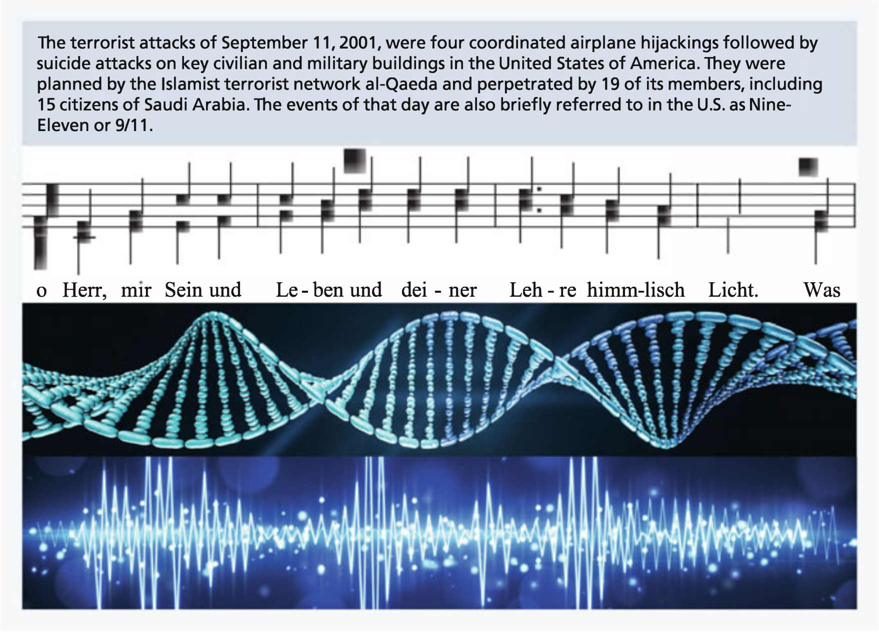
6.3 Recurrent Neural Networks for Sequence Modeling

Recurrent neural networks (RNN) are neural networks specialized for processing sequences (Rumelhart et al. 1986). Sequences are, for example, the consecutive words of a sentence, the notes of a piece of music, the building blocks of a DNA sequence, or even the sounds of a speech recording (Fig. 6.18). Just as CNNs specialize in the two-dimensional structure of pixel matrices in image processing, RNNs process the elements of a sequence. In the following, we will focus on



Query	Best Fitting WikipediaArticle
She is the 1962 Blue Swords champion and 1960 Winter Universiade silver medalist.	Article: Eva Groajov. Paragraph: Eva Groajov , later Bergerov-Groajov , is a former competitive figure skater who represented Czechoslovakia. She placed 7th at the 1961 European Championships and 13th at the 1962 World Championships. She was coached by Hilda Mdra.
The islands are accessible by a one-hour speedboat journey from Kuala Abai jetty, Kota Belud, 80 km north-east of Kota Kinabalu, the capital of Sabah.	Article: Mantanani Islands. Paragraph: The Mantanani Islands form a small group of three islands off the north-west coast of the state of Sabah, Malaysia, opposite the town of Kota Belud, in northern Borneo. The largest island is Mantanani Besar; the other two are Mantanani Kecil and Lungisan...
Maggie withholds her conversation with Neil from Tom and goes to the meeting herself, and Neil tells her the spirit that contacted Tom has asked for something and will grow upset if it does not get done.	Article:Stir of Echoes Paragraph: Stir of Echoes is a 1999 American supernatural horror-thriller released in the United States on September 10 , 1999 , starring Kevin Bacon and directed by David Koepp . The film is loosely based on the novel ”A Stir of Echoes” by Richard Matheson...

Fig. 6.17 StarSpace can use embeddings to calculate the best matching Wikipedia articles to a query (Wu et al. 2018)



The terrorist attacks of September 11, 2001, were four coordinated airplane hijackings followed by suicide attacks on key civilian and military buildings in the United States of America. They were planned by the Islamist terrorist network al-Qaeda and perpetrated by 19 of its members, including 15 citizens of Saudi Arabia. The events of that day are also briefly referred to in the U.S. as Nine-Eleven or 9/11.

o Herr, mir Sein und Le - ben und dei - ner Leh - re himm-lisch Licht. Was

Fig. 6.18 Examples of sequences: words of a document, notes and lyrics of a piece of music, DNA sequence, and sounds of an acoustic event. Image credits in Appendix A.3

sequences of words, but the corresponding models can be applied to other types of sequences as well.

The DNNs we have encountered so far have a fixed-size input (vector, matrix, or tensor). Sequences, on the other hand, have a variable length. Thus, a new scheme has to be found to process sequences with neural networks.

One important task is to predict the next word in a sentence. Such a model is also called a language model because it has to recognize the grammatical and semantic relationships of the words in a text. Since there are usually many possible words in the next position, one calculates for each word in the vocabulary the probability that it is the next word. This requires information about which words have appeared previously in the sentence. Language models are another example of self-supervised learning (Sect. 3.1.2), where one part of the observed training data is predicted from another part. We will see that language models can be used to produce grammatically correct texts with consistent content.

6.3.1 Recurrent Neural Networks as Language Models

We have seen that in DNN the information from the input vector is transformed by the operator corresponding to a layer (Sect. 4.3.1). The result is a hidden vector representing the input. Similarly, a hidden vector can be used to represent the backward part of a sentence. The words of the vocabulary can appear at almost any position in a sentence. Therefore, one needs a network that performs the same processing at the different positions in the sequence. Since such a network uses the same operators over and over again, the name recurrent neural network has been chosen.

The operation of an RNN is shown in Fig. 6.19 for individual steps. Here, the task is to predict the probabilities of the words of the sentence *"The boy ran home"* one by one.

First, for the prediction of the running output word y_t, the **previous** word v_{t-1} is encoded as input to the network. For this purpose an embedding vector $x_{t-1} = emb(v_{t-1})$ of length n_x (e.g., $n_x = 100$) is used, which can represent the meaning of a word v_{t-1} at position $t-1$ (Sect. 6.2.1). For the non-existent word v_0, embedding $x_0 = (0, \dots, 0)$ is taken.

Then, a hidden vector h_{t-1} with a preselected length (e.g., $n_h = 200$) is defined to store the contents of the previous words of the sentence before position t. At the beginning ($t = 1$), h_t is usually set to 0. In the simplest case, the RNN consists of a linear transformation and a nonlinear activation, which computes the current hidden vector h_t from the previous hidden vector and the embedding x_{t-1} of the previous word (Elman 1990)

$$h_t = \tanh(U * h_{t-1} + W * x_{t-1} + b)$$

The matrices U and W and the vector b are unknown parameters. The probability of the next word v_t is predicted from h_t with a logistic regression model:

$$\hat{p}(v_t | v_{t-1}, \dots, v_1) = \text{softmax}(A * h_t + c)$$

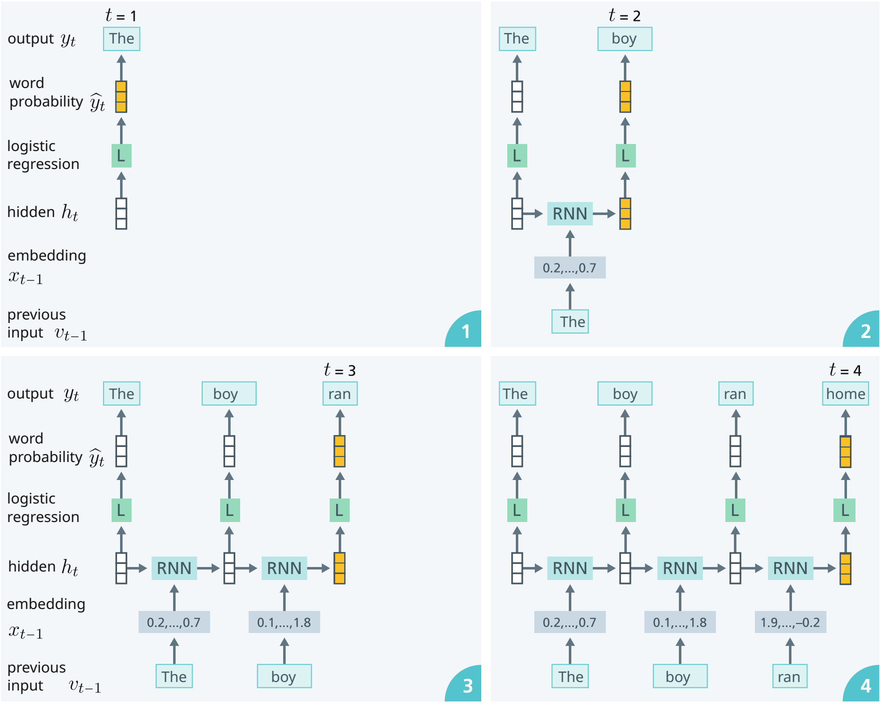


Fig. 6.19 Recurrent neural network for predicting the probability of the next word. The vectors computed in each step are colored orange. At the beginning ($t = 1$ top left), the probability of the first word (observed was *"The"*) is predicted from the initial hidden vector h_1 using the logistic regression model L. Then ($t = 2$ top right), the RNN predicts the next hidden vector h_2 from embedding x_1 of the previous word *"The"* and the previous hidden vector h_1. The logistic regression model L then computes the probability of the next word (observed was *"boy"*) from the hidden vector h_2. Subsequently, the probabilities of the observed words *"ran"* ($t = 3$) and *"home"* ($t = 4$) are predicted analogously

The conditional probability is only approximated by the RNN; therefore, it has been tagged with the estimation symbol $\hat{p}$. The matrix A and the vector c are also unknown parameters. Figure 6.20 lists the computational steps necessary to determine the conditional probabilities of the words in a sentence. The conditional probability is a numeric vector containing a probability for each word in the vocabulary. If the vocabulary has a size $n_v = 50,000$, the probability vector comprises 50,000 probabilities that add up to 1.0. Note that the previously computed hidden vector h_{t-1} enters the computation of the next hidden vector h_t along with the embedding $emb(v_{t-1})$. Thus, the hidden vector can potentially store the contents of all previous words.

Fig. 6.20 Computing the conditional probability of the next word in a text

```
For $t = 1, \ldots, 4$
    1. Determine the embedding
       $x_{t-1} = emb(v_{t-1})$ for word $v_{t-1}$
    2. Compute the hidden vector
       $h_t = \tanh(U * h_{t-1} + W * x_{t-1} + b)$
    3. Compute the probability of the next word
       $\hat{p}(v_t | v_{t-1}, \ldots, v_1) = softmax(A * h_t + c)$
```

6.3.2 *Training of Recurrent Neural Networks*

The RNN approximates the conditional probability of the next word when the previous words are known. This can be used to calculate the probability of the whole sentence. In our example this results in

$$p(\text{The boy ran home}) = p(\text{home}|\text{The boy ran}) * p(\text{ran}|\text{The boy}) * p(\text{boy}|\text{The}) * p(\text{The})$$

This relationship follows directly from the definition of the conditional probability. This means that one can calculate the probability of a complete document with T words by the conditional probabilities of the next words in each case. If a language model is trained using a large training set of texts, the plausibility of a new text can then be evaluated very accurately. Thus, one can potentially reproduce all syntactic, but also all semantic relations between words that occur in the training set. We will see later that there are advanced language models which approximate the actual conditional probability very well and can thus generate texts that are surprisingly consistent in content.

The prediction of the conditional probabilities depends on the unknown parameters, which are randomly set to small values at the beginning. Unknown parameters in our case are the matrices U, W and the vector b of the RNN, the matrix A and the vector c of the logistic model and the embeddings $emb(v_t)$ of the words, i.e. $w = (U, W, A, b, c, emb(v))$ is the vector of all parameters. w is to be modified by optimization in such a way that the probability of the training data becomes as high as possible.

If the training set consists of N sentences $s_1, \ldots, s_N$, parameters are changed so that the product of the probabilities of all sentences becomes as high as possible (Sect. 3.5.1)

$$\max_w \prod_{i=1}^{N} \hat{p}(s_i; w)$$

To avoid too small numbers, this expression is transformed with (− log) and we have to minimize

$$L(w) = \sum_{i=1}^{N} - \log \hat{p}(s_i; w) = - \sum_{i=1}^{N} \sum_{t=1}^{n_i} \log \hat{p}(v_{i,t}|v_{i,t-1}, \ldots, v_{i,1}; w) \tag{6.1}$$

if the i-th sentences consists of the words $v_{i,1}, \ldots, v_{i,n_i}$. Analogously to Sect. 3.5.2, this defines the loss function $L(w)$ of the RNN. To optimize the RNN for a minibatch of sentences in the training set, the gradient of the loss function $\partial L(w)/\partial w$ is needed. This is automatically provided by modern Neural Network toolkits. Then stochastic gradient descent (SGD) optimization produces near-optimal parameters (Sect. 3.6.6). As with the Word2vec model, Noise Contrastive Estimation (Sect. 6.1.3) is used for the approximate estimation of word probabilities.

During the optimization, the probability of the next words in asentence are calculated from the previous words. In contrast to the DNNs considered so far, we do not have to consider only a single observed word for the loss function, but a number of different observed words. All these loss values are evaluated simultaneously during the optimization and collectively affect the gradient according to the loss function (6.1). Figure 6.21 shows the information flow that takes place during the gradient calculation. Here, the calculation direction of the forecast (gray arrows) is reversed. From each output word, the error signal (loss value) is first propagated back to the logistic model (orange arrows). Then it is transferred to the RNN module. From there, it is propagated back to the previous RNN modules. Finally, the error signal is transferred to the embeddings. Based on this information, the parameters of the RNN, the logistic model and the embeddings are modified simultaneously.

6.3.3 *The Properties of RNN Gradients*

There is a remarkable effect due to the repeated transformation of the hidden vector h_t by the RNN. The derivation in Fig. 6.22 for a very simple RNN with a one-dimensional hidden vector h_t shows that the derivatives with respect to a word appearing k positions earlier in the sentence always contain the factor u^k, where u is a parameter.

If $u = 2$, then for $k = 20$ the value of u^k is greater than one million. This case is called exploding gradient. Changing the parameter with this gradient during optimization, the parameter value takes an extreme value where the model behavior becomes very unrealistic and all future gradients can be close to zero.

A simple and effective solution to this problem is to limit the Euclidean length l_{grad} of the gradient vector $\partial L(w)/\partial w$. If the vector is longer than a threshold γ, e.g. $\gamma = 5.0$, the length of the gradient is reduced to γ by multiplication with the

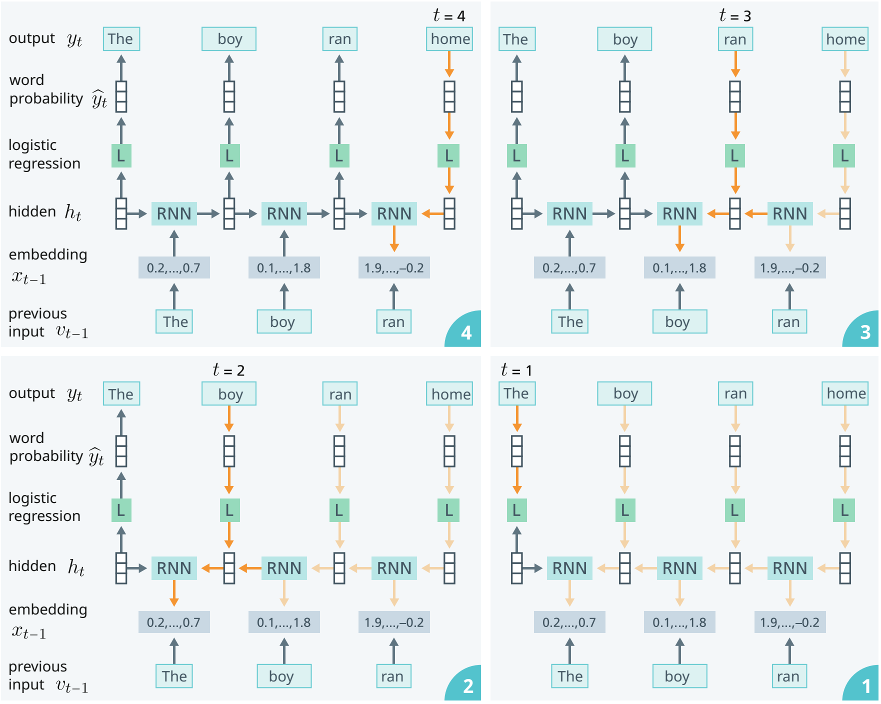


Fig. 6.21 Information flow during back propagation of the error signal, symbolized by orange arrows. The values of the hidden vectors stored during forward propagation are employed. First, the error signal of the forecast of *"home"* is evaluated and derivatives for the logistic model, the RNN, and the embeddings of *"ran"* are calculated ($t = 1$). Then, the error signal for the prediction of *"ran"* is combined with the error signal of *"home"*. From this, derivatives for the logistic model, the RNN and the embeddings of *"boy"* are calculated ($t = 3$). The same procedure is followed in the bottom row for the remaining input words *"boy"* ($t = 2$) and *"The"* ($t = 1$)

RNN with one-dimensional hidden vector h_t

$$h_t = \tanh(u * h_{t-1})$$

$$\frac{\partial\, h_t}{\partial h_{t-1}} = \frac{\partial \tanh(x)}{\partial x}(u * h_{t-1}) * \boldsymbol{u}$$

by the chain rule. After two steps

$$h_t = \tanh(u * \tanh(u * h_{t-2}))$$

$$\frac{\partial\, h_t}{\partial h_{t-2}} = \frac{\partial \tanh(x)}{\partial x}(u * h_{t-1}) * \boldsymbol{u} * \frac{\partial \tanh(x)}{\partial x}(u * h_{t-2}) * \boldsymbol{u}$$

That is, at each step we get an additional factor u.

After k steps we get:

$$\frac{\partial\, h_t}{\partial h_{t-k}} = \boldsymbol{u}^{\boldsymbol{k}} * (\text{rest terms})$$

Fig. 6.22 Recurrent neural networks have an exponential factor u^k in the derivatives. As a result, the gradients can become very large: exploding gradients

scalar factor γ / l_{grad}, and the gradient update changes the parameter vector w in the right direction with a moderate step size. This strategy is called gradient clipping.

On the other hand, if $u = 1/2$, then for $k = 20$ the value of u^k is less than one millionth. This phenomenon is called vanishing gradient. This means that far-distant relationships between words cannot be exploited by gradients; these small values have little effect during optimization and are almost completely obscured by the random fluctuations of the stochastic gradient descent.

In most languages, connections between widely separated words in a sentence are important. Example: *"Barack Obama first saw the king in his palace and then met with the head of the government."*. Obviously, *"Obama"* is the subject of the verb *"met"*, and the sentence is intelligible only if the relationship between these two words is captured. For this reason, simple RNNs were not very successful.

6.4 The Long Short-Term Memory (LSTM) is a Long-Term Memory

6.4.1 Controling Memory Operations by Gates

A solution was proposed by Hochreiter and Schmidhuber (1997) (Fig. 6.24). They designed a network called Long Short-Term Memory (LSTM), which uses an additional memory vector c_t as a long-term memory for dependencies between distant inputs (Figs. 6.23, 6.24 and 6.25).

This memory vector is intended to store information for an extended period of time and to be retrieved and erased as needed. In contrast to a digital computer, it is not possible to select a memory area that is filled with data by a processor read out later. Rather, an alternative mechanism is needed that can delete, store and retrieve data and that can also be trained via gradients.

This mechanism is shown in Fig. 6.23 and consist of gates. As input it takes the memory vector c_{t-1}, which is a vector of given length of arbitrary numbers. Further, there is a control vector s_t of the same length, which may only contain numbers between 0.0 and 1.0. These two vectors are multiplied component by component to give the result c_t. If a component of the control vector has the value …

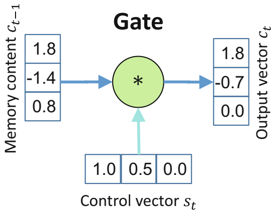
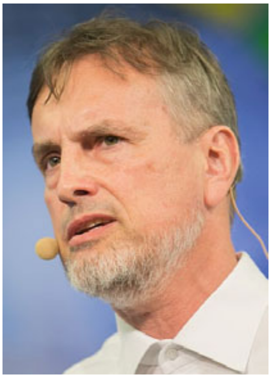


Fig. 6.23 A gate modifies each component of the memory contents by multiplication with a number between 0.0 and 1.0



Jürgen Schmidhuber studied computer science and mathematics at the TU Munich. Together with Sepp Hochreiter he analyzed the problem of vanishing gradients. As a solution, both developed the Long Short-Term Memory (LSTM). In 1995 he became scientific director of IDSIA in Lugano. By using GPUs, he was able to greatly reduce the training time requirements and thus greatly increase the accuracy of CNNs in 2011. He and his research group provided important contributions to many areas: Robot control in partially visible environments, music composition, aspects of speech processing, and handwriting recognition. In 2016, he received the IEEE CIS Neural Networks Pioneer Award. Because of the problems expected in the labor market due to AI, Schmidhuber recommends an unconditional basic income.

Fig. 6.24 Jürgen Schmidhuber, one of the inventors of the recurrent neural network LSTM, * 17.01.1963 in Munich. Image credits in Appendix A.3

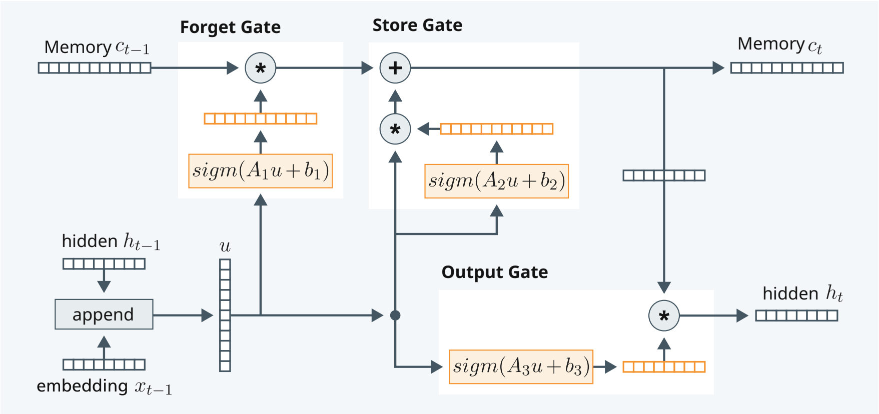


Fig. 6.25 Simplified scheme of the LSTM. A memory vector c_t shall is designed to store content for a long time. The previous hidden vector and the embedding of the previous word are concatenated and form the input u. The control vectors of the gates are generated from u using different sigmoid layers (orange). A forget gate is used to delete individual components of the memory. The store gate adds content from u to individual components of memory c_t. With the output gate, individual components of the memory c_t are retrieved and form the new hidden vector h_t

- 1.0, the corresponding component of the memory vector is not changed.
- 0.0, the corresponding component of the memory vector is set to 0, i.e. deleted.
- a value between 0.0 and 1.0, then the corresponding component of the memory vector is reduced accordingly, i.e. partially deleted.

Hochreiter et al. (1997) proposed the use of three gates that can independently delete components of the memory vector (forget gate), add values to the memory vector (store gate), and retrieve values from the memory gate (output gate). In this way, they achieved a high flexibility in changing and using the contents of the memory vector.

One still needs a way to generate the control vectors for the individual gates. Hochreiter et al. (1997) used extra modules with sigmoid activation functions (Fig. 4.5), which always generate a vector of outputs between 0.0 and 1.0.

$$s_t = \text{sigmoid}(Au_{t-1} + b)$$

Here $u_{t-1} = [h_{t-1}, x_{t-1}]$ is the hidden vector h_{t-1} to which the current input x_{t-1}, the embedding of the current word, is attached. Thus, u_{t-1} includes all information available at time $t-1$.

The resulting RNN unit is shown in Fig. 6.25. In the lower left, $u = u_{t-1}$ is first formed by concatenating the hidden vector h_{t-1} and the input x_{t-1}. The forget gate is controlled by a sigmoid layer with input u_{t-1} and erases individual components of the memory vector c_{t-1}. The store gate is also controlled by another sigmoid layer with input u_{t-1} and adds individual components of u_{t-1} to the memory vector c_{t-1}. The output gate is again controlled by another sigmoid layer with input u_{t-1} and generates the current hidden vector h_t from the memory vector c_t.

To gain additional flexibility, Hochreiter et al. (1997) further extend the network with two fully connected layers with tanh activation, which provide the connection between the memory vector c_t and the inputs/outputs. The full LSTM is shown in (Fig. 6.26) and forms an LSTM cell that can be used in place of the simple RNN.

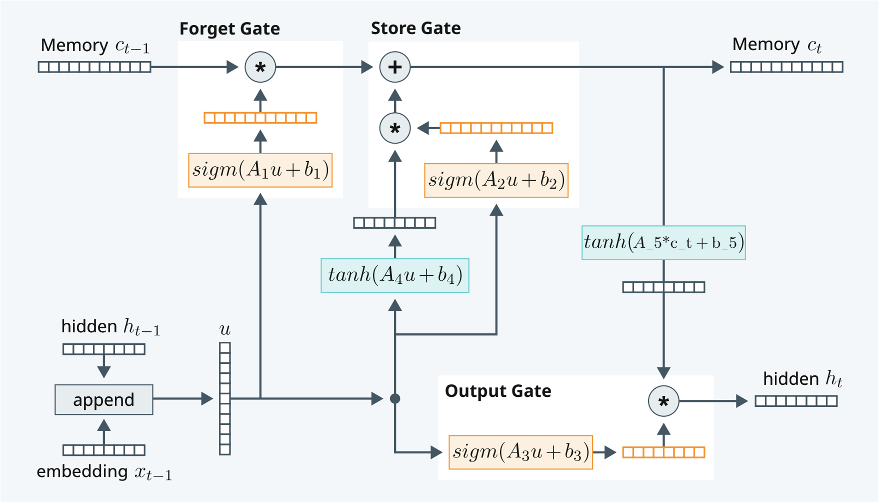


Fig. 6.26 In addition to the gates (orange) from Fig. 6.25, the complete LSTM contains two trainable tanh layers (blue) to transform the inputs to the memory vector and prepare the memory vector for the output gate

The memory mechanism of the LSTM is differentiable with respect to its parameters and can therefore be adapted to the data by training. During training, the unknown parameters of the subnetworks $A_1, \ldots, A_5, b_1, \ldots, b_5$ are optimized by stochastic gradient descent. Each component of the memory vector represents a specific aspect that is automatically defined during training. The length of the memory vector is usually chosen between 100 and 2000, therefore the memory vector can store a large spectrum of aspects. Training also determines how long an aspect remains stored and when it is cleared depending on the input. Regularization methods must always be used to avoid overfitting. Very popular is the dropout (Sect. 4.6.6), which sets randomly selected parameters to 0.

It is amazing that even for this complicated model the training with the stochastic gradient descent works. Note that in a gate two quantities are multiplied by each other, namely the control vector with the corresponding input vector. The gradient can also be calculated for this structure. Here, the error signal from the output of a gate is split and propagated back to the input and to the control vector. However, the computational effort for the training can take hours or several days depending on the problem.

6.4.2 LSTMs with Multiple Layers

One can improve LSTMs by using multiple layers of LSTM cells with different parameters. The hidden vectors $h_{1,t}$ produced by the first layer serve as input for the LSTM cell of the next layer (Fig. 6.27). In case of further layers, this procedure is repeated. The calculated hidden vectors of the top layer are used as input for the logistic regression model to predict the probability of the next word.

As observed in other DNNs, additional layers can increase the representation capacity of the multilayer RNN. While an additional layer of the same size doubles the number of parameters, doubling the length of the hidden vectors and the memory vectors quadruples the number of parameters. For this reason, multiple layers are advantageous and one often uses 2–4 layers in LSTM networks.

6.4.3 Applications of the LSTM

In recent years, sufficient computing power has been available to successfully train LSTM networks. In one application, an LSTM was trained on 96 MB of text from the English Wikipedia, which still contained markups for links to other articles. Figure 6.28 shows a text (Karpathy 2015) generated by the LSTM. The text is grammatically correct to a large extent, but the content is just made up. Many of the terms used do not exist. As can be seen, LSTMs lose context after a number of words and then generate text on other topics. The model can be used to generate any number of texts of this type.

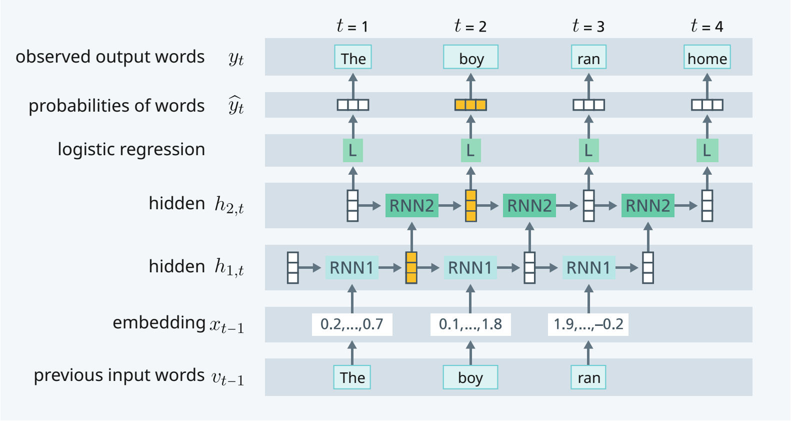


Fig. 6.27 LSTM with two layers. For example, to predict the probabilities for the word at position ($t = 2$), RNN1 uses the hidden vector $h_{1,t=1}$ of the first layer and the embedding of the previous word *"The"* as input and generates the hidden vector $h_{1,t=2}$. Together with $h_{2,t=1}$, this serves as input to RNN2, which predicts $h_{2,t=2}$. This vector is used by the logistic regression model L to predict the probability of the next word at position $t = 2$ (observed *"boy"*). The vectors calculated in this way are colored orange. For all words of the sentence, probabilities are predicted according to this scheme

> Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communicaon, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Anoch, Perth, October 25—21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

Fig. 6.28 This text was generated by an LSTM trained on texts from the English Wikipedia (Karpathy 2015). It does not stick to one topic, but has a tendency to connect many themes in a random way. The training texts contained links *"[[. . .]]"* to other Wikipedia articles

A standard corpus for testing language models is the Penn Treebank corpus. It contains about 4.5 million words in American English, which have been additionally annotated with syntactic properties. This corpus was used to train different types of language models, each of which predicts the next word. As a measure of the quality of the prediction of the language model one uses the perplexity. This is the inverse probability of the test set $w_1, \ldots, w_N$ normalized by the number of words in the test set, i.e. $p(w_1, \ldots, w_N)^{-1/N}$. The smaller the perplexity, the better the model can predict the words in the test set. Figure 6.29 shows the perplexity for different recurrent models. LSTMs have been able to improve the perplexity substantially in recent years.

Another LSTM network operates at the letter level: it predicts one letter of text after the next. It uses three LSTM layers, with memory and hidden vectors each having size 512. The works of Shakespeare with a size of 4.4 MB were used as

Fig. 6.29 Perplexity for the Penn Treebank corpus for different RNN models. The smaller values are better (Pascanu et al. 2013)

Authors	Model / layers	Number of Parameters	Perplexity
(Pascanu et al. 2013)	RNN	6.1 mil	107.5
(Melis et al. 2017)	LSTM / 4	10.0 mil	60.1
(Melis et al. 2017)	LSTM / 4	24.0 mil	58.3
AWD-LSTM	LSTM / 3	24.0 mil	57.3

Fig. 6.30 This text was generated by a three-layer LSTM network trained on the letters of Shakespeare's plays (Karpathy 2015)

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep. and subjects of his death, I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

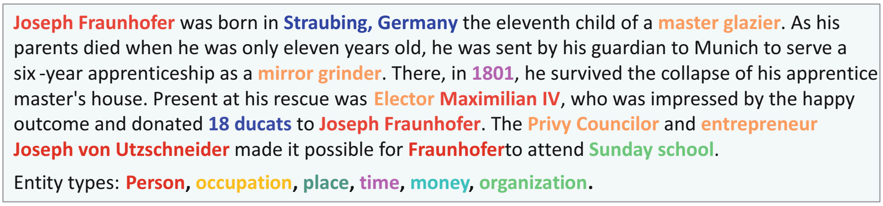
DUKE VINCENTIO:
Well, your wit is in the care of side and that.

training material. The generated text captures Shakespeare's linguistic style well and contains only few orthographic errors (Fig. 6.30). However, as with the Wikipedia model, the context of the content is lost after a number of words and the model jumps to other topics.

The basic LSTM structure in Fig. 6.26 was constructed according to heuristic considerations, and it is not at all clear whether this structure is optimal. Melis et al. (2017) have studied a number of recurrent neural network architectures and compared their performance. All models had the same number of trainable parameters and the hyperparameters were extensively optimized. Overall, LSTM networks had slight advantages over other network architectures. In particular, it was also shown that regularization by appropriate dropout is very important.

6.4.4 Bidirectional LSTM Networks for Word Property Prediction

One application for RNNs is the prediction of characteristics of sequence elements. Every day new names (named entities) of people, products, organizations, places, etc. appear in the press and on the Internet. For example, there are over 10 million geographical names in the GeoNames database, and virtually every common word in the language appears there as a place name. Therefore, it is not possible to identify names by matching them against lists; instead, one must identify the properties of names using contextual information. This information extraction task is called Named Entity Recognition (NER) (Fig. 6.31).



Joseph Fraunhofer was born in Straubing, Germany the eleventh child of a master glazier. As his parents died when he was only eleven years old, he was sent by his guardian to Munich to serve a six -year apprenticeship as a mirror grinder. There, in 1801, he survived the collapse of his apprentice master's house. Present at his rescue was Elector Maximilian IV, who was impressed by the happy outcome and donated 18 ducats to Joseph Fraunhofer. The Privy Councilor and entrepreneur Joseph von Utzschneider made it possible for Fraunhoferto attend Sunday school.

Entity types: Person, occupation, place, time, money, organization.

Fig. 6.31 Wikipedia text about the scientist and entrepreneur Joseph Fraunhofer with annotated named entities

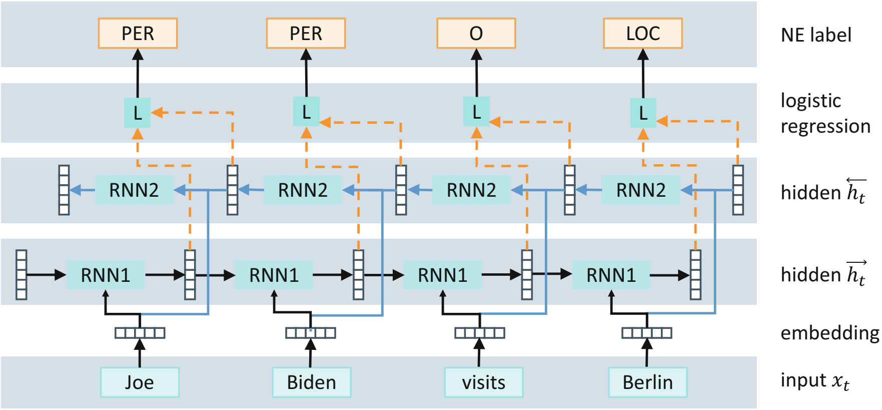


Fig. 6.32 Bidirectional LSTM (BiLSTM) for word property prediction. RNN1 in the bottom layer predicts from the beginning of the sentence to the end of the sentence (black arrows) and RNN2 in the second layer predicts from the end of the sentence to the beginning of the sentence (blue arrows). Both hidden vectors are concatenated and used to predict the label by the logistic regression model (orange arrows)

A benchmark dataset for this task is the CoNLL 2003 dataset. It consists of English news agency reports from Reuters with 17 million words. Each word was manually annotated with one of the categories PER (person), ORG (organization), LOC (location), MISC (other name), and O (other word).

Figure 6.32 shows a bidirectional LSTM model for this task. As input, it uses embeddings for words, which are estimated during training. Lample et al. (2016) propose to use a "forward" LSTM that updates hidden vectors from the start to the end of the sentence and stores information on previous words in its hidden vectors $\overrightarrow{h_t}$. Another "backward" LSTM propagates other hidden vectors from the end of the sentence to its beginning and stores the information on words appearing later in the sentence $\overleftarrow{h_{t+1}}$. The actual prediction of the NER categories is done by a logistic regression model, which has the concatenated hidden vectors $[\overrightarrow{h_t}, \overleftarrow{h_{t+1}}]$ of the two LSTMs as input. Such a network is called a bidirectional LSTM because it operates in two directions.

Fig. 6.33 Recurrent neural networks can also be used to predict time series, such as stock market prices. Image credits in Appendix A.3

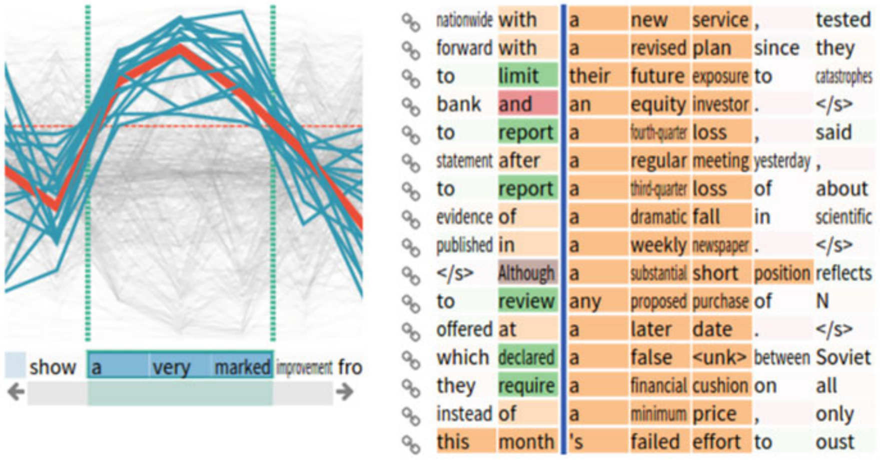
The embeddings can in principle be computed using the CoNLL data. However, since the dataset is relatively small, Lample et al. (2016) use precomputed embeddings of length 100 trained on very large document collections. To compute the embeddings, they employ a variant of Word2vec that takes into account the order of words in the neighborhood.

With this model Lample et al. (2016) achieve an F-value of 90.2% for the English CoNLL data and of 73.1% for the German CoNLL data. The F-value is the harmonic mean of precision and recall. The F-values were averaged over all categories. Thus, on average, the English model correctly recognizes nine out of ten names, while the German model correctly identifies only about three out of four.

RNNs can also be used to predict the next values of time series (Fig. 6.33). A time series is a temporally ordered sequence of one or more measured values x_t, e.g., stock prices of several companies at certain points of time t. Here, instead of word embeddings, the vector x_t of time series values at time t is used as input to the RNN. The output is not a probability of words, but the vector of the target values, e.g. the vector y_{t+1} of time series values at the next time point. Mushtaq et al. (2019) give an overview of such approaches, with LSTM models playing a prominent role. They can be used in economic forecasting, in meteorology, or even in monitoring engineering processes.

6.4.5 Visualization of Recurrent Neural Networks

LSTMs have extensive memory vectors and hidden vectors and are capable of capturing complex relations between words and phrases. Therefore, it is helpful to understand how these relations are stored in the vectors. Strobelt et al. (2016) have developed a visualization method for this task. Given a trained language model, the



nationwide	with	a	new	service	,	tested
forward	with	a	revised	plan	since	they
to	limit	their	future	exposure	to	catastrophes
bank	and	an	equity	investor	.	</s>
to	report	a	fourth-quarter	loss	,	said
statement	after	a	regular	meeting	yesterday	,
to	report	a	third-quarter	loss	of	about
evidence	of	a	dramatic	fall	in	scientific
published	in	a	weekly	newspaper	.	</s>
</s>	Although	a	substantial	short	position	reflects
to	review	any	proposed	purchase	of	N
offered	at	a	later	date	.	</s>
which	declared	a	false	<unk>	between	Soviet
they	require	a	financial	cushion	on	all
instead	of	a	minimum	price	,	only
this	month	's	failed	effort	to	oust

Fig. 6.34 Visualization of components of the memory vector with activation above a threshold for the words *"a very marked"* are selected. Curves show the activation over time. In the table other text passages are reproduced in which the activations of these memory components are also high (intense coloring). Image credits in Appendix A.3

user selects a numerical threshold and a passage of text in a sentence, e.g. *"a very marked"*. The system analyzes this text passage and selects those components of the memory vector that have an activation for the words in the text passage that is higher than the threshold. This is shown in the left of Fig. 6.34.

The system then examines the activation of the selected components for the rest of the text and indicates text passages (right side of Fig. 6.34) in which the activation of the components is also high, e.g. *"a new service"* or *"a regular meeting"*. These passages are indicated by intense coloring. Apparently, these word sequences cause similar activation of the components of the memory vector. These passages contain phrases with nouns (nominal phrases), almost all of which contain the word *"a"*. Flexible selection of match criteria can be used to refine the visualization and illustrate the meaning of individual components of the memory vector. The visualization can be used for the hidden vectors as well as the memory vectors of any layers of the RNN.

6.5 Transformation of one Sequence into Another Sequence

The automatic translation of texts from one language to another is an old dream of mankind. With the availability of computers, research on this subject began in the middle of the last century. Initially, machine translation was mainly used in the military field. Since the 1990s, translation programs were available for consumers.

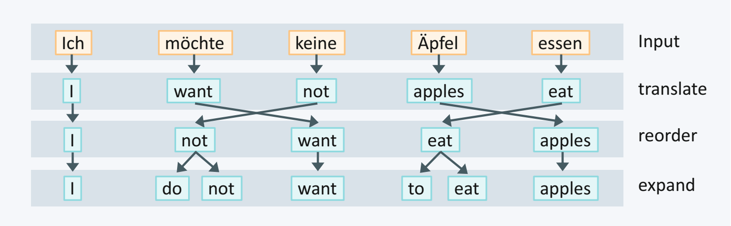


Fig. 6.35 A translation requires several steps: after a rough translation, words often need to be rearranged. Additional words and inflections may need to be introduced to match the grammatical structure of the target language

However, the available systems were often unable to provide a satisfactory translation.

Figure 6.35 shows the difficulties involved in a translation. It is not enough to translate individual words and phrases (self-contained units of one or more words), but the words must also be rearranged and, if necessary, additional words must be inserted and suffixes modified. This cannot be captured by a few rules. Common, fluent phrases must be memorized, and extensive knowledge of the world is also required. An example is the sentence *"I saw the man on the hill with the telescope"*. Here it is not clear whether *"see with the telescope"* or *"the hill with the telescope"* is meant. This can only be determined by looking at the context across sentences.

If a person can translate one language into another, she basically has to understand both languages. She must grasp the meaning and context of the words and must have world knowledge in order to know the correct equivalent of phrases. Already the ancient Greeks distinguished between *"metaphrase"* (literal translation) and *"paraphrase"*, the circumscription of a content by formulations which reflect the sense of the source text.

A translation is often ambiguous. Quine (1969) noted that there is often no objective preference between different translation variants. Moreover, language is generally interpretable only in the context of one's own experience. Bar-Hillel (1962) even believed that machine translation is not only practically but also theoretically impossible. *"Human translators often unconsciously use their background knowledge to resolve syntactic and semantic ambiguities"*. Machines often cannot resolve these ambiguities or apply a mechanical *"rule, which often results in an incorrect translation"*. Thus, according to Bar-Hillel, the ability to translate a language is a capacity of intelligence that cannot be automated.

6.5.1 Sequence-to-Sequence Networks for Translation

In the 2000s, statistical methods were often used in translation. In this process, words and phrases of the source and target languages were assigned to each other according to their frequency from large collections of translations. However, the accuracy of these translation methods was found to be insufficient by many users.

Sutskever et al. (2014) proposed a new type of DNN built on recurrent neural networks. They started from the observation that a recurrent neural network is able to "*store*" a text in a hidden vector. This led to the idea of using an encoder RNN to encode the source language text as a hidden vector during translation. This hidden vector then forms the initial hidden vector of a decoder RNN that translates the sentence into the target language. Such a DNN is called a sequence-to-sequence model (Fig. 6.36).

The decoder RNN predicts the probability of the output words of the target language. During training it receives the information about the correct previous observed word (or the beginning-of-sentence symbol <BOS>) via its embedding for the prediction of the next word and can use this in the computation. Note that this scheme is correct because of the definition of the conditional probability of the target text $z_1, \ldots, z_m$ given the source text $x_1, \ldots, x_k$

$$p(z_1, \ldots, z_m|x_1, \ldots, x_k) = p(z_m|z_1, \ldots, z_{m-1}, x_1, \ldots, x_k) * \cdots * p(z_1|, x_1, \ldots, x_k)$$

Using the previously observed words speeds up training, similar to the language model. In this way, probabilities for the subsequent words of the sentence (or the end-of-sentence symbol <EOS>) are generated one after the other. This is done until a maximum length is reached.

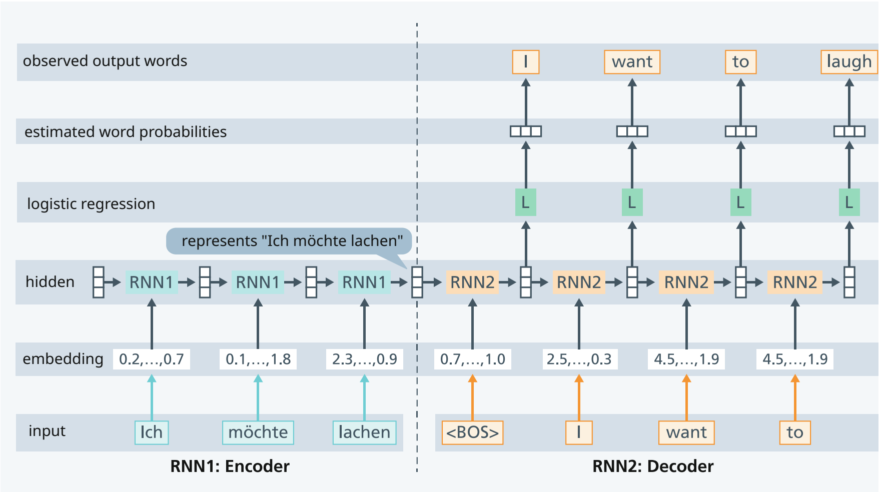


Fig. 6.36 A sequence-to-sequence model for translating a sentence. The encoder RNN1 (left) encodes the sentence of the source language into the hidden vector, which is the initial value for the hidden vector of the decoder RNN2 (right). A logistic model **L** predicts the probabilities of the output words. During training the previous correct word of the target language is used as input for the translation of the next word

The model is trained as a whole in an optimization process. A large error results if the predicted probability of the correct output word is low. The error signal is propagated back to the encoder RNN1 and finally to the embeddings via the logistic model and the hidden vectors of the decoder RNN2. The information flow runs in the opposite direction of the arrows in Fig. 6.36. Unknown parameters are the embeddings of the words in both languages, and the parameters of the logistic model, the decoder RNN2 and the encoder RNN1.

For training, the model of Sutskever et al. (2014) uses an extensive dataset of 12 million English sentences and their French translations. A fixed vocabulary of 160,000 frequent English words and 80,000 frequent French words was used. As the encoder RNN and decoder RNN, Sutskever et al. (2014) respectively utilized LSTM models with four layers (Sect. 6.4.2) with hidden vectors of length 1,000 and embeddings of length 1000. The resulting RNN has a total of 380 million parameters, which are optimized by stochastic gradient descent. Although the training was parallelized on 8 GPUs, it took 10 days.

Creating and Evaluating a Translation

When the translation model (Fig. 6.36) is trained, it can compute the probability of a word sequence for the target language. In the application, however, a concrete translation is needed, probabilities alone are not sufficient. Theoretically, one could generate all word sequences and keep the one with the highest probability. This is impossible because of the computational effort. The alternative is to always select the subsequent word with the highest probability. But this "greedy" strategy quickly leads to inferior results. As a compromise, one can perform a beam search, which always tracks k alternatives in parallel. In Fig. 6.37 this is demonstrated with a small example with $k = 2$. Values between 2 and 20 are common for k.

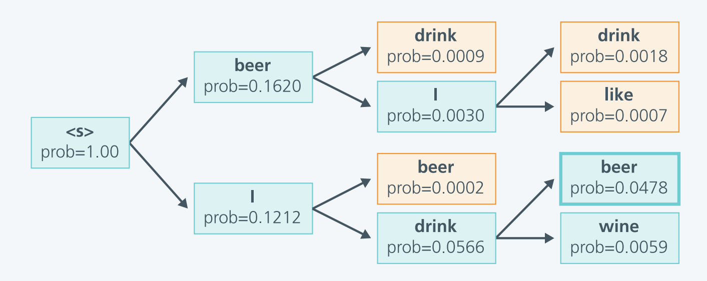
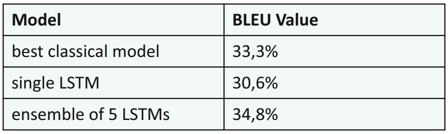


Fig. 6.37 Beam search ($k = 2$) to generate a translation of the sentence *"Ich trinke Bier"*. To predict the next word, the k subsequent words with the highest path probabilities are generated. Then, all paths except the k paths with the highest path probabilities are deleted (orange). These two steps are repeated for each of the k paths until the end of the sentence is reached for all paths. Finally, the path with the highest overall probability is selected

Ulrich UNK , membre du conseil d' administration du constructeur automobile Audi , affirme qu' il s' agit d' une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d' administration afin qu' ils ne soient pas utilisés comme appareils d' écoute à distance .

Ulrich Hackenberg , membre du conseil d' administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu' ils ne puissent pas être utilisés comme appareils d' écoute à distance , est une pratique courante depuis des années .

Fig. 6.38 Example of a translation from English to French of a sentence from the test data generated by the model (top) and the corresponding reference translation (bottom) (Sutskever et al. 2014)



Model	BLEU Value
best classical model	33,3%
single LSTM	30,6%
ensemble of 5 LSTMs	34,8%

Fig. 6.39 Quality of translations from English to French for different model types (Sutskever et al. 2014)

A measure is needed that can be used to assess translation quality by a numerical score. For this purpose, the so-called BLEU value compares the number of identical word sequences of length 1–4 in the generated translation with the corresponding numbers in one or more reference translations and aggregates this into a number between 0.0 and 1.0. The fraction of matching word sequences is used as a similarity measure. Further, a correction is made to give less weight to longer, repetitive translations. In evaluation experiments, BLEU correlated well with human ratings of translation quality.

At first glance, the translation results of the model (Sutskever et al. 2014) look very good. Figure 6.38 shows a translation generated by the model and the reference translation from the test data (Sutskever et al. 2014). As always, the model could not access this sentence during training.

Formal evaluation of translation quality using the BLEU value on the test data yielded the results in Fig. 6.39. While a single LSTM achieved a BLEU value of 30.6%, an ensemble of five LSTMs was able to attain a BLEU value of 34.8%. In such an ensemble (Sect. 5.11.2), several (here 5) models are trained independently with different initial values. In prognosis, the probabilities of the individual models for the output words are averaged at the end. This result showed that a neural translation model can do a better job than a mature classical model. The model was found to translate even long sentences of up to 80 words well. Because of the limited vocabulary, the model had problems with words that were not in the vocabulary. If this problem can be eliminated, the BLEU value increases further.

A useful property of the model is that it learns to represent a variable-length input sentence by a fixed length vector. Since there are always several translations which

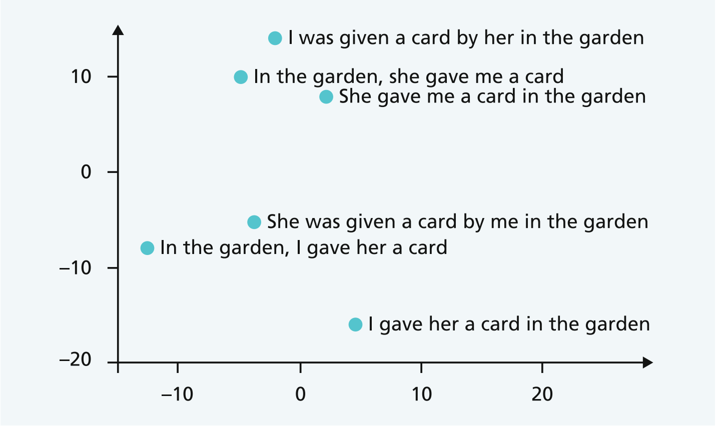


Fig. 6.40 Two-dimensional projection of different hidden vectors after processing the sentences by the encoder RNN. The embeddings of the sentences are automatically grouped according to their meaning (Sutskever et al. 2014)

can be understood as a paraphrase of the output sentence, the representation can be expected to reflect the meaning of a sentence.

Figure 6.40 shows the embeddings of different sentences in a two-dimensional projection. Depending on whether "I" or "she" is the subject of the sentence, the sentences lie in different clusters. The model recognizes that a passive formulation ("I was given a card by her") and an active formulation ("she gave me a card") express almost the same thing. The internal structure of the clusters is also similar. This shows that the sentence embeddings represent the meaning of the sentences well.

6.5.2 *Attention: Improving Translation by Recourse to Input Words*

The sequence-to-sequence model (Fig. 6.36) encodes the input sentence into a hidden vector, which is then used as the start vector for the decoder RNN. This works well for short and medium length sentences. However, for long sentences, the hidden vector becomes a bottleneck. Therefore, the idea came up (Bahdanau et al. 2015) to use the previously computed hidden vectors of the encoder RNN for translation. For an output word to be generated by the decoder RNN, one therefore needs a back reference to the relevant hidden vectors of the words of the input sentence.

In Fig. 6.41 this procedure is shown schematically. First, the encoder LSTM E encodes the input sentence into the hidden vectors $h_1^{enc}, \ldots, h_4^{enc}$. The decoder LSTM D receives h_4^{enc} as initial value and the sentence initial symbol *"BOS"* as

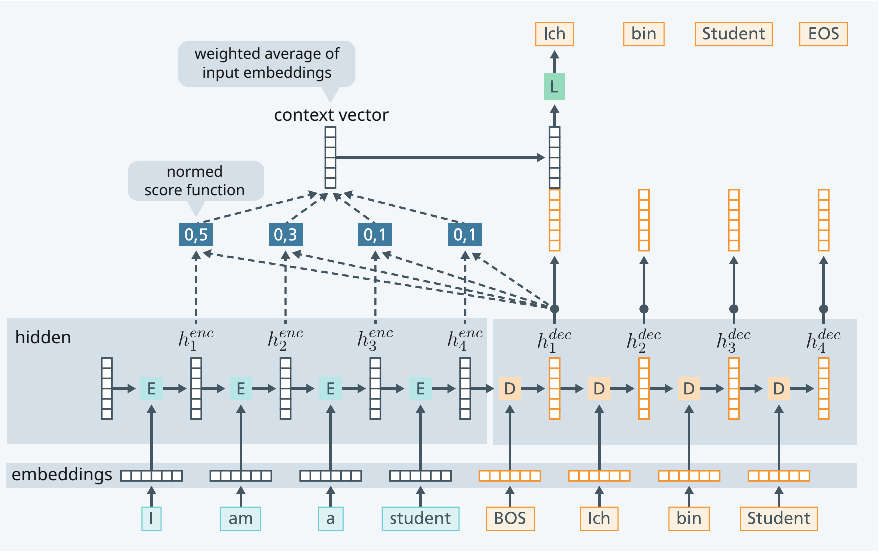


Fig. 6.41 Inclusion of encoder information by attention. First, the encoder ***E*** computes hidden vectors $h_1^{enc}, \ldots, h_4^{enc}$ of the inputs. The decoder ***D*** then computes the hidden vector h_1^{dec} from which to generate the probabilities for the first word (observed: "*Ich*"). Through the score function, normalized score values $score(h_t^{enc}, h_t^{dec})$ are determined for each of the hidden vectors $h_1^{enc}, \ldots, h_4^{enc}$ with h_1^{dec}. These "attention weights" indicate the relevance of each hidden vector h_t^{enc} of the input sequence for generating the next output word. The attention weights are used to compute a weighted average of the hidden input vectors $h_1^{enc}, \ldots, h_4^{enc}$, the context vector. The context vector is used together with h_t^{dec} to predict the output word ("*Ich*") by the logistic regression model L. This is repeated for all output words

input and generates the first hidden vector h_1^{dec}, which is to be used to predict the first word ("*Ich*"). Now, the similarity of the hidden vectors $h_1^{enc}, \ldots, h_4^{enc}$ with the hidden vector h_1^{dec} of the decoder RNN is computed. A score function is used for this purpose:

$$\text{score}(h_t^{enc}, h_1^{dec}) = a' * \tanh(W_1 h_t^{enc} + W_2 h_1^{dec})$$

The vector a and the matrices W_1 and W_2 are parameters which are optimized during the training. The results are normalized by a softmax function to a probability vector

$$(\alpha_1, \ldots, \alpha_T) = \text{softmax}\left[\text{score}(h_1^{enc}, h_1^{dec}), \ldots, \text{score}(h_4^{enc}, h_1^{dec})\right]$$

such that the so-called attention weights $(\alpha_1, \ldots, \alpha_T)$ are all greater than zero and have 1.0 as their sum. They can be viewed as the relative "attention" received by each hidden vector of the encoder from the target word ("*Ich*").

Next, one forms a weighted average of the hidden vectors of the input $\alpha_1 * h_1^{enc} + \cdots + \alpha_4 * h_4^{enc}$. This so-called context vector summarizes the information in the hidden vectors of the input, which can be used to predict the next word (*"Ich"*). The context vector is appended to the hidden vector h_1^{dec} and used by the logistic regression model L to predict the next word *"Ich"* in the target language. After the probability of *"Ich"* has been computed these calculations are repeated for all words of the output sentence. The generation of new words stops when the sentence end symbol *"EOS"* has been generated.

According to Wikipedia, the term attention refers to the allocation of (limited) consciousness resources to different objects. The strength of relationship between the hidden decoder vector h_1^{dec} and the hidden encoder vectors h_t^{enc} is therefore called attention. However, this bears only a remote resemblance to the human ability to focus, since we are dealing here with purely statistical correlations. Since two sentences are related to each other, this is also called cross-attention.

Again the model is trained as a whole. To illustrate the flow of information during the gradient computation, simply reverse the direction of all arrows in Fig. 6.41. The loss function sets the goal to increase the probability of correct words of the output sentence as much as possible. The error information is propagated back from the word probabilities to the context vectors and the hidden vectors h_t^{dec} of the decoder and the output embeddings. In the process, the parameters a, W_1, W_2 of the score function are also optimized to maximize the probabilities of correct words in the output sentence. The error information is then propagated back into the decoder RNN, the encoder RNN and the input embeddings. In this way the unknown parameters of the logistic model, the score function, the two RNNs, and the input and output embeddings are trained with the stochastic gradient descent.

6.5.3 Attention Generated Translation Results

Although the hidden vectors encode the complete sentence, they are particularly influenced by the input word just included. Therefore, one can also assign the hidden vectors to the respective input words. Figure 6.42 shows for an English sentence the attention weights that result from the translation into German. It is clear that the model was able to automatically reconstruct the semantic relation between the words of the languages to a large extent. The network had to derive this mapping from the higher-level goal of translating the input sentence into the target language sentence. It also becomes clear that this "soft" mapping is better than a "hard" one-to-one mapping because in many cases one word has a semantic relation to several others. Thus, phrases of different lengths can be mapped to each other.

There are a number of ways to improve this basic model of attention. The model presented in Fig. 6.41 has the disadvantage that when generating embeddings, the encoder cannot take into account the words in the input sentence that appear later in the sentence. For example, when generating h_1^{enc} from *"I"*, it is not known that *"am"* will appear as the next word. For this reason, a bidirectional LSTM (Sect. 6.4.4) is often employed, which independently generates hidden vectors $\overrightarrow{h_1}, \ldots, \overrightarrow{h_T}$ by its

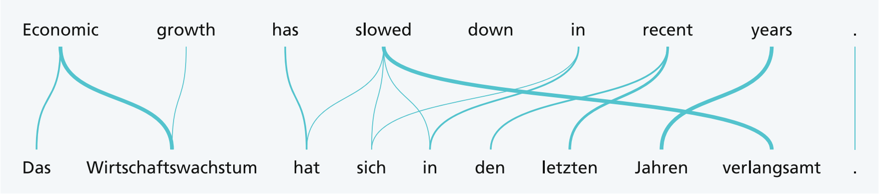
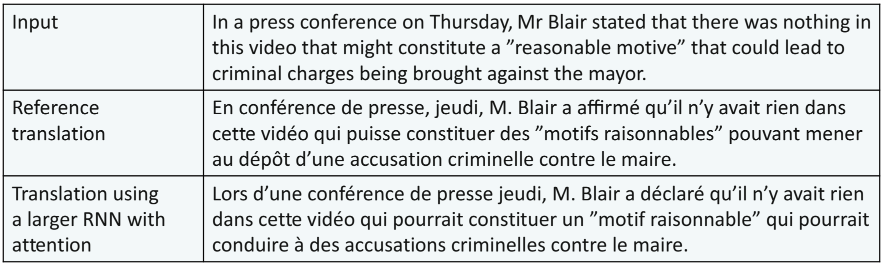


Fig. 6.42 Attention weights when translating an English sentence into German. The thickness of the connecting lines symbolizes the size of the attention weights. Image credits in Appendix A.3



Input	In a press conference on Thursday, Mr Blair stated that there was nothing in this video that might constitute a ”reasonable motive” that could lead to criminal charges being brought against the mayor.
Reference translation	En conférence de presse, jeudi, M. Blair a affirmé qu’il n’y avait rien dans cette vidéo qui puisse constituer des ”motifs raisonnables” pouvant mener au dépôt d’une accusation criminelle contre le maire.
Translation using a larger RNN with attention	Lors d’une conférence de presse jeudi, M. Blair a déclaré qu’il n’y avait rien dans cette vidéo qui pourrait constituer un ”motif raisonnable” qui pourrait conduire à des accusations criminelles contre le maire.

Fig. 6.43 A sentence from the test set (top) with reference translation (middle) and translation provided by the larger model with attention (bottom)

forward LSTM and $\overleftarrow{h}_T, \dots, \overleftarrow{h}_1$ with its backward LSTM. The score function is then applied to the combined hidden vectors $[\overrightarrow{h}_t, \overleftarrow{h}_{t+1}]$. This approach consistently yields better results than a simple LSTM.

Furthermore, several layers of LSTMs were used by Bahdanau et al. (2015) as encoders and decoders, with hidden vectors having a length of 1000. The model was trained by stochastic gradient descent with minibatches of 80 sentences. The WMT 2014 dataset for English-French with 850 million words was used as the training dataset. The training lasted for about 5 days. Figure 6.43 shows for illustration an English sentence from the test set, the reference translation provided by an interpreter, and the translation into French generated by the larger RNN model with Attention.

Figure 6.44 shows the translation quality measured with the BLEU value as a function of the length of the input sentence. The usual Sequence-to-Sequence models (RNN-Enc) and the models with Attention (RNN-Attn) were compared. Here, two models were trained, one with sentences up to a length of 30 words and the other with sentences up to a length of 50. The results show that the models with Attention perform better than the standard models when the sentence length is 8 or longer. The RNNAttn-50 model has about the same performance for all sentence lengths, which is also maintained for sentences up to length 60. This demonstrates that the use of Attention improves translations in a sustainable way. Even this early translation model RNN-Attn-50 performed slightly better than the then best

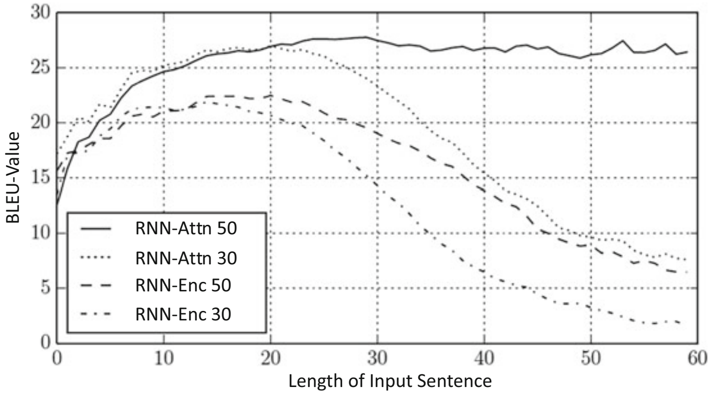
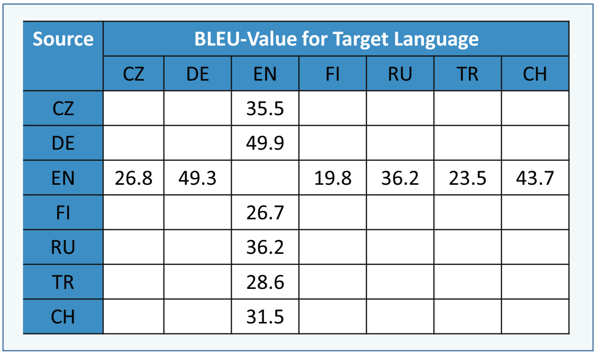


Fig. 6.44 Comparison of the BLEU values of different models depending on the length of the input sentence in a translation from English to French. The large attention model (RNN-Attn 50) performs better than the two standard models (RNN-Enc) and has consistent quality for all sentence lengths. Image credits in Appendix A.3



Source	BLEU-Value for Target Language						
	CZ	DE	EN	FI	RU	TR	CH
CZ			35.5				
DE			49.9				
EN	26.8	49.3		19.8	36.2	23.5	43.7
FI			26.7				
RU			36.2				
TR			28.6				
CH			31.5				

Fig. 6.45 Results of the WMT 2017 competition on the Newstest 2018 data for the languages Czech (CZ), German (DE), English (EN), Finnish (FI), Russian (RU), Turkish (TR), and Chinese (CH). The first column shows the source language, the second row the respective output languages. Image credits in Appendix A.3

conventional translation model Moses when the words of the sentence were present in the vocabulary.

In Fig. 6.45 the results of the WMT 2017 competition for different languages are shown (Bojar et al. 2017). Note that the training sets are different from the previous case. For languages with a complex grammar and diverse morphology, the results are not as good as for less complex languages such as English, Chinese, and Russian. An exception is German as a relatively complex language, but for which a lot of training data is available.

It turns out that the attention mechanism can sustainably improve the quality of translations, especially for long sentences. The model is no longer forced to encode

- embeddings/hidden vectors up to the length of 2048 bring a small increase in BLEU values.
- LSTM cells are consistently better than other RNNs.
- Bidirectional encoders with 2-4 layers are better than other combinations.
- Decoders with 4 layers are best.
- A well-tuned beam search with a width of 5-10 is crucial.

Fig. 6.46 Most important results of a comparative test of for RNN translation systems

all information into the hidden vector and store it for the whole sequence, but it is sufficient to update the essential information in the hidden output vectors and extract the details from the input sentence by the attention mechanism. For the architecture of RNN translation systems, Britz et al. (2017) performed a large-scale comparison and arrived at the results summarized in Fig. 6.46.

6.6 BERT: A Model for the Representation of Meanings

BERT is an acronym for **B**idirectional **E**ncoder **R**epresentations from **T**ransformer (Devlin et al. 2018). It is based on the Transformer proposed by Vaswani et al. (2017), which contains BERT as a component and is described later in Sect. 6.8. In each layer BERT computes the "correlation" of the embeddings of all word pairs of a text by "self-attention". These correlations are used to generate a new "context-sensitive" or contextual embedding for each word as the output of the layer. These embeddings depend on the surrounding text, and represent the meaning of a word much better than the static embeddings of Word2vec or an RNN. They are refined in several subsequent layers.

The resulting embeddings can be leveraged in further language processing applications. The advantage is that with a large, non-annotated training dataset, a general embedding model for words is obtained covering a rich knowledge about syntax and semantics. The model can then be trained for a specific task with relatively little training data, yielding higher accuracy. Such an approach is also called transfer learning (Sect. 5.7), since the knowledge gained in one training task is transferred to a different, but related task.

6.6.1 Tokenization to Limit Vocabulary Size

We have seen (Sect. 6.5.3) that the quality of translation suffers greatly when input words are not in the vocabulary. Vaswani et al. (2017) therefore use Byte Pair Encoding (BPE) to obtain a relatively small vocabulary of tokens consisting of all characters, frequent character sequences and frequent words. Typically, frequent words will be encoded as single tokens, while rare words will be encoded as a

Text	Vocabulary
a * r o s e * i s * a * r o s e *	a e i o r s t *
a* r o s e * i s * a* r o s e *	a a* e i o r s t *
a* ro s e * i s * a* ro s e *	a a* e i o r ro s t *
a* ro se * i s * a* ro se *	a a* e i o r ro s se t *
a* rose * i s * a* rose *	a a* e i o r ro rose s se t *

Fig. 6.47 Byte Pair Encoding. The first line shows the initial situation where the vocabulary consists of the letters and the end-of-word symbol “*”. In the next lines successively the most frequent pairs of tokens are connected to a new token and the vocabulary is extended accordingly. Links across the word end are not allowed

sequence of a few tokens, where these tokens represent meaningful word parts. This eliminates the problem of words that are not in the vocabulary.

The BPE vocabulary is built from the training data. Initially, all letters, digits and other characters of the training set are designated as tokens and included in the vocabulary. Each word initially consists of isolated letters and an end-of-word symbol (Fig. 6.47). Now all pairs of adjacent tokens are counted, and the most frequent pair is merged into a new token of the vocabulary. Merges beyond an end-of-word symbol are not allowed. This continues until the vocabulary reaches a predetermined number of tokens. As the algorithm proceeds, the most frequent words are represented as single tokens, while rarer words are composed of tokens. It is important that in an application new and unknown words can also be created from tokens, since all letters are contained in the vocabulary as well.

Sometimes the same word in BPE can be combined from different pairs of tokens. The WordPiece algorithm (Y. Wu et al. 2016) offers a mechanism to prioritize pair formation. The only difference between WordPiece and BPE is the way in which token pairs are added to the vocabulary. At each iterative step, WordPiece chooses a token pair, which will result in the largest increase in likelihood upon merging. BERT uses a WordPiece vocabulary of 30,000 tokens.

Position Encoding Static embeddings of length m of the input tokens are used as the input to the first layer of the Transformer. Position embeddings of length m for each token position $0, 1, 2, \ldots$ in the sentence are added to these embeddings and form the input embeddings for the model. Position encodings are required because the algorithm needs to take into account the positions of the tokens in the text. All input embeddings for the first layer in the following sections are the sum of word embeddings and position embeddings. Optionally, other embeddings such as section embeddings can be added (Fig. 6.51).

As an example, sine and cosine values of different frequencies are used as position encodings (Fig. 6.48). They have the advantage that arbitrary position differences can be calculated by linear functions. As an alternative, BERT uses trained position embeddings instead of trigonometric functions, because they gave better results.

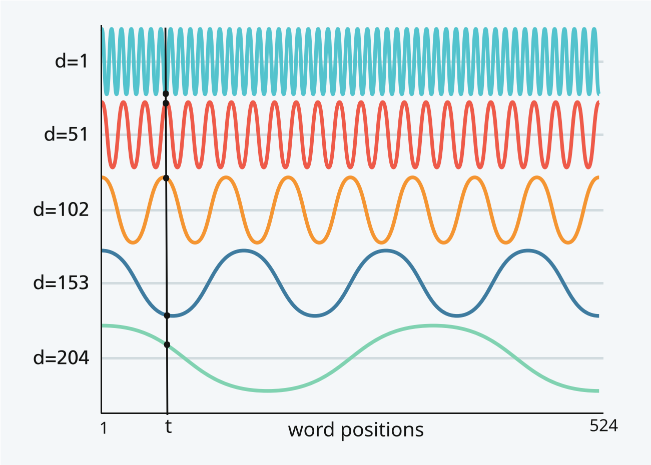


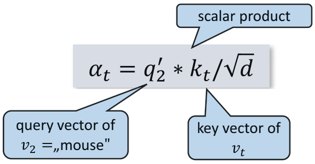
Fig. 6.48 The values of sine and cosine curves of different frequency define one type of position embeddings at different token positions. For an example position t the values of the position embeddings are marked by dots

6.6.2 Self-Attention Analyzing the "Correlation" of Different Tokens

The computation of self-attention (Vaswani et al. 2017) is shown in Fig. 6.49 for the text *"The mouse eats cheese"*. For the $T = 4$ input tokens $u_1, \ldots, u_T$ static input embeddings $x_1, \ldots, x_T$ are determined by lookup and adding position embeddings. The input embeddings x_t are multiplied by three matrices Q, K, and V, yielding three vectors: "query" (orange), "key" (light blue), and "value" (dark blue).

$$q_t = Q * x_t \qquad k_t = K * x_t \qquad v_t = V * x_t$$

Typically, the query vectors q_t and the key vectors k_t are of the same length, but shorter than the x_t embeddings. If a new embedding for the second token $v_2 =$ *"mouse"* with the current embedding x_2 is to be computed, the scalar product of the query vector q_2 of *"mouse"* with the key vectors k_t of the other tokens is formed to determine the "correlation" between the two embeddings



$$\alpha_t = q_2' * k_t/\sqrt{d}$$

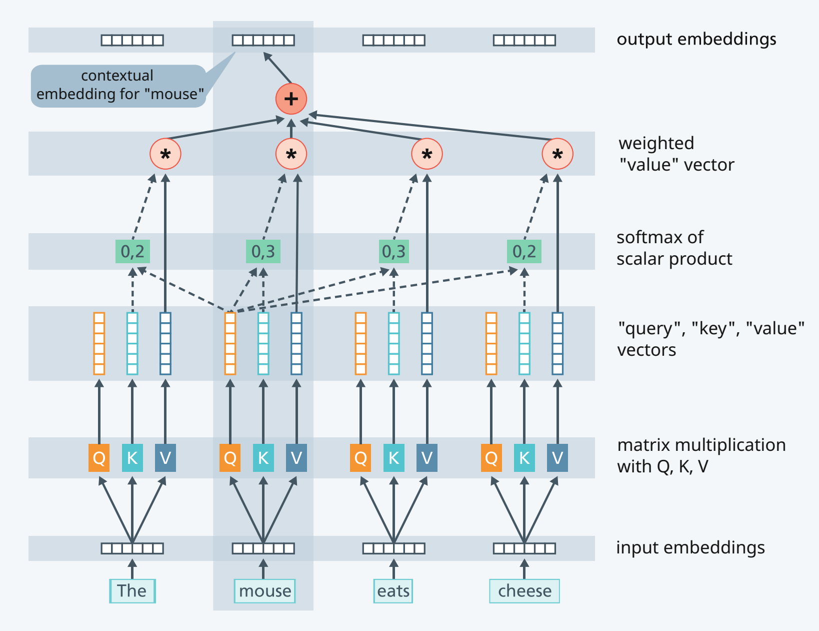
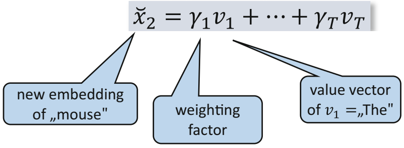

output embeddings
contextual embedding for "mouse"
+
weighted "value" vector
*
*
*
*
softmax of scalar product
0,2
0,3
0,3
0,2
"query", "key", "value" vectors
matrix multiplication with Q, K, V
Q
K
V
Q
K
V
Q
K
V
Q
K
V
input embeddings
The
mouse
eats
cheese

Fig. 6.49 Calculation of self-attention for the token *"mouse"*. Each of the tokens is represented by an input embedding. This input embedding is multiplied by the three matrices Q, K, V and results in three vectors "query" (orange), "key" (light blue) and "value" (dark blue). To compute a new embedding for the token *"mouse"*, the scalar product of the "query" of *"mouse"* and the "key" vectors of all tokens is formed and normalized by softmax. With these computed weights, a weighted average of all "value" vectors is formed, which is used as the new output embedding for *"mouse"*. The new context-sensitive embedding includes information from *"eats"*, which means that *"mouse"* has to be an animal. These calculations are repeated for all input tokens to generate an output embedding for each token

where $1/\sqrt{d}$ is a fixed normalization factor, and d is the length of the key vector. Note that the scalar product of two vectors of fixed length is maximal, if the two vectors are identical. Hence, a large value of the scalar indicates that both vectors are similar and have a large "correlation". The results of the computations are scalar values α_t, which are normalized to positive weights $(\gamma_1, \ldots, \gamma_T)$ with sum 1.0 using the softmax function.

$$(\gamma_1, \ldots, \gamma_T) = \text{softmax}(\alpha_1, \ldots, \alpha_T)$$

Then a weighted sum of the value vectors v_t with these weights is formed and the new embedding vector $\breve{x}_2$ is obtained:


$\breve{x}_2 = \gamma_1 v_1 + \cdots + \gamma_T v_T$
new embedding of „mouse"
weighting factor
value vector of v_1 =„The"

The new embedding $\breve{x}_2$ for “mouse” is **context-sensitive** or contextual, as it includes information from the other tokens *“The”*, *“eats”* and *“cheese”*. For example, because the *“eats”* token typically co-occurs with living creatures, the *“mouse”* embedding is changed to an animal embedding, as opposed to a computer device or a comic character. This property is crucial for the performance of self-attention layers, as the resulting embeddings very accurately capture the meaning of words.

This calculation is repeated in parallel for all tokens $v_1, \ldots, v_T$ of the input text yielding new embeddings $\breve{x}_1, \ldots, \breve{x}_T$. The matrices Q, K, V are parameters that must be learned. This type of attention differs from the RNN attention model of Fig. 6.41 in three aspects. First, the attention computations always refer to embeddings associated with individual tokens, rather than hidden vectors that combine information from different tokens. Second, the embeddings are multiplied by the matrices Q, K, V at the beginning, which gives more flexibility in the definition of “correlation”. Finally, to obtain more expressive embeddings, the self-attention is computed for all token pairs of the input text.

The computation of the dependency between distant tokens is done directly by forming the scalar products of their respective embeddings. In the case of an RNN, on the other hand, the prediction of the RNN must be executed very often one after the other in order to determine the relation between two widely separated tokens. This requires more computational effort. Moreover, by running the model multiple times, the different relations of different tokens are superimposed and become increasingly inaccurate. Therefore, self-attention has the potential to better capture the content relations of tokens.

To gain flexibility, not only one attention computation per layer is performed, but r simultaneous computations with different matrices Q_j, K_j, V_j, $j = 1, \ldots, r$, defining different “correlations” between embeddings. This is called multi-head attention. The length of the value vectors is then chosen as the length$(x_t)/r$. This allows to form a new embedding $\breve{x}_t$ from the r resulting weighted averages of the value vectors by concatenating them.

6.6.3 BERT Computes Contextual Embeddings by Self-Attention

The architecture of BERT is shown in Fig. 6.52. Its smallest version employs embedding vectors of size $m = 768$ and has 12 consecutive encoder blocks, each with $r = 12$ self-attention heads. The encoder block of the lowest layer receives

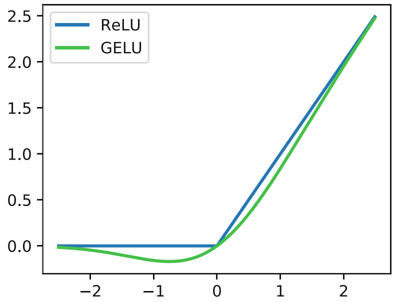


Fig. 6.50 The GELU activation function together with the ReLU activation. Image credits in Appendix A.3

as input embeddings the sum of the static embeddings of the input tokens, their position embeddings, and their section embeddings (explained later). Each encoder block has two operators.

The first operator is a self-attention layer, as described in Fig. 6.49. However, it is computed not only once, but $r = 12$ times in parallel in the same layer with different $Q_j, K_j, V_j, j = 1, \ldots, r$, matrices (multi-head attention). Consequently, the query, key, and value vectors and the generated embedding vectors $\breve{x}_t$ have length $768/12 = 64$. The twelve results are concatenated and yield again an embedding vector $\bar{\breve{x}}_t$ of length $m = 768$.

$$\bar{\breve{x}}_t = concat(\breve{x}_1, \ldots, \breve{x}_r) \tag{6.2}$$

The second operator is a variant of a fully connected layer (Vaswani et al. 2017, p. 5) with a GELU activation function (Hendrycks and Gimpel 2016, Fig. 6.50), which transforms the embeddings for each token separately.

$$\tilde{x}_t = GELU(\bar{\breve{x}}_t * W_1 + b_1) * W_2 + b_2 \tag{6.3}$$

The result of an encoder block is a new embedding $\tilde{x}_t$ for each token v_t of the input sequence. Around the operators, as with ResNet (Sect. 5.5.2), residual connections (bypass) are used, which simply copy the inputs upwards. This greatly simplifies the optimization. After each operator, a layer normalization standardizes the mean and variance similar to batch normalization (Sect. 4.6.7). The generated embeddings of the encoder are further processed by the next encoder block.

The input of BERT usually consists of two sections, where the first section starts with the marker [CLS] and ends with [SEP]. The second section is appended and also ends with the marker [SEP]. This second section is actually in half of the cases the follow-up text of the first section and a randomly selected section in the other half of the cases. The joint length of the sections is limited to 512 tokens. Further, a trainable embedding is added to the input to indicate if the token belongs to the first or the second section. Overall, the input embedding is the sum of a token's static embedding, its position embedding, and its section embedding. Figure 6.51 shows an example.

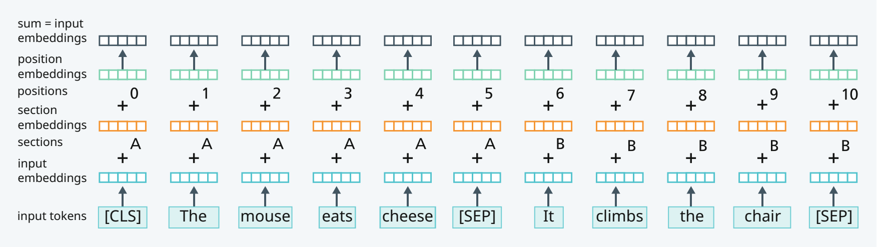


Fig. 6.51 BERT input embeddings: the token embedding (bottom), the section embedding (middle) for A=first section and B=second section and a position embedding (top) (Fig. 6.48) are added. The sum forms the BERT input. Image credits in Appendix A.3

6.6.4 BERT Prediction Tasks for Unsupervised Pre-training

The parameters of a BERT model (Fig. 6.52) include the static embeddings of the input tokens, the 512 position embeddings and the two section embeddings. In addition, the matrices $Q_{l,h}$, $K_{l,h}$, and $V_{l,h}$ of the different attention heads $h = 1, \ldots, r$ and layers l, the parameters of the fully connected feedforward operators, and the parameters of the logistic regression model are parameters of BERT. As usual, the parameters are initially set to random values. BERT simultaneously uses

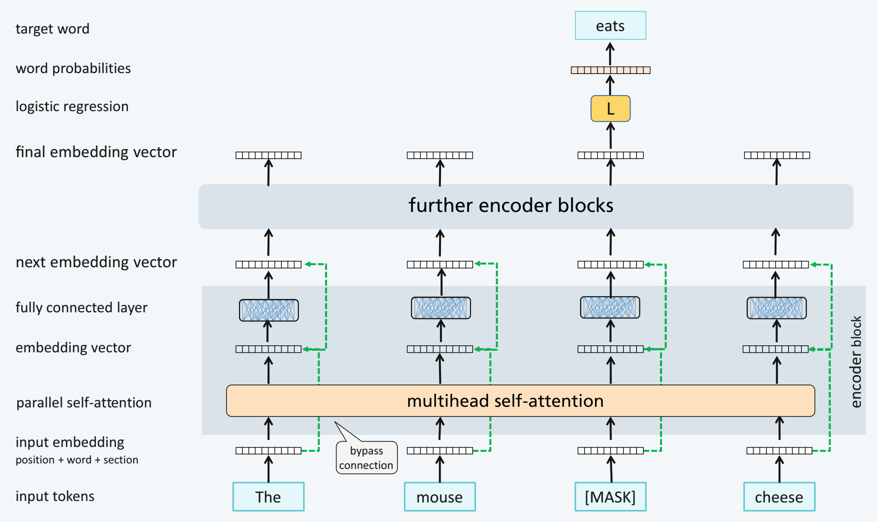


Fig. 6.52 A BERT encoder block consist of a multihead self-attention layer and a fully connected layer. Residual connections and layer normalization facilitate training. The output embeddings of a layer are used as input to the next layer. The embeddings of the final layer serve as input to a logistic regression classifier

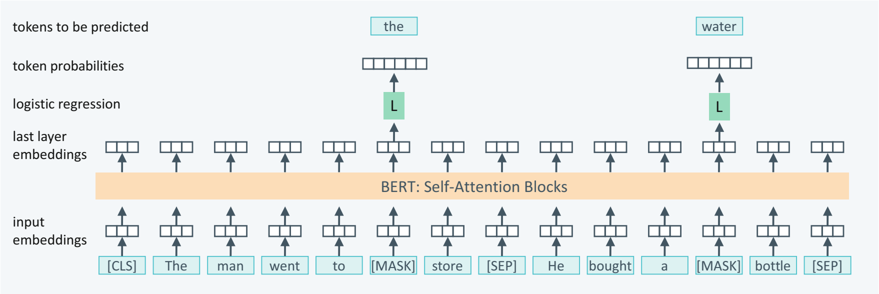


Fig. 6.53 Prediction task for unsupervised pre-training: the masked language model. The logistic regression model must predict the probability of the masked tokens. To do so, it uses the embeddings at the MASK position of the last layer as input

two prediction tasks to determine these parameters. They require a very detailed knowledge of the meaning of words and sentences in order to warrant embeddings with as much information as possible.

The first prediction task is called the masked language model (Devlin et al. 2018). For this, 15% random tokens in a sentence are selected as prediction tokens, 80% of the prediction tokens are replaced by the special token [MASK], 10% of the prediction tokens are replaced by a random token, and 10% of the prediction tokens are kept. The model has to predict the probability of the prediction tokens at each selected prediction position using logistic regression. An example can be found in Fig. 6.53. Here, the embeddings at the positions of [MASK] in the last layer of the encoder are used as input to the logistic regression model. This forces the encoder to gather all available information about these tokens from their environment and to include it into the embeddings for the prediction tokens.

For the second prediction task the training data is compiled such that each section with probability 0.5 is followed by the real next section after a separator [SEP], or else by a randomly selected section. During training the task is the classification of the next section as a true following section (next) or randomly selected section (nonext). Here, the embedding computed for the [CLS] token is used to predict the probability of "next" with logistic regression. Two examples are given in Fig. 6.54. The model is trained with the stochastic gradient descent optimization (Sect. 3.6.6), which includes the gradients for all tokens of a mini-batch.

In contrast to the LSTM language model, BERT has significant advantages:

- To predict a masked token the model can use information from token embeddings before and after the target token.
- The relationships to more distant tokens are modeled directly and do not need to be predicted via repeated application of the recurrent model.
- The model is forced to compile information about possible subsequent sections as a whole.

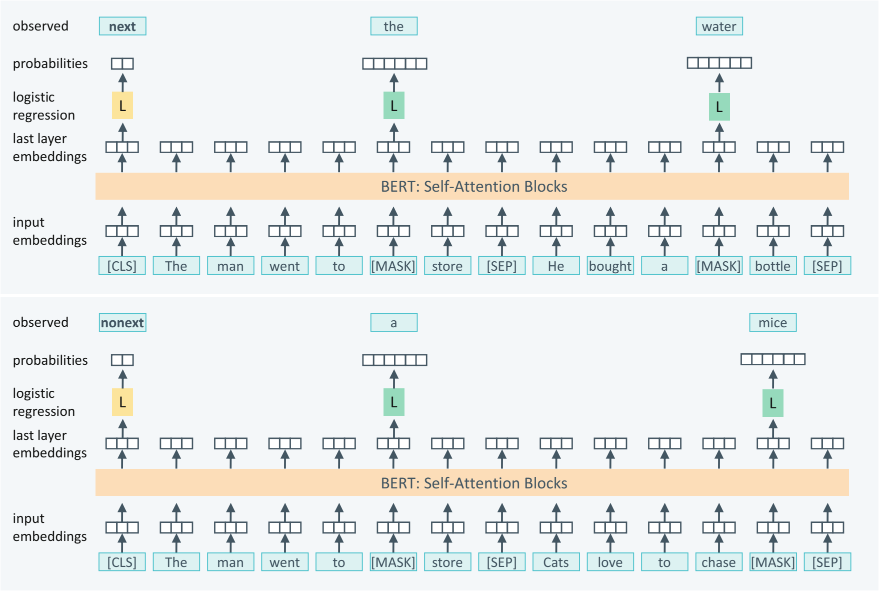


Fig. 6.54 Two examples of unsupervised pre-training for prediction of masked tokens and classification of the next section with two different logistic regression models. Using the embedding of [CLS] as input, the logistic regression classification model must predict the probability that the section after the first [SEP] is the successor of the previous section (next) or not (nonext)

The model was pre-trained on the BookCorpus (Zhu et al. 2015) with texts from books and on Wikipedia with a total of 3.3 billion words over 40 epochs. It should be noted that the data does not need to be annotated and the training is self-supervised. The “large” model has 24 Transformer blocks, hidden vectors of length 1024, and 12 parallel self-attention modules. It has a total of 345 million parameters and required four days of training time with 16 TPUs (very powerful GPU processors).

6.7 Transfer Learning with BERT Language Models

Transfer learning can be used to train models for tasks for which there is actually too little training data available (Sect. 5.7). For this purpose, the model is first pre-trained on a very large training data set similar to the final task data. During this pre-training the model learns basic relationships about the domain, e.g., about the syntax and semantics of the language. Subsequently, the model is fine-tuned with the actual small training data set, adapting the model to the specific task. Transfer learning has been discussed as early as the 1990s (Thrun 1998), but is only now taking off because of the vast increase in computing power. This section discusses transfer learning for the unsupervised pre-trained BERT model.

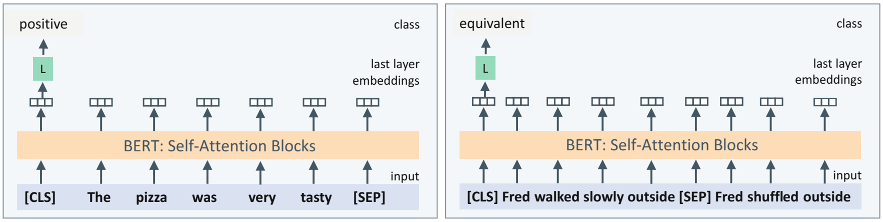
6.7.1 Semantic Classification Tasks

The pre-trained BERT model (Sect. 6.6) was fine-tuned for a series of classification tasks requiring a detailed understanding of the language. During fine-tuning, the pre-trained logistic regression models were omitted. Instead, the embedding of [CLS] was used as input to another logistic regression model that predicts the likelihood of each class (Fig. 6.55 top row). All parameters of BERT and the logistic regression model were adjusted to optimize the class probabilities on a relatively small, manually annotated fine-tuning dataset. Training for this fine-tuning required three or four epochs of the fine-tuning dataset (Devlin et al. 2018).

In Fig. 6.56 a typical task is shown. Two texts are given and BERT has to decide whether the statement of the second text is a consequence of the first text, forms a contradiction to the first text, or is neutral. The texts come from various sources (Williams et al. 2018).

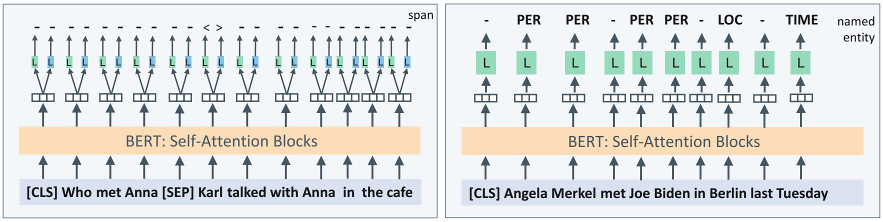
Figure 6.57 summarizes the results of five such semantic tests. The second column of the table provides a brief description of the task. The 3rd column lists the accuracy achieved by BERT on the test set, and the last column lists the best prior result. In all cases, BERT outperforms the accuracies of the prior systems, and in some cases is able to achieve considerable improvements in the state of the art.

BERT was also applied to the CoNLL 2003 benchmark for named entity recognition (Fig. 6.31). Here, an entity class had to be assigned to each token


positive
class
L
last layer embeddings
BERT: Self-Attention Blocks
input
[CLS] The pizza was very tasty [SEP]
equivalent
class
L
last layer embeddings
BERT: Self-Attention Blocks
input
[CLS] Fred walked slowly outside [SEP] Fred shuffled outside

Classification of a text

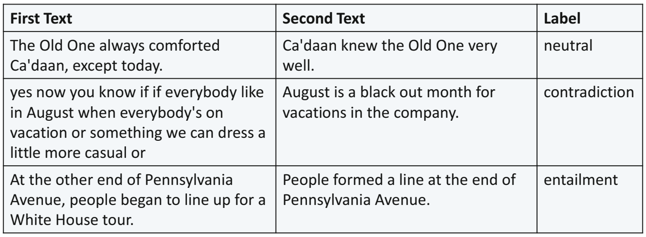
Classification of sentence pairs


span
< >
L
BERT: Self-Attention Blocks
[CLS] Who met Anna [SEP] Karl talked with Anna in the cafe
- PER PER - PER PER - LOC - TIME
named entity
L
BERT: Self-Attention Blocks
[CLS] Angela Merkel met Joe Biden in Berlin last Tuesday

Annotation of spans, e.g. question answering

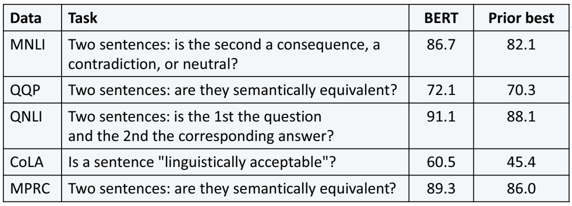
Annotation of word properties

Fig. 6.55 There are different ways to use BERT for semantic tasks. When classifying a text, the class probability for the CLS token is calculated (top left). The same procedure is used for the classification of text pairs (top right). When answering a question, the start and end of the answer is marked in the text by two different logistic regression models (bottom left). When annotating the words of a text, a class label is predicted for each word (bottom right)



First Text	Second Text	Label
The Old One always comforted Ca'daan, except today.	Ca'daan knew the Old One very well.	neutral
yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or	August is a black out month for vacations in the company.	contradiction
At the other end of Pennsylvania Avenue, people began to line up for a White House tour.	People formed a line at the end of Pennsylvania Avenue.	entailment

Fig. 6.56 Multi-genre Natural Language Inference (MultiNLI) consists of 433k sentences that are related to each other (Williams et al. 2018). They can be neutral, form a contradiction, or express an implication. The texts come from different sources, e.g., novels (top), telephone conversations (middle), and news reports (bottom)



Data	Task	BERT	Prior best
MNLI	Two sentences: is the second a consequence, a contradiction, or neutral?	86.7	82.1
QQP	Two sentences: are they semantically equivalent?	72.1	70.3
QNLI	Two sentences: is the 1st the question and the 2nd the corresponding answer?	91.1	88.1
CoLA	Is a sentence "linguistically acceptable"?	60.5	45.4
MPRC	Two sentences: are they semantically equivalent?	89.3	86.0

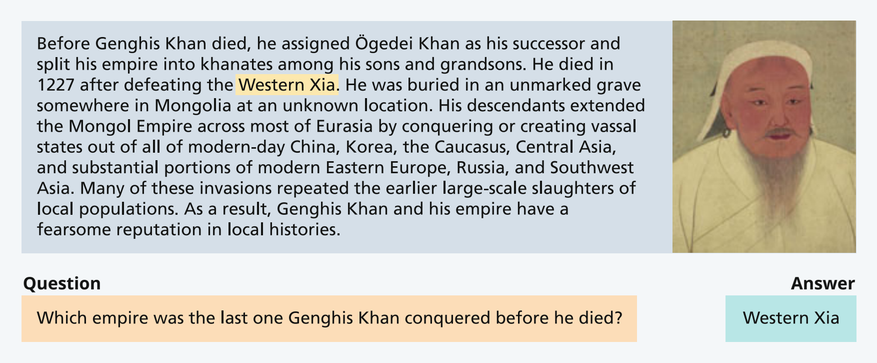
Fig. 6.57 The performance of BERT on different semantic classification tasks

(Fig. 6.55 bottom right). BERT achieved an F1 value of 92.4 on the test set, which is an improvement of 0.2 on the state-of-the-art value.

Sentiment analysis and opinion mining have the task of identifying feelings and opinions expressed in texts and classifying them into two or more classes. The Stanford Sentiment Treebank rates entire sentences as negative or positive. An example is the sentence *"This movie was actually neither that funny, nor super witty"*, which expresses a negative opinion even though the words funny and witty occur in the sentence. BERT was also able to perform this task better than previous methods, achieving 94.9% accuracy (Devlin et al. 2018).

6.7.2 Question Answering

The SQuAD1.1 (Stanford Question Answering Dataset) (Rajpurkar et al. 2016) benchmark tests whether the model has "understood" the content of a text and can answer questions about it. It consists of passages from articles of the English Wikipedia and questions about them. The answers must be found as a word sequence in the passage (Fig. 6.58). The question and the passage are entered into the BERT model as two texts. The BERT model then outputs an embedding vector x_t for each input word w_t. During fine-tuning with the SQuAD training data, two different logistic regression models are trained (Fig. 6.55 bottom left). The first one predicts



Before Genghis Khan died, he assigned Ögedei Khan as his successor and split his empire into khanates among his sons and grandsons. He died in 1227 after defeating the Western Xia. He was buried in an unmarked grave somewhere in Mongolia at an unknown location. His descendants extended the Mongol Empire across most of Eurasia by conquering or creating vassal states out of all of modern-day China, Korea, the Caucasus, Central Asia, and substantial portions of modern Eastern Europe, Russia, and Southwest Asia. Many of these invasions repeated the earlier large-scale slaughters of local populations. As a result, Genghis Khan and his empire have a fearsome reputation in local histories.

Question	**Answer**
Which empire was the last one Genghis Khan conquered before he died?	Western Xia

Fig. 6.58 The SQuAD benchmark consists of a passage from a Wikipedia article (top), a question (left), and an answer (right) taken from the passage (highlighted in yellow). Image credits in Appendix A.3

the probability for each token, that it is the beginning of the answer. The second predicts the probability that the token constitutes the end of the answer.

The original BERT model achieved an F1 value of 91.8%. Currently, a variant of BERT, the XLNet (Yang et al. 2019) is the best model and achieves an F1 value of 95.1%. The performance of human experts is 91.2%, so both BERT and XLNet outperform human accuracy. Overall, XLNet achieves significantly higher accuracy than BERT in most of 20 semantic tasks.

SQuAD2.0 makes the task of text comprehension more realistic. For the prior SQuAD version it was certain that the answer could be found in the passage. In addition to the 100,000 questions of SQuAD1.1, another 50,000 questions were formulated that are similar to the answerable questions but cannot be answered from the passage. Thus, the procedure must first decide whether the answer is contained in the passage at all. In this case it must mark the words of the answer. The best method is currently a modification of BERT. It achieves an F1 value of 92.4% (Lan et al. 2020). The F1 value of human experts is 89.5%. Thus, this model is better than human readers on the SQuAD2.0 benchmark as well.

What is important in all these approaches is that the model can be directly applied to other languages: no dictionaries, grammars, or knowledge bases are needed, but only a pre-training in the new language. Meanwhile, research focuses on solving more complex question-answering problems where the answer must be formulated from scratch, is based on large amounts of text, and requires world knowledge and logical reasoning. Overall, unsupervised pre-training followed by supervised fine-tuning has been shown to be highly successful in language processing.

Lee et al. (2020) have pre-trained BERT on a large collection of medical articles. By fine-tuning on specific training data, they determined extraction methods for specific applications. They achieved significant improvements for biomedical entity extraction (0.5% F1 improvement to 86%) and biomedical relation extraction (3.5% F1 improvement to 81.3%). Finally, they trained on answering biomedical questions, such as. *"What does mTOR stand for?"* with the answer *"mammalian target of rampamycin"*. Here they achieved an absolute F1 improvement of 9.6%.

6.7.3 Extraction of World Knowledge

In AI, world knowledge consists of facts about everyday life, such as *"onions are hot"* and *"water extinguishes fires"* (Fig. 6.59). Most people assume that everyone knows these facts. In order to act meaningfully, an Artificial Intelligence agent must also have access to such world knowledge.

Previous approaches in AI have attempted to manually encode world knowledge. A famous example is Cyc, a knowledge base (Lenat 1995) that has been continuously developed since 1984. It formulates world knowledge as facts and rules, and allows to derives logical conclusions from them. Such approaches have the great advantage that logically true conclusions can be derived. The disadvantage compared to deep neural networks is that new relationships cannot be learned automatically from data, but must be explicitly entered manually. In addition there is a problem with rules that are not always true but only in most cases.

Fig. 6.59 World knowledge includes basic, commonly known facts such as *"water extinguishes fire"*. The image shows how water is used to fight a wildfire in California. Image credits in Appendix A.3

On stage, a woman takes a seat at the piano. She	The woman is now blow drying the dog. The dog ...
a) sits on a bench as her sister plays with the doll.	a) is placed in the kennel next to a woman's feet.
b) smiles with someone as the music plays.	b) washes her face with the shampoo.
c) is in the crowd, watching the dancers.	c) walks into frame and walks towards the dog.
d) nervously sets her fingers on the keys.	d) tried to cut her face, so she is trying to do something very close to her

Fig. 6.60 Two tasks from the SWAG data set. The correct answer must be selected. The answer to the first question is (d) and (a) is the answer to the second question

The BERT model provides, for the first time, a way to extract world knowledge by unsupervised pre-training from a large text collection. For this purpose, one can for instance use SWAG (Situations With adversarial Generations), a dataset for testing world knowledge (Zellers et al. 2018). It consists of 113,000 questions about a given situation, each with four possible answers. Figure 6.60 shows two such questions and the possible answers. In a number of previous datasets of this type, it was found afterwards that there were unintended word patterns in the possible answers, which pointed to the correct answer. For this reason, the authors chose a complicated procedure to rule out such effects. In total, advanced language models were used to construct four possible responses that matched the situation well. One of the answers was true for the situation, the others were certainly not true.

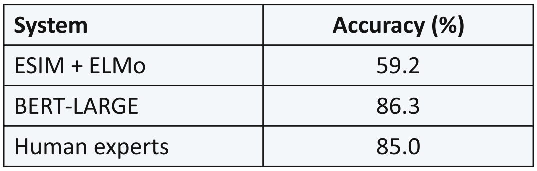
Devlin et al. (2018) have addressed this task using the pre-trained "big" BERT model (Sect. 6.6). The SWAG data was partitioned into a training set of 73,000, a validation set of 20,000, and a test set of 20,000 examples. On the training set, fine-tuning of the pre-trained model was performed. In each case, the situation description was associated with a response, e.g. *"[CLS] On stage, a woman takes a seat at the piano. [SEP] She sits on a bench as her sister plays with the doll. [SEP]"* (Fig. 6.61). The *"[CLS]"* output embedding of the BERT model is then input to a logistic regression model that predicts the probability of the four alternatives (Fig. 6.55 top right). This fine-tuning was done for three epochs. Here, the model must be able to say something about a scene as in Fig. 6.61.

The results are summarized in Fig. 6.62. While the best previous model on the test data had an accuracy of 59.2%, BERT achieved an accuracy of 86.3%. This is 1.3% better than the accuracy of human experts. This demonstrates that DNNs are in some cases better at reproducing linguistic world knowledge than humans in the domains studied for this task.

Importantly, BERT can also access information from the large training set, the BookCorpus and Wikipedia, with a total of 3.3 billion words. This contains considerably more information than the relatively small training set with 73,000 question-answer examples, which were used for fine-tuning. With the help of transfer learning, it is thus possible to extract knowledge from large text collections without supervision and make it queryable.

Petroni et al. (2019) show that BERT can extract facts and relations (e.g., birthplace of a person) as well and sometimes better than previous supervised learning methods. However, this is not true for all relations in which multiple objects

Fig. 6.61 On stage, a woman takes her seat at the piano. The American pianist Robin Spielberg. This scene matches a task in the SWAG dataset. Image credits in Appendix A.3



System	Accuracy (%)
ESIM + ELMo	59.2
BERT-LARGE	86.3
Human experts	85.0

Fig. 6.62 Accuracy of different algorithms on SWAG's test data. Source: (Devlin et al. 2018)

are involved at the same time (N-by-M relations). An application of the BERT approach can be found in the Aristo system (Clark et al. 2019). The task of this system is to answer multiple-choice questions from the "Grade 8 New York Regents Science Exam" for 8th grade students (Fig. 2.10). The system uses three types of subsystems: for information retrieval (i.e., intelligent searching of documents), for logical reasoning, and language models for predicting missing words and phrases.

The search in documents takes place on the one hand via classical search engines, but on the other hand also according to the correspondence of words and n-grams. The subsystems for logical reasoning uses, among other things, knowledge graphs that connect relevant phrases by content relations. The approach employs AristoBERT, a variant of BERT, to answer questions analogous to the scheme of Fig. 6.54. This is also trained with fine-tuning for the task. Each module outputs a score for confidence calibrated by a logistic model. These scores are then used in selecting the answer.

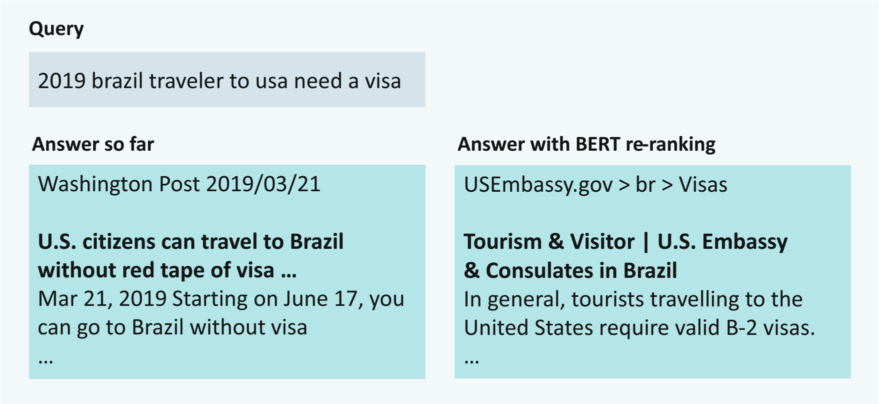
It turned out that the BERT modules achieved the highest answering accuracy and the other modules made only marginal contributions. Overall, the system could correctly answer 91.6% of eighth grade questions and 83.5% of twelfth grade questions (Clark et al. 2019). Although Aristo answers only multiple-choice questions without diagrams and covers only the domain of science, it represents an important milestone for systems that can capture content.

For many years, thinking in symbolic AI was considered as the discrete symbolic manipulation of statements expressed in a formal language. With the arrival of deep learning, this metaphor has changed. Algorithms can solve intellectually challenging tasks in an “associative” way using DNN rather than symbolic representational languages. This indicates that the machine has actually “learned” something about language and the world, and has mechanisms by which it can combine and exploit this knowledge.

6.7.4 Using BERT for Web Search

The first generation of search algorithms for web pages on the Internet identified documents which contain the words of the search query, using word frequency and the internet link structure to rank document. The word order of the search query and its meaning were ignored. The query *“2019 brazil traveler to usa need a visa”* had so far also resulted in Google hits of the type *“U.S. citizens can travel to Brazil without the red tape of a visa . . .* ” (Nayak 2019), because the search algorithm did not recognize the direction of travel (*“to usa*”) from the question (Fig. 6.63). Google now announces in its blog that such subtleties of queries are taken into account by the search engine with the help of BERT.

In the simplest case, a web search is performed in three steps. First, a large number (e.g. 1000) of possibly relevant documents are gathered via the separate string comparison of the search terms. Then, these documents are evaluated by a ranking algorithm in terms of their relevance. Finally, the top ten of these documents are presented to the user. Now, the BERT algorithm is used in Google search for ranking. As input BERT get the query as well as a retrieved paragraph and estimates



Query

2019 brazil traveler to usa need a visa

Answer so far

Washington Post 2019/03/21

U.S. citizens can travel to Brazil without red tape of visa …
Mar 21, 2019 Starting on June 17, you can go to Brazil without visa
…

Answer with BERT re-ranking

USEmbassy.gov > br > Visas

Tourism & Visitor | U.S. Embassy & Consulates in Brazil
In general, tourists travelling to the United States require valid B-2 visas.
…

Fig. 6.63 In a web query, the previous search algorithm cannot correctly assign the meaning of the conjunction *“to”* (left). This is possible with the new BERT ranking method

the probability, that the paragraph is the answer to the query. The probability then can be used as ranking score. This criterion is trained using the MS-MARCO dataset for ranking responses. The quality of such a ranking method is measured by the mean reciprocal rank of the best ten answers found (MRR@10). While the best classical methods on the test data were at an MRR@10 of 28.1, BERT-based approaches now achieve an MRR@10 of 39.3. This means much better hits for web searches, although the algorithm still makes individual errors (Nayak 2019).

6.8 Transformer Translation Models

The attention mechanism has been found to be very effective for improving translation with a RNN. However, it has problems with very long input texts and requires a significant amount of computation for training. As an alternative, the contextual embeddings created with BERT's self-attention can be used to represent rich information on the semantic and syntactic properties of a word. This approach is used by the transformer to improve automatic translation.

The transformer translation model (Vaswani et al. 2017) does not encode the input text into a hidden vector, but instead performs a large number of attention computations. In a first encoder module, the "correlation" of the tokens of the source sentence with each other is evaluated using BERT self-attention encoder layers to generate contextual embeddings for each token. Within a second decoder module, a "cross-attention" layer is used to establish the relationship between the embeddings of tokens of the source sentence and the embeddings of tokens of the translated output sentence. The model requires much less training time than RNNs and has a high potential for improving the quality of translations.

Historically the Transformer model was developed in 2017 before BERT. The BERT model is just the encoder part of the transformer and constructs very informative token embeddings. In Sect. 9.2.3 we describe the GPT language model, which is just the decoder part of the transformer and can generate text of unprecedented quality.

6.8.1 Cross-Attention Exploits the Input-Output "Correlation"

The transformer consist of two modules: the encoder and the decoder. The encoder uses several layers of self-attention to generate expressive context-sensitive embeddings of the input text tokens. The encoder architecture is identical to the BERT encoder (Fig. 6.52), except that ReLU activations are used in the fully connected layers (6.3). The embeddings of the last encoder layer serve as input to the decoder, which generates embeddings in several layers one after the other. These embeddings are finally used to predict the tokens of the translation.

To relate the input text with token $v_1, \ldots, v_T$ to the tokens $u_1, \ldots, u_R$ of the translation, the transformer decoder uses cross-attention. Similar to an RNN, the translation is generated iteratively token by token. Assume the tokens $u_1, \ldots, u_t$ have already been generated. In a first step contextual embeddings of $u_1, \ldots, u_t$ are computed using self-attention. Then cross-attention matches these embeddings of $u_1, \ldots, u_t$ with the embeddings of each input text token $v_1, \ldots, v_T$ from the last encoder layer. The calculations are performed analogously to the self-attention in Fig. 6.49, with the difference that the query vectors are calculated from the embeddings of the output sequence.

Figure 6.64 illustrates the procedure to compute the new embedding for the output token *"Maus"* by cross-attention. First a self-attention is computed for the already translated tokens *"BOS Die Maus"* yielding new contextual embeddings. Then the current embedding of *"Maus"* is multiplied by the Q matrix to obtain the query vector q_3. In parallel, the embeddings $\tilde{x}_1, \ldots, \tilde{x}_T$ in the highest encoder layer of the input sequence are multiplied by the K and V matrices to compute the key and the value vectors, respectively. Subsequently, a scalar product of the key vectors

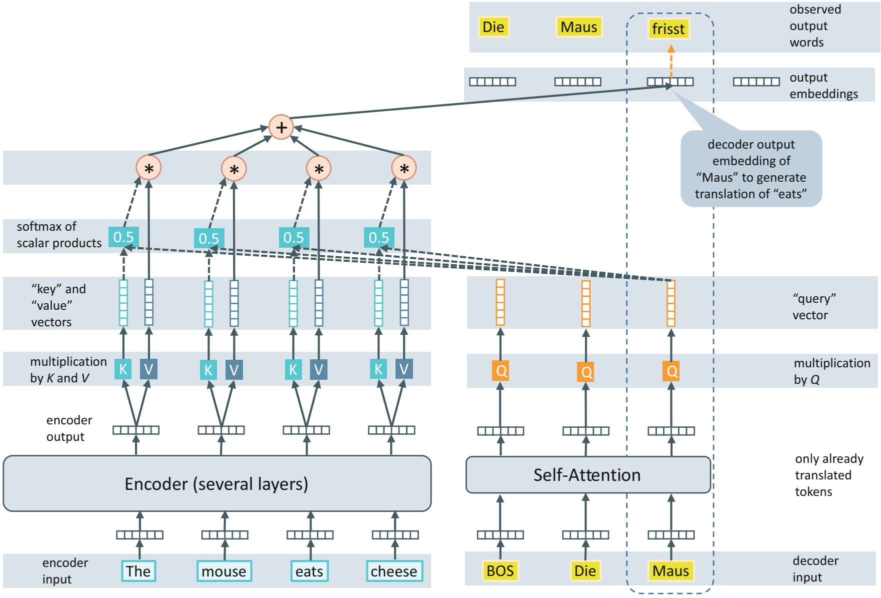


Fig. 6.64 The Transformer cross-attention operates on the already translated tokens *"BOS Die Maus"*. As input it uses the encoder embeddings of the top encoder layer of the input sequence. The computations are analogous to Fig. 6.49 except that the query vector (orange) is computed from the embedding of the translated tokens (e.g. *"Maus"*). This generates new embeddings for the translated tokens. After several layers the embedding of the last token *"Maus"* is used to generate the probabilities for the next token of the translation (observed: *"frisst"*)

and the query vector q_3 is formed

$$\alpha_t = q_3' * k_t/\sqrt{d}$$

and the weights $(\gamma_1, \ldots, \gamma_T)$ are calculated by normalizing the α_t with softmax. These weights indicate the "correlation" of the input tokens with the embedding of the output token *"Maus"* and are multiplied by the value vectors, resulting in the new embedding for the token *"Maus"*.

Then the results from different attention heads are concatenated (6.2) yielding new embeddings $\bar{\bar{u}}_1, \ldots, \bar{\bar{u}}_t$ for all actual tokens $u_1, \ldots, u_t$ of the translation. Note that the embeddings of the remaining tokens $u_{t+1}, \ldots, u_R$ of the translation after *"Maus"* are not used in the computation. Therefore, the operation is called masked cross-attention.

These computations are repeated with different decoder layers. As explained later, the top layer embedding of *"Maus"* is used to generate the probabilities of the next token of the translation by logistic regression. In our example the token *"frisst"* was observed.

6.8.2 Transformer Architecture Uses Self- and Cross-Attention

The Transformer (Vaswani et al. 2017) is a DNN for translating a text from an input language (e.g. English) to the target language (e.g. German). It consists of six consecutive BERT encoder blocks (Fig. 6.52) and then six consecutive Transformer decoder blocks. The model uses embeddings of length 512. Figure 6.65 shows its architecture.

The decoder uses the embeddings of source tokens generated by the last layer of the encoder to successively compute the probabilities of the output tokens in the target language. Thus, to translate *"The mouse eats cheese"* and generate the output tokens *"Die Maus frisst Käse EOS"*, the encoder (left side of Fig. 6.65) must be run once. The decoder (right side of Fig. 6.65) must be run five times to successively compute the probabilities of the tokens *"Die"*, *"Maus"*, *"frisst"*, *"Käse"*, and *"EOS"*.

The decoder of the lowest layer receives the static embeddings of the already translated tokens $u_1, \ldots, u_t$, e.g. *"BOS Die Maus"*, and embeddings of their positions as input. A mask is used to ensure that only already generated tokens and their embeddings are considered. This trick makes the model autoregressive and causal, i.e., the output tokens probability does not depend on the "future" output tokens. Each decoder block contains three operators. The first operator is an 8-head self-attention (Fig. 6.49) computing context-sensitive embeddings of the already translated tokens in the same way as in the encoder but with different Q, K, and V matrices.

A second operator computes an 8-head cross-attention connected to the input embeddings $\tilde{x}_1, \ldots, \tilde{x}_T$ of the top layer of the encoder (Fig. 6.64). It proceeds similarly to the self-attention in Fig. 6.49, except that the query vector is computed

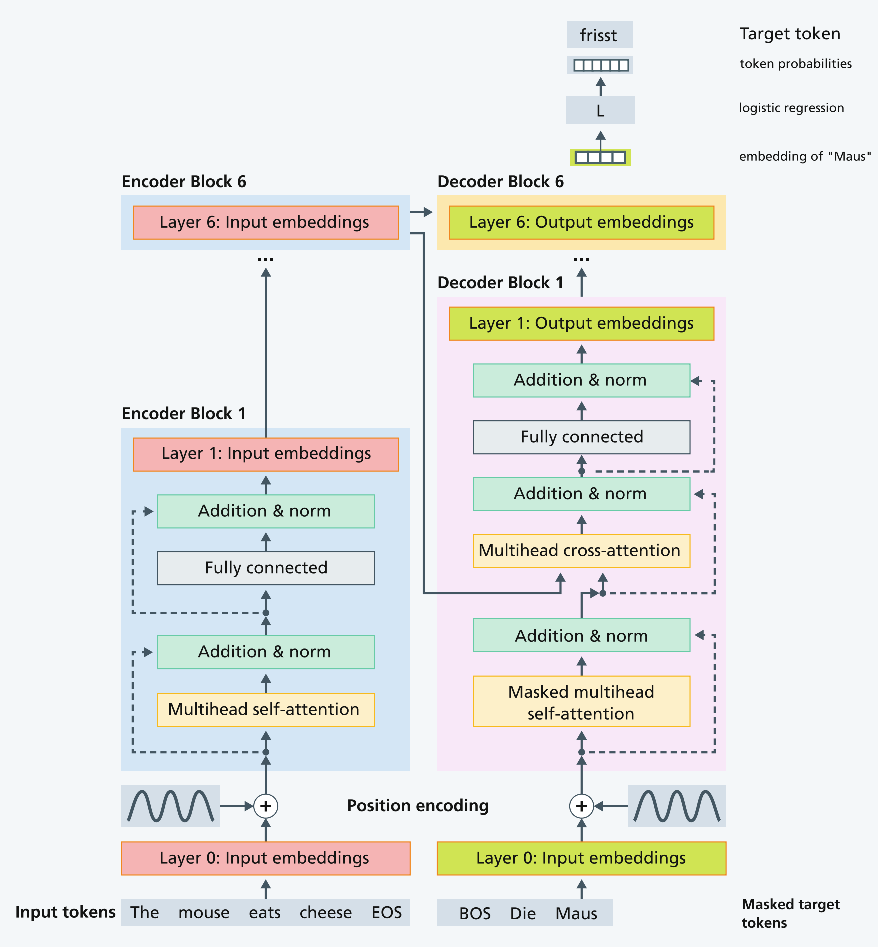


Fig. 6.65 To translate a text from one language to another the Transformer uses six encoder blocks (left) and six decoder blocks (right). The last layer of the encoder generates contextual embeddings of the input tokens. Each decoder block applies a self-attention layer to the already translated tokens and a cross-attention layer relating the translated tokens to the top layer encoder embeddings, as well as a fully connected layer generating new embeddings (light green). From the top layer embedding of the last translated token (“*Maus*”) a logistic regression model calculates the probabilities for the output tokens. The encoder and decoder contain residual connections (dashed) and layer normalization

from the decoder embeddings of $u_1, \ldots, u_t$, and key and value vectors are computed from the embeddings of the last layer of the encoder. The result is a modified output embedding for $\breve{\tilde{u}}_1, \ldots, \breve{\tilde{u}}_t$ including information from the input text.

The third operator is a fully connected layer with rectified linear ReLU activation (6.3) that transforms the embeddings for each token. Result of each layer is a new embedding for $u_1, \ldots, u_t$.

Around the operators, as with ResNet (Sect. 5.5.2), residual connections (bypass) are used, which simply copy the inputs upwards. This greatly simplifies the optimization. After each operator, a layer normalization standardizes the average and variance similar to batch normalization (Sect. 4.6.7). The generated embeddings of the encoder are further processed by the next encoder block.

From the calculated output embeddings of the last (sixth) Transformer layer for $u_1, \ldots, u_t$, a logistic regression with the embedding of u_t as input computes the probabilities of the next output token u_{t+1} of the translation, which is *"frisst"* in our example. Note that the translation of *"eats"* depends on the subject of the sentence. If the subject is a human, then *"isst"* is the proper translation of *"eats"*; for an animal, *"frisst"* is correct. This information is encoded in the embedding of *"Maus"* by the cross-attention computations.

6.8.3 Training the Transformer for Language Translation

It is instructive to realize the information flow when training the Transformer with just two encoder and decoder layers each (Fig. 6.66). During training the model gets a source text (e.g. *"The mouse eats cheese EOS"*) and its translation (e.g. *"BOS Die Maus frisst Käse"*) as input. The goal of the training is to adjust the parameters so that the observed target tokens (translation) are generated with maximum probability. As always, the parameters are set to random values at the beginning. For the forward step, static token embeddings are selected for the tokens of the source text, resulting in source embeddings 0 after adding the position embeddings. Then, by applying the encoder blocks, the source embeddings 1 and source embeddings 2 are computed.

The target tokens are then generated successively for the individual positions. At the beginning we only have the prior target token *"BOS"* (Begin of Sentence). For this input, the corresponding static target embedding 0 is selected by lookup (Fig. 6.66 left). It is propagated through the decoder blocks and target embedding 1 and target embedding 2 are computed. In this process, cross-attention is used to include information from the last embedding layer of the encoder, source embedding 2. Finally, the logistic regression model calculates the probabilities of the first observed target token (observed: *"Die"*), using the embedding of *"BOS"* as input.

To predict the probabilities for the second target token (observed: *"Maus"*) the input of the decoder contains the observed prior source tokens *"BOS"* and *"Die"*. Again the static target embedding 0 is selected by lookup (Fig. 6.66 right). The tokens *"BOS"* and *"Die"* are propagated through the decoder blocks including information from the encoder embeddings to produce target embedding 1 and target embedding 2. Finally, the logistic regression model estimates the probability of the next target token *"Maus"* using the computed target embedding of *"Die"* as input. The probabilities for the remaining tokens of the translation are generated in the same way.

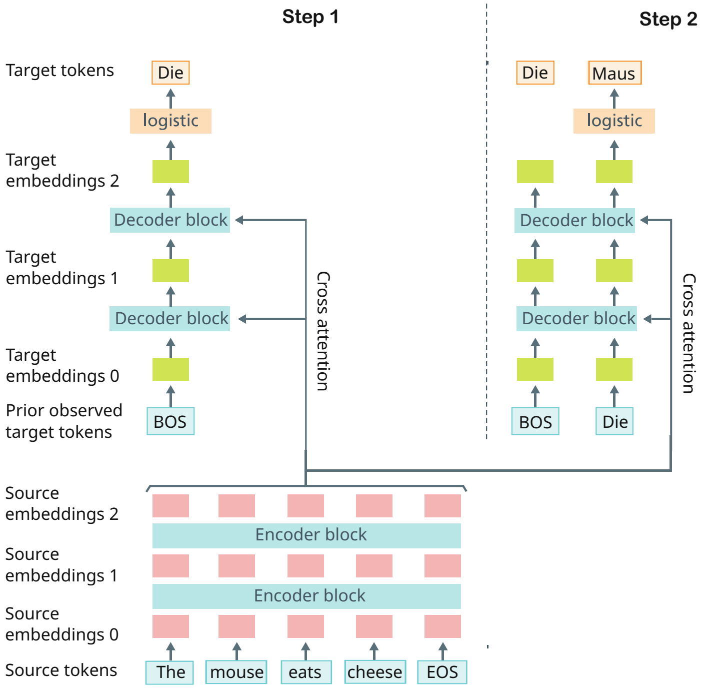


Fig. 6.66 Generation of the tokens of the translation by the Transformer. First, the encoder blocks generate the source embeddings (pink) of the top encoder layer. To determine the probability of the first target token (left graph), a target embedding of the begin-of-sentence token *"BOS"* is used as input. Then, two decoder blocks and the logistic regression model are employed to generate the probability of the first target token (observed: *"Die"*). In this process, the information from the source text is transferred by cross-attention. The computation of the probabilities for the second target token (right graph, observed: *"Maus"*) is performed analogously with the prior observed target tokens *"BOS Die"* as input

The model has a large number of parameters: the encoder and decoder matrices Q_j, K_j, V_j of the different heads and layers, the parameters of the fully connected layers, and the static input token embeddings of the source and target tokens. In total, the model has 213 million parameters. The log-likelihood loss function (Sect. 3.5.2) is used here, as is standard practice for logistic regression.

For gradient calculation during training, the computation sequence shown in (Fig. 6.66) is run in the reverse direction. The error signal is generated by the logistic regression model, which determines the probability of the observed tokens (e.g. *"Die"*). It then traverses the decoder blocks sequentially from top to bottom, providing gradients for their parameters. Importantly, the information flow splits into the 8 attention heads and modifies the respective matrices Q_j, K_j,and V_j. Via cross-attention the error signal reaches the top layer of the encoder.

Then the error signal is propagated in the same way through the encoder blocks from top to bottom and provides gradients for their parameters. This gradient computation is repeated for the other target tokens. Consequently, the encoder, the decoder and the logistic regression model as well as the static source and target embeddings are trained simultaneously with the stochastic gradient descent optimization (Sect. 3.6.6), which aggregates the gradients for all tokens of a mini-batch.

The Transformer model was trained by Vaswani et al. (2017) on two training datasets: the English-German WMT 2014 translation data with 4.5 million English sentences with German translations and the English-French WMT 2014 translation data with 36 million English sentences with French translations. The optimization used 8 GPUs and took 3.5 days to train the large model.

6.8.4 Translation Results for the Transformer Model

To produce a translation with the trained model, the encoder generates contextual embeddings of the source text. Then the token probabilities of the first token of the translation are computed. As described in Sect. 6.5.1 either the token with the highest probability is selected or several tokens with the highest probability are generated during a BEAM search. The generated token is appended to the input of the Transformer decoder and the probability of the following token is predicted. In this way the tokens of the translation are created one after the other.

For the WMT 2014 data set for English to German translation, the Transformer model achieved the best score at the time of publication with a BLEU value of 28.4 (Fig. 6.67). In addition, the training costs were only a fraction of the other models. For the WMT 2014 data set for English to French translation, the Transformer model was also better than all other models with a BLEU value of 41.8. Again, the training times were only 25% of the time required by other models. Meanwhile, with an extension of the Transformer model (with back-translation), a BLEU score of 45.6 was obtained.

It is instructive to illustrate the self-attention results. Figure 6.68 shows, for an English sentence, the activations for two different self-attention operators in the fifth-layer. It can be seen that the two operators cover different interpretative

	BLEU		**FLOPS**	
Model	**EN-DE**	**EN-FR**	**EN-DE**	**EN-FR**
Deep Att (Zhou et al. 2016)		40,6		$8 * 10^{20}$
ConvS2S (Gehring et al. 2017)	26,4	41,3	$8 * 10^{19}$	$1 * 10^{21}$
Transformer (Vaswani et al. 2017)	28,4	41,8	$2.3 * 10^{19}$	

Fig. 6.67 Results for the large Transformer model compared with the RNN Attention model and the Convolutional Sequence-to-Sequence model. FLOPS are the number of floating point operations to train. The training cost of the Transformer is only a fraction of the alternative models (Gehring et al. 2017; Zhou et al. 2016)

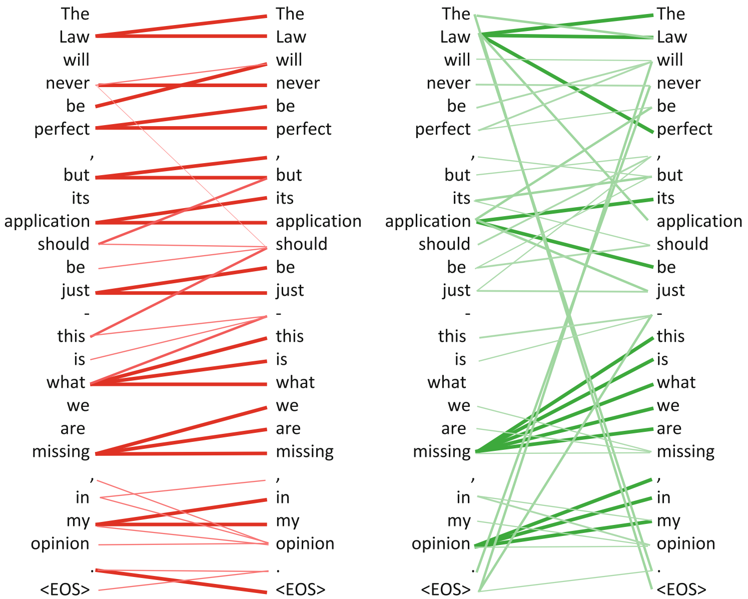


Fig. 6.68 Display of the attention weights on the 5th layer of the Transformer model for two complementary self-attentions heads. The size of the weights is expressed by the thickness of the lines

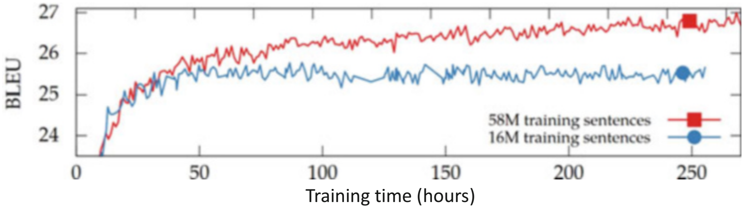


Fig. 6.69 The Transformer network with a large training set achieves a better translation quality than a model trained on less data Popel and Bojar (2018). Image credits in Appendix A.3

aspects. For example, on the left-hand side *"perfect"* is associated with *"be"* and *"perfect"*, while on the right-hand side *"be"* and *"perfect"* are associated with *"law"*. Apparently, the model has learned what *"perfect"* refers to.

Popel and Bojar (2018) examined how to improve the optimization of the Transformer model. As Fig. 6.69 shows, it is apparently better to use many training examples instead of few. They also show that large models with longer hidden vectors and a higher number of multi-head attention modules per layer are more efficient, but at the cost of higher computation time requirements.

Researchers at Microsoft announced (Hassan et al. 2018) that their machine translation system had succeeded in translating a text from Chinese into English with the same accuracy as a human. External bilingual experts were consulted to evaluate the translation quality. Even Microsoft did not expect this breakthrough in machine translation so soon.

The basis for the Microsoft system was a Transformer model with additional improvements. The system follows the strategy of dual learning: Translation models from Chinese into English and from English into Chinese are trained simultaneously. By this back-translation one gains an additional error signal, since the second translation should be equal to the source text. Thus, additional monolingual texts can be used for training translation models.

6.8.5 Simultaneous Translation Requires a Time Delay

In many situations, e.g. conferences and lectures, it is important that the translation becomes available almost simultaneously with the speaker's words (Fig. 6.70). Otherwise, listeners cannot adequately interpret the speaker's facial expressions and gestures. However, this simultaneous translation is very difficult for many language pairs because of the different word order of the languages. As an example, Fig. 6.71 shows a German sentence and its translation into English. The verb *"einigen"* in the German sentence appears as the 12th word, while *"agreed"* appears as the 7th word in the English translation. Thus, to provide a reliable translation, the translation would have to start with at least a 5-word delay.

Fig. 6.70 At a training event, a simultaneous translation system can overcome the language barrier and allow interactions. Image credits in Appendix A.3

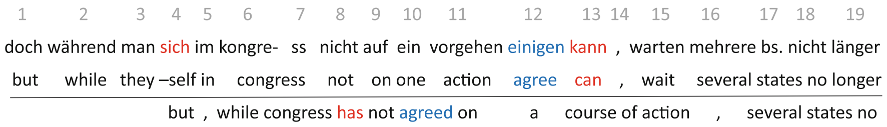

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
doch während man sich im kongre- ss nicht auf ein vorgehen einigen kann , warten mehrere bs. nicht länger
but while they –self in congress not on one action agree can , wait several states no longer
but , while congress has not agreed on a course of action , several states no

Fig. 6.71 German input sentence (first line), the corresponding word-by-word translation (second line) and the English translation (third line). The English translation starts after the third German word *"man"* is available. Therefore, during translation, the system has to predict the word *"agreed"*, for example, because the corresponding verb *"einigen"* is not available until later

Due to the simultaneous understanding and generation in two languages, simultaneous translation is extremely challenging and exhausting for human interpreters because of the concentration required and the strain on the voice. The number of qualified simultaneous interpreters worldwide is very limited, and each can only translate for about 15 to 30 minutes at a time, because the error rate rises sharply after just a few minutes.

Human simultaneous interpreters usually start translating a few seconds after the speaker has started and finish a few seconds after the speaker's speech has finished speaking. Ma et al. (2019) therefore propose to use a language model to predict the next words based on the words available so far in the input sentence. By this, they obtain hints about missing parts (e.g., verbs) of the input sentence. Their "wait-k" model is based on the Transformer and starts translation after k words of the input sentence are available. To do this, the decoder of the Transformer gets access to the embeddings of the first k words of the encoder at time k and uses the decoder to generate a first word of the translation. If the next word at time $k+1$ is known, this word is added to the encoder input and the decoder generates the next output word with the new embeddings, and so on. In this way, the encoder and decoder are continuously updated. This forces the decoder to generate a translation based on just a few words and implicitly trains the prediction of the following words of the original sentence (Fig. 6.71).

The translation quality depends of course on the waiting time k (Fig. 6.72). If the translation starts after the first word ($k = 1$), the translation from German to English on the WMT15 data set results in a BLEU value of 17. For $k = 3$ one reaches a BLEU value of 22, which increases to 26 for $k = 9$. If one starts the translation with the normal Transformer model already after k words (wait-k normal), much worse values result. The stars show the BLEU value, if the whole sentence is available without beam search (filled star) and with beam search (open star).

This model can be combined with a speech recognition and speech generation system so that speech input is translated into speech. Jia et al. (2019) have presented a translation system that translates directly from audio to audio.

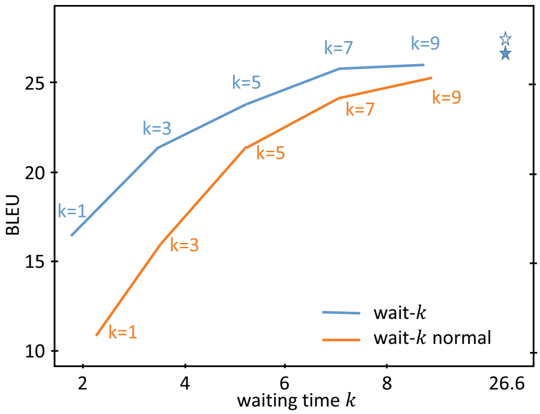


Fig. 6.72 Simultaneous translation from German to English. Results of the predictive Transformer model (wait-k) and an unadjusted Transformer model (wait-k normal) if both start translation after k words

6.8.6 Transfer Learning for Translation Models

In recent years, neural machine translation (NMT) has made very great progress (cf. Sect. 6.8.4). However, good translation performance requires training sets of many hundreds of thousands of sentence pairs from both languages. However, there are also language pairs for which very little training data is available. Here one can potentially use translation models of related languages to improve translation results.

Multilingual translation models use a single translation model to translate many languages into each other (Johnson et al. 2017). Here, the input sentence is prefixed with an abbreviation for the target language (e.g., “DE” for German). For example, Aharoni et al. (2019) use a Transformer model (Sect. 6.8) with six layers in the encoder and decoder with hidden vectors of length 2048, eight attention heads, and 93 million parameters. Especially for languages with small training sets, the BLEU value is greatly improved by such a multilingual model. For example, the simple translation model from Belarusian to English had a BLEU value of 2.8 for a training set of 4500 sentences, and the multilingual model achieved a BLEU value of 21.7. This clearly shows that the multilingual model can greatly improve the translation of single resource-poor language pairs through transfer learning.

Kudugunta et al. (2019) study the embeddings generated by multilingual translation models for the words of each language. They analyze the embeddings of 103 languages using a multidimensional technique (Singular Value Canonical Correlation Analysis) that arranges the average embeddings in a plane. Figure 6.73 shows that the embeddings can be grouped into language families, marked here by ellipses. However, one can also distinguish subgroups, for example, the South Slavic and the North Slavic languages.

For languages with large training datasets, multilingual language models usually do not improve but decrease the BLEU value. Bapna and Firat (2019) present a model called Mixture of Experts (Shazeer et al. 2017). It consists of k distinct “expert networks” $E_1(x), \ldots, E_k(x)$, each of which computes an output y from an

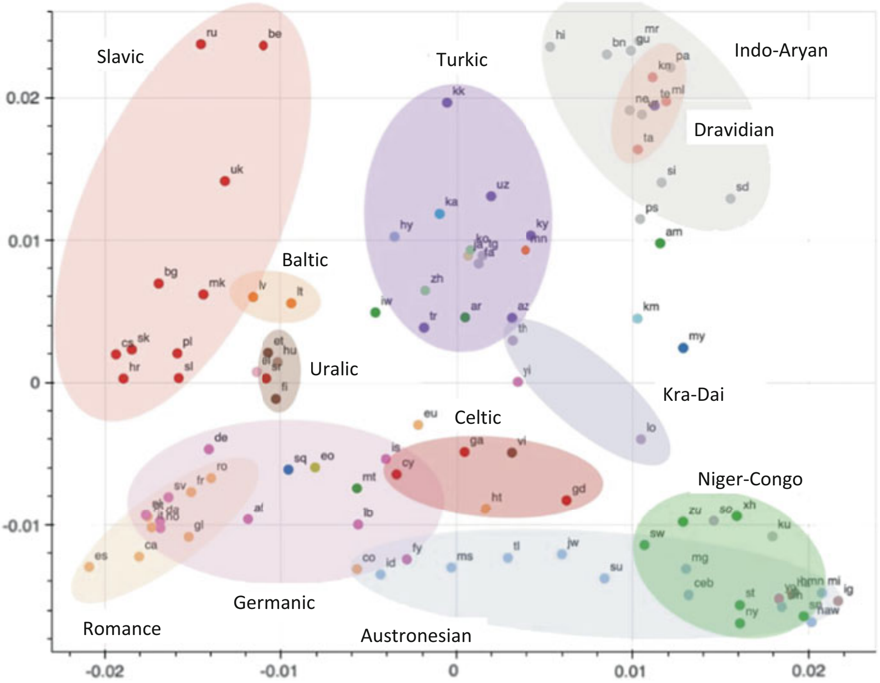


Fig. 6.73 As part of a multilingual model, texts from 103 languages were translated into English. Using the SVCCA technique, the embeddings of the words of each language were represented in one layer. They can be obviously grouped according to individual language families, which are represented by ellipses. Closely related languages can benefit more from each other in translation. Image credits in Appendix A.3

input x (Fig. 6.74). In addition, there is a gating network $G(x)$, which generates a weight for each net with which it should process the input x. The expert nets can be, for example, different layers in a Transformer model. It is crucial that the gating net selects only a few experts (usually two) for each input, so that no computations need to be done for the remaining experts. Both the expert networks and the gating network are trained together. This design allowed Bapna and Firat (2019) to dramatically increase the representational capacity of the translation models and train models with 50 billion parameters without increasing computational effort.

In total, the model incorporated more than 25 billion sentence pairs from 103 languages for training. This required the use of specialized parallel computer architectures. As shown in Fig. 6.75, for such comprehensive multilingual models, the translation quality for all language pairs is at least 5 BLEU points higher than for normal bilingual models. For languages with only a small amount of training data, BLEU values sometimes increase by more than 15 points. With this kind of transfer learning, the existing linguistic information of related languages can be

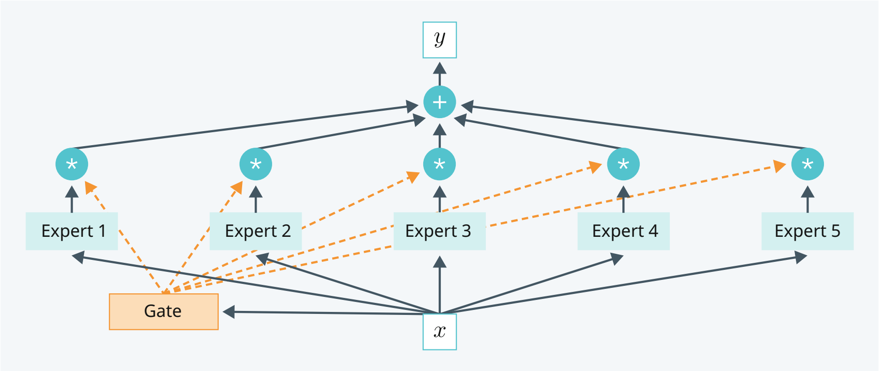


Fig. 6.74 Mixture of expert networks controlled by a gating network. The gating network decides which of the nets will process an input. Only for these few nets the output has to be calculated. The expert nets can be e.g. different Transformer layers

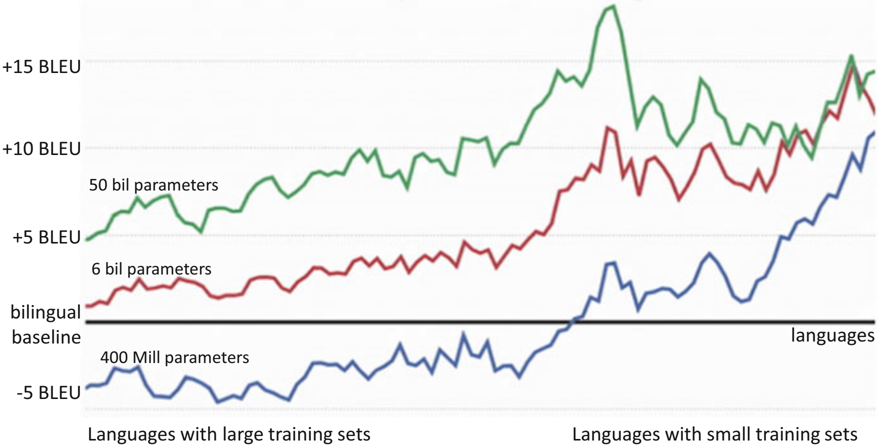
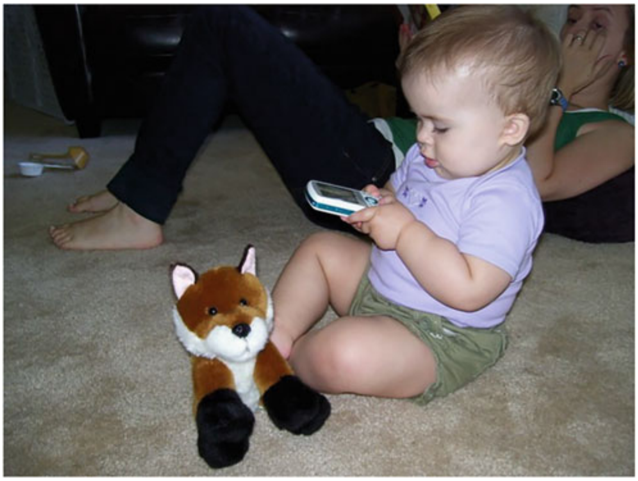


Fig. 6.75 Comparison of the results of simple bilingual translation models with the results of multilingual models with 400 million to 50 billion parameters. It is clear that the multilingual model provides a substantial improvement for all languages if the number of parameters is large enough. Image credits in Appendix A.3

exploited so efficiently that even the resource-rich languages benefit. However, the computational cost is extremely high.

6.9 The Description of Images by Text

The goal of image captioning is to automatically create a description for an image in natural language. This requires not only the listing of the most important objects



- A baby sitting on the floor playing with a cell phone.
- A baby girl and a toy fox with woman laying down.
- A small boy and his stuffed animal is playing with a game.
- A little baby holding a smart phone next to a stuffed animal.
- A small child holding a phone sitting near a stuffed animal.

Fig. 6.76 Image of an infant and associated descriptions (right) from the MS-Coco dataset. Image credits in Appendix A.3

in an image, but also describing the relationship between these objects and their meaning or role in the scene they represent. In Fig. 6.76, one can see the following objects in an image from the MS-COCO dataset: Infant, cell phone, stuffed animal, and woman. An important aspect of the scene is that the baby is playing with the cell phone, not texting. This requires everyday knowledge, such as what babies normally do. Five alternative captions are listed near the image. If a system is able to create suitable image descriptions, it can help people with visual disabilities to perceive images. In addition, an image description is often the basis for searching images in large image databases.

To train a model for image captioning, the MS-COCO dataset (Lin et al. 2014) often is used. It includes 120,000 images, each annotated with five different texts. Sharma et al. (2018) have created a new dataset of 3.3 million images with descriptions, the Conceptual Captions data. These images were collected from the web and annotated with text. Image descriptions were selected with respect to information content, fluency, and learnability of the results. Proper names were replaced with appropriate generic terms.

Sharma et al. (2018) use a three component system (Fig. 6.77) to generate an image description: A deep CNN (Sect. 5.5) receives an input image and uses it to generate a vector x with image embeddings for each 8×8 pixel region. An encoder module transforms x into a hidden vector h. The encoder's task is to relate the most relevant features extracted from the CNN and prepare them to generate a description. A decoder then generates an image description z from h as output.

Sharma et al. (2018) compare two variants of this structure. The first variant "L2L" consists of an LSTM encoder and LSTM decoder as in a translation model

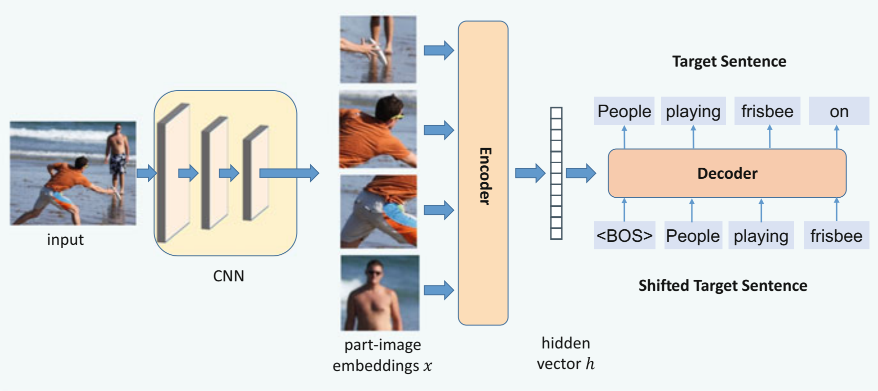
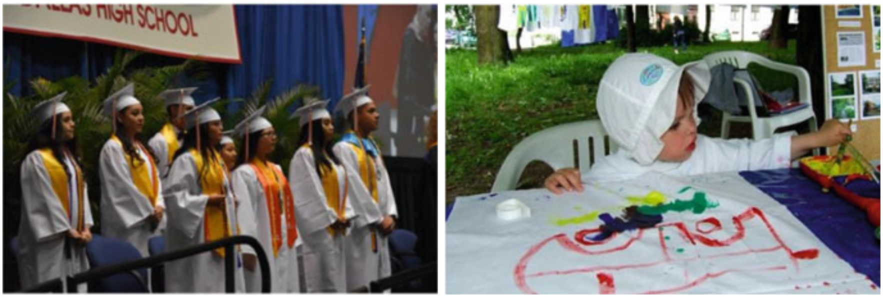

input
CNN
part-image embeddings x
Encoder
hidden vector h
Target Sentence
People playing frisbee on
Decoder
<BOS> People playing frisbee
Shifted Target Sentence

Fig. 6.77 Block function picture of a DNN to describe a photo in language. First, a CNN generates embeddings for 8×8 pixel regions. Then, an encoder formats them and generates a hidden vector h. Finally, a decoder generates the description text from the hidden vector. Image credits in Appendix A.3


RNN: graduates line up for the commencement ceremony
T2T: graduates line up to receive their diplomas

RNN: a child 's drawing at a birthday party
T2T: learning about the arts and crafts

Fig. 6.78 Two images and the descriptions predicted by an RNN and a Transformer (T2T) (Sharma et al. 2018). Image credits in Appendix A.3

for language (Sect. 6.5.1). The second variant "T2T" uses a Transformer model (Sect. 6.8) as encoder and decoder.

Figure 6.78 shows example images and the generated corresponding image descriptions. Evaluating the quality of image descriptions with text comparison measures (e.g., BLEU) is not meaningful. Therefore, human viewers evaluate the performance. The results on the Flickr data are shown in Fig. 6.79. Here, the Transformer model T2T performs better. However, it is still a long way from the 65.9% of 1+ ratings to the human performance of 96.0%. In the meantime, new methods have been presented (X. Li et al. 2020), which describe the objects of the image and their position in detail and thus produce better image captions.

It is evident that substantial progress is still necessary before the methods are useful in practice. This requires, on the one hand, that more meaningful information

Model	1 x good	2 x good	3 x good
L2L	57.1 %	41.8 %	27.7 %
T2T	65.9 %	50.6 %	35.5 %
human	96.9 %	90.3 %	78.5 %

Fig. 6.79 Quality of generated image descriptions. Three viewers are asked which of the descriptions of a picture are "good". The "1x good" column lists the proportion of descriptions that at least one viewer thought were good. The "3x good" column contains the proportion of cases where all three viewers judged the description to be good

about objects and their spatial and factual relations can be extracted from images. This information has to be combined with differently acquired knowledge about plausible relations between objects.

6.9.1 Explanation of DNN Forecasts

6.9.2 Explanations Are Necessary

Deep neural networks work well in many applications. However, it is not clear how the results were constructed in individual cases and whether the results can be trusted, i.e. they are black-box models. In the European Union, the General Data Protection Regulation (GDPR) came into force in 2018. An innovative aspect of the GDPR are the clauses on automated (algorithmic) decision making on individuals, including individual profiling. To a certain extent, this is the first time that all persons have the right to obtain a "meaningful explanation of logic" when automated decisions are made. One example of such a decision is when a model calculates a credit score, which aims to represent a person's creditworthiness. All participants agree that the implementation of such a principle is urgently needed and that it is a major open scientific challenge. Explainable models are needed to ensure user confidence in a trained system, to obtain fair and justifiable decisions, or to gain insight and deeper understanding of the data being analyzed.

Guidotti et al. (2018) give a detailed overview of the explainability of AI models. They distinguish different dimensions of interpretability.

- A DNN is globally interpretable if the overall logic of the model is clear.
- The model is locally interpretable if only the reasons for a single prediction can be explained.

6.9.3 Global Explanatory Models

One approach to explain a DNN is to determine simple global explanatory models that largely imitate the predictions of DNN and, on the other hand, are interpretable. The most important such models are decision trees, rule sets and linear models, which are shown in Fig. 6.80.

In a decision tree, questions on individual features are answered successively. Depending on the answer, questions on other features are answered subsequently. Finally, depending on the combination of features found, there is a prediction for the desired quantity.

In a rule set, rules are processed one after the other. These consist of logical conditions for characteristic values and give a prediction for an auxiliary variable or the desired quantity. It is important that the rules are handled in the specified order.

A linear model predicts an output variable, e.g. a temperature, or a probability (via a softmax function). The input features are multiplied by separate factors and all contributions are aggregated. The factors show the positive or negative contribution of each input variable. "const" is a feature with a fixed value of 1 that is always present and represents the global trend. A variant of the linear model is the generalized additive model (GAM). Here the individual characteristics of a linear model are transformed still before with a smooth function, e.g. with a logarithm or with $a_2x^2+a_1x$. The parameters of the linear model then still show the contributions of the individual features to the forecast.

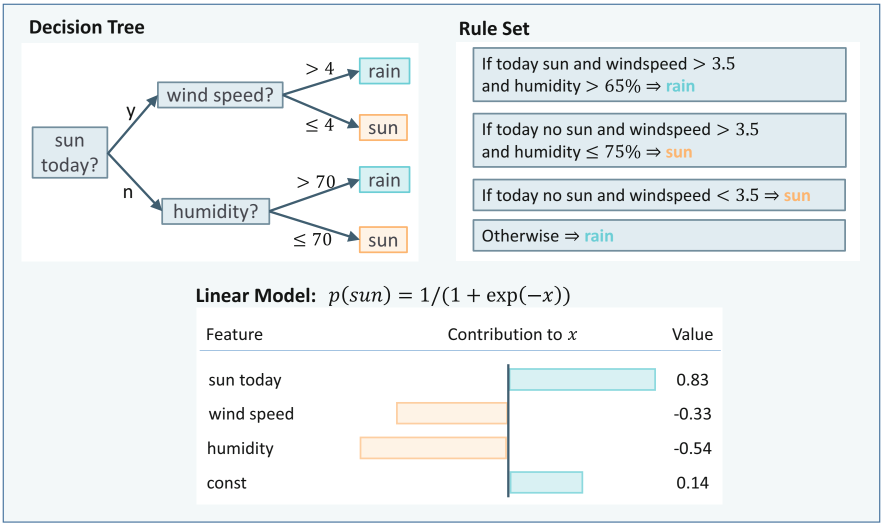

Decision Tree
sun today?
y
n
wind speed?
humidity?
> 4
≤ 4
> 70
≤ 70
rain
sun
rain
sun
Rule Set
If today sun and windspeed > 3.5 and humidity > 65% ⇒ rain
If today no sun and windspeed > 3.5 and humidity ≤ 75% ⇒ sun
If today no sun and windspeed < 3.5 ⇒ sun
Otherwise ⇒ rain
Linear Model: p(sun) = 1/(1 + exp(−x))
Feature
Contribution to x
Value
sun today
wind speed
humidity
const
0.83
-0.33
-0.54
0.14

Fig. 6.80 Three simple model types suitable to explain a relationship: decision tree (top left), rule set (top right), and linear model (bottom)

Even if models of this kind are used, interpretability depends on the complexity of the models, e.g. measured by the number of features or feature-value combinations. If, for instance, a decision tree contains several dozen decision queries, the interaction of these queries is often not transparent either. The explanatory models are directly applicable to numerical data. It becomes more difficult with images, texts and other sequences. In this case, easily interpretable features must first be derived from this data. First approaches are presented in Sect. 5.6.1.

6.9.4 *Local Explanatory Models*

Ribeiro et al. (2016) have suggested the LIME method, which can provide a plausible explanation for the prediction of a single input x_0 for any Machine Learning method. The idea of this local explanation model is that the explanation can be derived locally from training data that are in the neighborhood of the data point x_0 to be explained and that are weighted according to their distance from x_0.

The concept can be explained using Fig. 6.81. The idea here is to classify an input x_0 as "blue" or "pink" using a classifier. The boundary between "blue" and "pink" is a complex curved line. LIME does not try to capture the complete boundary, but only aims to explain the boundary near the input x_0 to be classified.

LIME calculates the Euclidean distances between x_0 (red X) and the other data points. The inverse distance defines the weights of the data points (plus and dot), which are symbolized by their size. Using these weighted data points, a linear model is trained (dashed line), which can be used to explain the classification near x_0. The parameter values of the linear model can be used for an explanation as in the bottom part of Fig. 6.80 and can be interpreted in terms of the strength and direction of

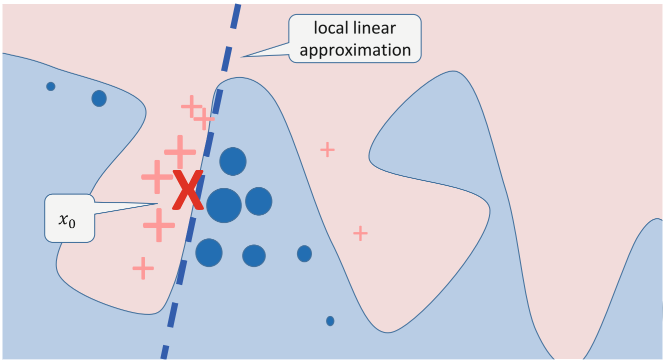


Fig. 6.81 LIME explains a DNN classification locally near an input x_0 (red X). Training data near x_0 are collected and weighted by their distance to x_0. The weights are symbolized by the size of the points. A linear model is then computed for the weighted training points (dashed line) and this is used to explain the prediction. Image credits in Appendix A.3

influence for different features. Hall (2020) gives detailed instructions on how to use this technique.

Level-wise relevance propagation (LRP) is an explanation method (Montavon et al. 2018) that is applicable to many Machine Learning approaches. In a first phase, a test example is propagated forward, storing the values of the hidden layers. In a second phase, the output activations of the last layer are distributed among the components of the previous hidden vectors in such a way that the sum of activations is preserved (Montavon et al. 2018). In this way, the activations can be progressively propagated back to the input, e.g., the input pixels of an image. The method can be embedded in the theoretical framework of a Taylor approximation. Figure 6.82 shows the LRP explanation for the classification of the top two images as "ship". Here, the contour and superstructure are marked for the passenger ship, while the sails play a crucial role for the sailing ships. Thus, for the same class, very different characteristics can be important for the classification. In the image below, you can see that the source note at the bottom left is relevant for the classification as "horse". If this reference is removed, the image is no longer classified as "horse". It turned out that the source reference was present on all training images with horses. This shows

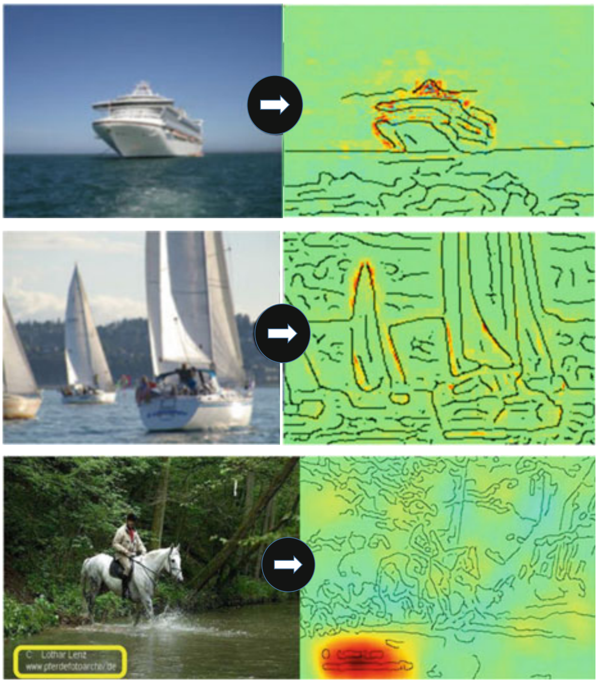


Fig. 6.82 Explanation of the classification of images. For classification as a ship, different features are marked as relevant by red coloring (top two images). The classification of the lowest image as a horse is based solely on the source note at the bottom left (Lapuschkin et al. 2019). Image credits in Appendix A.3

Languages	Test Data Input Errors (%)	BLEU. Training Data …	
		without Errors	with Errors
de → en	0.0	34.2	33.5
	9.7	27.9	31.3
	39.4	12.5	23.3
fr → en	0.0	39.6	39.9
	13.5	30.5	34.1
	53.7	11.5	19.4

Fig. 6.83 BLEU value for machine translation in the presence of varying levels of input errors in the text to be translated (3rd column). If these errors are added randomly during training, the BLEU value improves (4-th column) (Karpukhin et al. 2019)

that the explanation of the classification is necessary for assessing the transferability of the classifier to new data. Lapuschkin et al. (2019) have provided a number of programs to explain DNN.

6.10 Reliability of Text Understanding

6.10.1 Robustness in Case of Text Errors and Domain Change

Humans have an amazing ability to understand texts with errors. They can still capture the meaning of a sentence despite massive typos. An example is the following text from Twitter: *"Government confirms blast n nuclear plants n japan … don't knw wht s gona happen nw …"* (Ritter et al. 2011). Applying a standard named entity recognition model (Sect. 6.4.4) to Twitter texts yields an F-value of 0.44. By using training data that covered these errors, it was possible to achieve an F-value of 0.67 with the same model. By these results Ritter et al. (2011) demonstrated that it is absolutely necessary to use representative training data for Twitter and similar domains.

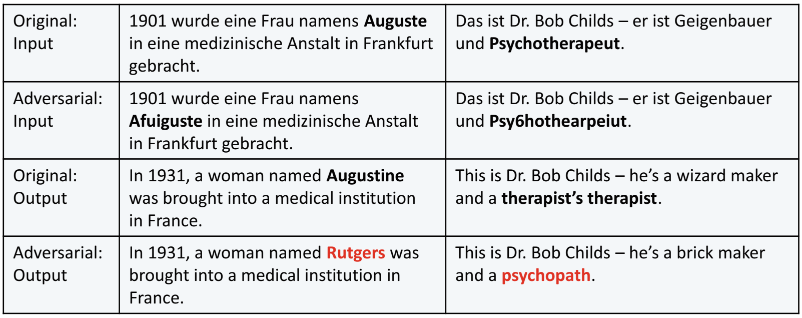
The susceptibility of sequence-to-sequence models for translation to textual errors is investigated by Karpukhin et al. (2019) to text errors. They analyze the influence of character errors on the translation result. For example, if in a translation from German to English 10% of the characters in the input sentence are changed, the BLEU value decreases from 34.2 to 27.9 (Fig. 6.83). The authors suggest to perform different types of character changes in the words of the training texts during the training of the translation model: Deletion, Insertion, Substitution, and Letter Swapping. These changes are made dynamically so that different modifications take place in each epoch. The distribution of letter modifications is based on the empirical frequency of typos. As can be seen in the last column of Fig. 6.83, the additional use of typos corrupted training texts leads to a substantial increase in the BLEU value without noticeably affecting the quality when translating error-free texts.

6.10.2 Vulnerability to Malicious Modification of Inputs

In Sect. 5.12.2 it was shown that in an adversarial attack images can be manipulated by minimal changes in such a way that the result of the image classification is altered. However, this approach cannot be directly applied to the textual data. First, pixel values are continuous, but text is composed of discrete tokens. Second, small changes in pixel values are hardly noticeable to viewers, whereas changes to tokens are immediately apparent.

Figure 6.84 contains adversarial examples of automatic language translation (Ebrahimi et al. 2018). The attacker's goal is to modify a phrase in the translation and replace it with a targeted text. For this purpose, words or syllables are first searched for in the source text via the calculation of derivations, which strongly influence the probability of the target text. These words or syllables are replaced by other parts of the text that avoid the target word and, if necessary, produce the target text in the translation. In Fig. 6.84, two examples of adversarial input and the resulting translations are shown.

An overview of adversarial attacks in natural language processing is given by Zhang et al. (2020). There are many different approaches for generating the changes and also very many application areas, e.g., text classification, machine translation, question answering, dialog systems, etc. Overall, the generation of adversarial examples is more difficult than in image processing and is often still done manually. Different strategies are discussed as defense measures. The main defense strategy is adversarial training, where a DNN is trained with both the normal training examples and the adversarial examples.



Original: Input	1901 wurde eine Frau namens **Auguste** in eine medizinische Anstalt in Frankfurt gebracht.	Das ist Dr. Bob Childs – er ist Geigenbauer und **Psychotherapeut**.
Adversarial: Input	1901 wurde eine Frau namens **Afuiguste** in eine medizinische Anstalt in Frankfurt gebracht.	Das ist Dr. Bob Childs – er ist Geigenbauer und **Psy6hothearpeiut**.
Original: Output	In 1931, a woman named **Augustine** was brought into a medical institution in France.	This is Dr. Bob Childs – he's a wizard maker and a **therapist's therapist**.
Adversarial: Output	In 1931, a woman named **Rutgers** was brought into a medical institution in France.	This is Dr. Bob Childs – he's a brick maker and a **psychopath**.

Fig. 6.84 Adversarial examples of automatic translation from German to English. In the first example, the attacker wants to suppress the name of the person. In the second example, he wants to replace "therapist" with "psychopath" Ebrahimi et al. (2018)

6.11 Summary and Trends

Deep neural networks almost always transform textual inputs to embeddings, which represent the meaning of words. Historically, simple models, such as Word2vec, were used to compute static embeddings. Today, embeddings are contextual and are able to represent different meanings of the same word depending on the context.

Language models have the task to predict the next word of a text. The Long Short-Term Memory (LSTM) was the first successful language model. It was also the basis for the first translation models, which were composed of two LSTMs: The encoder LSTM encodes the input text into a hidden vector. The decoder LSTM decodes this hidden vector into the text of the target language.

These translation models could be improved with attention. In this process, the similarity between a hidden vector of the decoder LSTM and the hidden vectors of the encoder LSTM is computed. According to this similarity, the hidden vectors of the encoder LSTM are aggregated to a context vector, which provides additional information to generate the output word.

The BERT model is based on self-attention. It computes the "correlation" between the embeddings of different words or tokens and generates new embeddings by weighted averaging of related embeddings. This scheme is applied in parallel in multiple layers. The resulting embeddings can represent different meanings of the same token, depending on the context.

The BERT model applied self-attention to two training tasks: for masked language modeling, individual missing words must be predicted, and in the next sentence prediction, it has to be decided whether the second sentence is the continuation of the first. This model was pre-trained on very large text collections and acquired substantial syntactic and semantic knowledge.

By fine-tuning the model on a new small training data set for a few iterations, the pre-trained BERT model can be adapted to a large number of semantic problems, e.g., question answering, logical inferences, sentiment analysis, etc. By this transfer learning, BERT achieved many state-of-the-art results, often with higher accuracy than humans. Figure 6.85 shows the evolution of accuracy on the GLUE benchmark, which includes a mixture of such tasks. This is a real breakthrough in the field of text interpretation.

The Transformer model extends and generalizes this mechanism for text translation. An encoder generates embeddings for the input tokens and a decoder model successively produces the next translated token. The Transformer achieves higher translation performance accuracy and requires less computation.

A short section deals with the description of images by text. This is possible, for example, using a CNN for image processing and a subsequent transformer encoder-decoder that generates the text. The accuracy of these models is still much worse than that of humans.

The description of images with text requires training data, in which the objects contained in the image are annotated by captions. In most cases, a description of the actions visible in the image is missing. Models trained in this way make seemingly

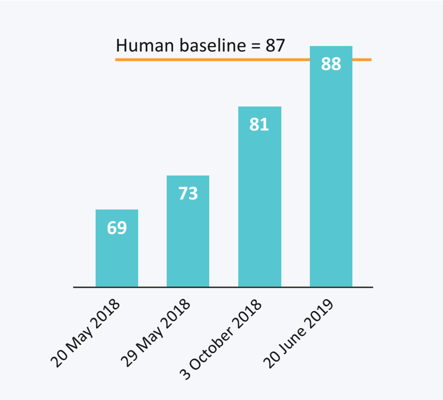


Fig. 6.85 Increased accuracy on GLUE, a collection of semantic text analysis benchmarks, over the course of a year. Image credits in Appendix A.3

obvious mistakes when trying to describe visual scenes. Rather, activity descriptions are also needed for training. This suggests that an AI system must have a reasonable world model of objects and people to truly understand what is happening in a visual scene.

The performance of the model degrades when textual errors occur. However, one can extend the training data with simulated errors and achieve much higher robustness. Adversarial attacks are not as big a problem as with images, since text modifications are very easy to detect.

We have seen that tokens, words, sentences or even whole documents are described by embeddings in the form of vectors. This learning of representations is a general approach (Bengio et al. 2013), which will be extended to other media (speech, videos, actions, etc.) in the further chapters. These representations are learned during training for specific tasks, and their interpretation always depends on the particular network architecture. The advantage of these representations is that they summarize the previously processed content in condensed form and that they can be seamlessly processed by subsequent layers.

Trends

- In the field of language translation, approaches are currently being examined to train translation models even for languages without extensive bilingual training data. These models start with a few words for which translations are known. They train language models on monolingual text collections, use translation and back-translation, and achieve surprisingly high performance (Lample et al. 2018).
- The pre-trained BERT model can extract world knowledge from very large collections of texts. Now there are approaches to acquire this world knowledge in a more structured form. Sap et al. (2019) have collected

(continued)

extensive training data in which plausible preconditions and consequences are compiled for events. One example is the event “person X bakes bread”. Preconditions might be “buy ingredients” or “turn on oven”, while possible outcomes might be “get dirty” or “eat”. From this type of data, DNNs can be trained that have information about plausible preconditions or consequences of events.

- The embeddings generated by pre-trained language models become ubiquitous. It will be rare for a current model not to use such embeddings. Additionally, there will be representations for specialized content, e.g. image parts. These different embeddings and representations can then be combined.
- It is now possible to combine the results of large language models with a web search to answer question. Riedel et al. (2020) use the BART model (M. Lewis et al. 2020) to encode a question, obtain search results from the web for this purpose, and finally combine the search results with the background knowledge of the language model to compose an answer.

References

Aharoni, R., Johnson, M., & Firat, O. (2019). Massively multilingual neural machine translation. arXiv: 1903.00089.

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. ICLR. arXiv: 1409.0473.

Bapna A., & Firat, O. (2019). Exploring massively multilingual, massive neural machine translation. https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html

Bar-Hillel, Y. (1962). The future of machine translation. *Times Literary Supplement Times Newspaper Londondon* April 20th 1962.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A Review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1798–1828.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146.

Bojar, O., Chatterjee, R., Christian, F., Yvette, G., Barry, H., Matthias, H., Philipp, K., Qun, L., Varvara, L., Christof, M., et al. (2017). Findings of the 2017 conference on machine translation (Wmt17). In *Proceedings of the Second Conference on Machine Translation* (pp. pp. 169–214). The Association for Computational Linguistics.

Britz, D., Goldie, A., Luong, M.-T., & Le, Q. (2017). Massive exploration of neural machine translation architectures. *Conference on Empirical Methods in Natural Language Processing* (P1442–1451).

Clark, P., Etzioni, O., Khashabi, D., Khot, T., Mishra, B. D., Richardson, K., Sab-harwal, A., Schoenick, C., Tafjord, O., Tandon, N., et al. (2019). From ’F’ to ’A’ on the NY regents science exams: An overview of the aristo project. arXiv: 1909.01958.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv: 1810.04805.

Diaz, F., Mitra, B., & Craswell, N. (2016). Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (vol. 1 Long Paper, pp. 367–377).

Dyer, C. (2014). Notes on noise contrastive estimation and negative sampling. arXiv: 1410.8251.

Ebrahimi, J., Lowd, D., & Dou, D. (2018). On adversarial examples for character-level neural machine translation. arXiv: 1806.09030.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*(2), 179–211.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-1955, volume 1952-59. *The Philological Society*

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Con-volutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning, 70*, 1243–1252.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys CSUR, 51*(5), 1–42.

Hall, P. (2020). On the art and science of machine learning explanations. arXiv–1810.

Hassan, H., Aue, A., Chen, C., Chowdhary, V., Clark, J., Federmann, C. Huang, X., Junczys-Dowmunt, M., Lewis, W., Li, M., et al. (2018). Achieving human parity on automatic Chinese to english news translation. arXiv: 1803.05567.

Hendrycks, D., & Gimpel, K. (2016). Bridging nonlinearities and stochastic regularizers with gaussian error linear units. arXiv: 1606.08415.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., & Wu, Y. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. arXiv: 1904.06037.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F., Wattenberg, M., Corrado, G., et al. (2017). Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics, 5*, 339–351.

Karpathy, A. (2015). The unreasonable effectiveness of recurrent neural networks. *Andrej Karpathy Blog, 21*, 23.

Karpukhin, V., Levy, O., Eisenstein, J., & Ghazvininejad, M. (2019). Training on synthetic noise improves robustness to natural noise in machine translation. arXiv: 1902.01509.

Kudugunta, S. R., Bapna, A., Caswell, I., Arivazhagan, N., & Firat, O. (2019). Investigating multilingual NMT representations at scale. arXiv: 1909.02197.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 260–270)

Lample, G., Ott, M., Conneau, A., Denoyer, L., & Ranzato, M. (2018). Phrase-based and neural unsupervised machine translation. arXiv: 1804.07755.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite BERT for self-supervised learning of language representations. arXiv: 1909.11942.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K.-R. (2019). Unmasking clever hans predictors and assessing what machines really learn. *Nature Communications, 10*(1), 1–8.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020). BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics, 36*(4), 1234–1240.

Lenat, D. B. (1995). CYC: A large-scale investment in knowledge infrastructure. *Communication of ACM, 38*(11), 33–38.

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics, 3*, 211–225.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv: 1910.13461.

Li, X., Yin, X., Li, C., Zhang, P., Hu, X., Zhang, L., Wang, L., Hu, H., Dong, L., & Wei, F. (2020). Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision* (pp. 121–137). Springer.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European Conference on Computer Vision* (pp. 740–755). Springer.

Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., Zhang, C., He, Z., Liu, H., Li, X., et al. (2019). STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3025–3036).

Melis, G., Dyer, C., & Blunsom, P. (2017). On the state of the art of evaluation in neural language models. arXiv: 1707.05589.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv: 1301.3781.

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73*, 1–15.

Mushtaq, M. F., Akram, U., Aamir, M., Ali, H., & Zulqarnain, M. (2019). Neural network techniques for time series prediction: A review. *JOIV : International Journal on Informatics Visualization, 3*(3), 314–320.

Nayak, P. (2019). Understanding searches better than ever before. Google Blog Oct, p. 25.

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. arXiv: 1312.6026.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, Y. Wu, A., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? arXiv: 1909.01066.

Popel, M., & Bojar, O. (2018). Training tips for the transformer model. In: *The Prague Bulletin of Mathematical Linguistics, 110*(1), 43–70.

Quine, W. V. O. (1969). Ontological relativity. *Ontological Relativity and Other Essays, 15*, 26–68.

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. arXiv: 1606.05250.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. arXiv: 1606.05386.

Riedel, S., Kiela, D., Lewis, P., & Piktus, A. (2020). *Retrieval augmented generation: Streamlining the creation of intelligent natural language processing models*. Retrieved November 02, 2020, from https://ai.facebook.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/

Ritter, A., Clark, S., Etzioni, O., et al. (2011). Named entity recognition in tweets: An experimental study. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1524–1534)

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.

Sap, M., Le Bras, R., Allaway, E., Bhagavatula, C., Lourie, N., Rashkin, H., Roof, B., Smith, N. A., & Choi, Y. (2019). Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (vol. 33, pp. 3027–3035)

Sharma, P., Ding, N., Goodman, S., & Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* (vol. 1 Long paper, pp. 2556–2565).

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv: 1701.06538.

Strobelt, H., Gehrmann, S., Huber, B., Pfister, H., Rush, A. M., et al. (2016). Visual analysis of hidden state dynamics in recurrent neural networks. arXiv: 1606.07461.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (pp. 3104–3112)

Thrun, S. (1998). Lifelong learning algorithms. In *Learning to Learn* (pp. 181–209). Springer.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using T-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008). https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Williams, A., Nangia, N., & Bowman, S. (2018). MultiNLI. Retrieved August 26, 2022, from https://cims.nyu.edu/~sbowman/multinli/

Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., & Weston, J. (2018). Starspace: Embed all the things! In *Proceedings of the AAAI Conference on Artificial Intelligence* (vol. 32, p. 1).

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., & Macherey, K. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv: 1609.08144.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems* (pp. 5753–5763)

Zellers, R., Bisk, Y., Schwartz, R., Choi, Y. (2018). Swag: A large-scale adversarial dataset for grounded commonsense inference. arXiv: 1808.05326.

Zhang, W. Sheng, E., Q. Z., Alhazmi, A., & Li, C. (2020). Adversarial attacks on deep-learning models in natural language processing: A survey. In *ACM Transactions on Intelligent Systems and Technology TIST, 11*(3), 1–41.

Zhou, J., Cao, Y., Wang, X., Li, P., & Xu, W. (2016). Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics, 4*, pp. 371–383.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 19–27).

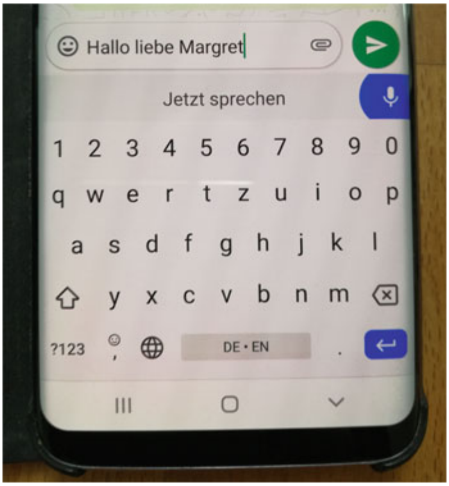
Chapter 7
Understanding Spoken Language

Abstract This chapter describes models for speech recognition, i.e. for transferring spoken language into text. Speech recognizers use derived sound features for small time intervals as input. For speech processing, deep sequence-to-sequence models based on LSTM or transformers are used, which generate the recognized text. Alternatively, Convolutional Neural Networks are employed. A hybrid model of Sequence-to-Sequence and CNN models is able to achieve a lower recognition error than humans. When generating speech from text, WaveNet's dilated CNN layers can reproduce the acoustic speech of a speaker extremely faithfully. Voice assistants, such as Siri and Alexa, allow users to engage in a dialogue. An example system is used to illustrate the construction of a variant of the Alexa voice assistant from subnetworks and other components. The classification of the events in a video is possible with variants of spatio-temporal convolutional layers. More difficult is the description of videos by subtitles, which can be done for example with the help of transformer translation models. In a last section, the influence of noise on speech recognition and the potential danger of adversarial attacks are discussed.

7.1 Speech Recognition

Speech recognition is the process of converting spoken language into audio signals and transcribing speech to text. A good speech recognition system replaces the typist who used to turn up for dictation and then type the correspondence into the machine. Today, speech recognition also enables verbal control of many devices, from the car phone to the smartphone (Fig. 7.1) to voice activation of lights or heating at home. In addition, speech recognition is making many dialog applications possible, e.g., for central call acceptance, for querying the concerns of customers in companies, or for reserving tables in restaurants. Personal assistants, such as Amazon's Alexa, Apple's Siri, or Google Home, are a particularly booming area of application. With such a personal assistant, the users can have music played, edit his appointment calendar, make search queries on the Internet, etc. The personal assistant thus becomes a "servant" who is available at any time of day and fulfills the user's wishes.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_7



Hallo liebe Margret

Fig. 7.1 Speech recognition through the Gboard keyboard app on the smartphone

7.1.1 *Why Is Speech Recognition Difficult?*

Humans can generally understand language without any effort. For machines, this is difficult for a number of reasons. When someone speaks to us, for example in a restaurant, we have to separate the words from the background noise. This background noise can have a number of sources, such as other people's conversations, traffic noise, wind, and so on. Another difficulty is that people speak at different speeds, and in the flow of speech it's hard to hear when words end or when a new sentence begins. In addition, each person has their own speech melody and there are many dialects and accents. Moreover, spoken sentences are often grammatically incorrect and there are many filler words like *"hmm"* and *"uh"* that have to be recognized and omitted. But there are also often ambiguities in the language itself. Many words sound similar but have very different meanings, e.g., *"too"* and *"two"* or *"by"* and *"buy"*.

Recognizing speech from different speakers poses particular problems. For this reason, speech recognition systems were initially aimed at individual speakers and had to be adapted to this speaker through additional training. In continuous speech recognition, the beginnings of individual words are not marked in the audio stream by pauses. Therefore, in the first speech recognition systems, the individual words had to be dictated with pauses in between.

7.1.2 *How to Represent Speech Signals in the Computer?*

Sound consists of movements of air molecules or other particles. These move rhythmically toward and away from the sound source. This changes the air pressure

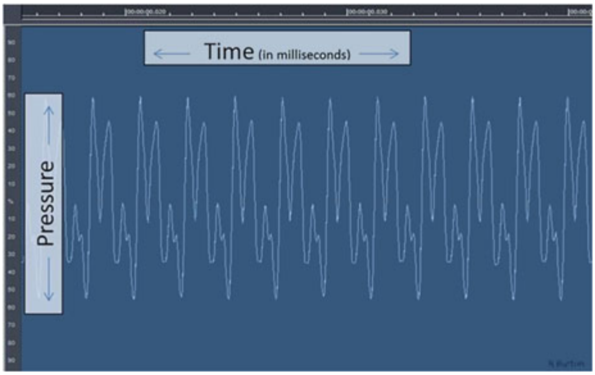


Fig. 7.2 Measurement of sound pressure over time of 20 msec for the sound of a clarinet. Image credits in Appendix A.3

(also called sound pressure), which periodically rises and falls again. Before the computer can recognize speech, a microphone must convert the vibrations of the spoken language into a wave-shaped electrical signal. In Fig. 7.2 for example, the sound of a clarinet is shown as sound pressure over time.

For processing, the recorded analog speech signal must then be converted in a digital signal, a sequence of real numbers. This can be done, for example, by recording the sound pressure level at certain time points and then processing it in the computer. But this requires a lot of computer memory, because the recording rate must be high.

An alternative offers the representation of the audio signal in the frequency domain using the Fourier transform, which transfers the oscillations into frequency information within time intervals. Such a signal in the frequency domain requires less memory, although the information about the spoken words is preserved.

The Fourier transform can represent the audio signal as a sum of simple functions—sine and cosine. This is shown schematically in Fig. 7.3. Here the audio signal is approximated by a sum of six sinus waves. Each of these sinus waves has a certain frequency (frequency of oscillation per time). One then needs only the "proportions" of each of the curves—six numbers—to represent the signal. Importantly, only the frequencies relevant to human speech (20 to about 8000 oscillations per second) need to be considered.

Figure 7.4 shows the audio signal of a spoken sentence for 10 seconds and the corresponding spectrogram. For each interval of 10 milliseconds it contains the sound energy present in different frequency ranges. High energies are symbolized by red color and low energies by blue color.

For speech recognition, it is not the pitch that is important, as this differs between women, men and children. Rather, the relative height of the frequencies is essential. In addition, the frequencies below 1000 Hz are more important than the higher frequencies. These requirements are taken into account by variants of the spectrogram, e.g. the Mel Frequency Cepstral Coefficients (MFCC), which are often used as input for speech recognition. The details of their computation are presented

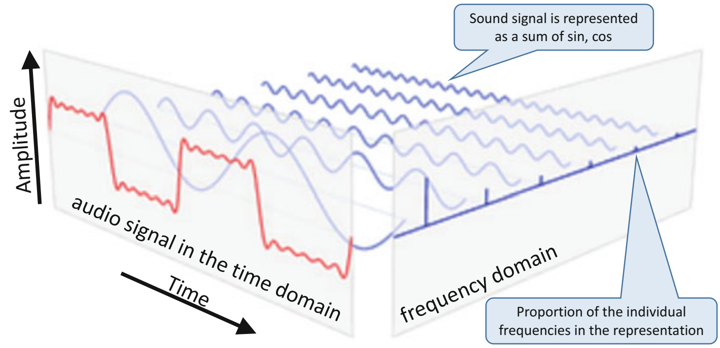

Sound signal is represented as a sum of sin, cos
Amplitude
audio signal in the time domain
Time
frequency domain
Proportion of the individual frequencies in the representation

Fig. 7.3 Representation of a sound signal (red) by sinus curves, each of which has a specific frequency. The proportions of the individual curves (blue, green, purple) are indicated by small upright bars (right). Image credits in Appendix A.3

by (MFCC 2019). They consist of a vector of about 40 features for each 10-msec interval.

7.1.3 Assessing Speech Recognition Accuracy

The Word Error Rate WER

The word error rate (WER) is used to measure the accuracy of speech recognition. It calculates the "difference" between a correct reference sentence and the sentence transcription generated by the model. First, the words of the two sentences are matched as closely as possible. This can be done with the help of dynamic programming, e.g. by the Levenshtein distance (Berger et al. 2020). The following types of errors are distinguished: S is the number of substitutions, D is the number of deletions, and I is the number of insertions. The word error rate is then defined as $WER = (S + D + I)/N$, where N is the length of the reference sentence. For example, consider the reference sentence *"that nice day ends"* and the model prediction *"this day ends tragically"*. Then one must make a substitution (*"this"* → *"that"*), an insertion (*"nice"*), and a deletion (*"tragically"*) to produce the reference sentence (Fig. 7.5). The word error rate is therefore $3/4 = 0.75$, which is very bad.

Established Speech Recognition Benchmarks

A standard benchmark for measuring the accuracy of speech recognition is the Switchboard corpus. It contains recordings of telephone conversations about general topics such as sports and politics. Since the participants in the conversation are not speaking to their own family members, the conversation is in North American

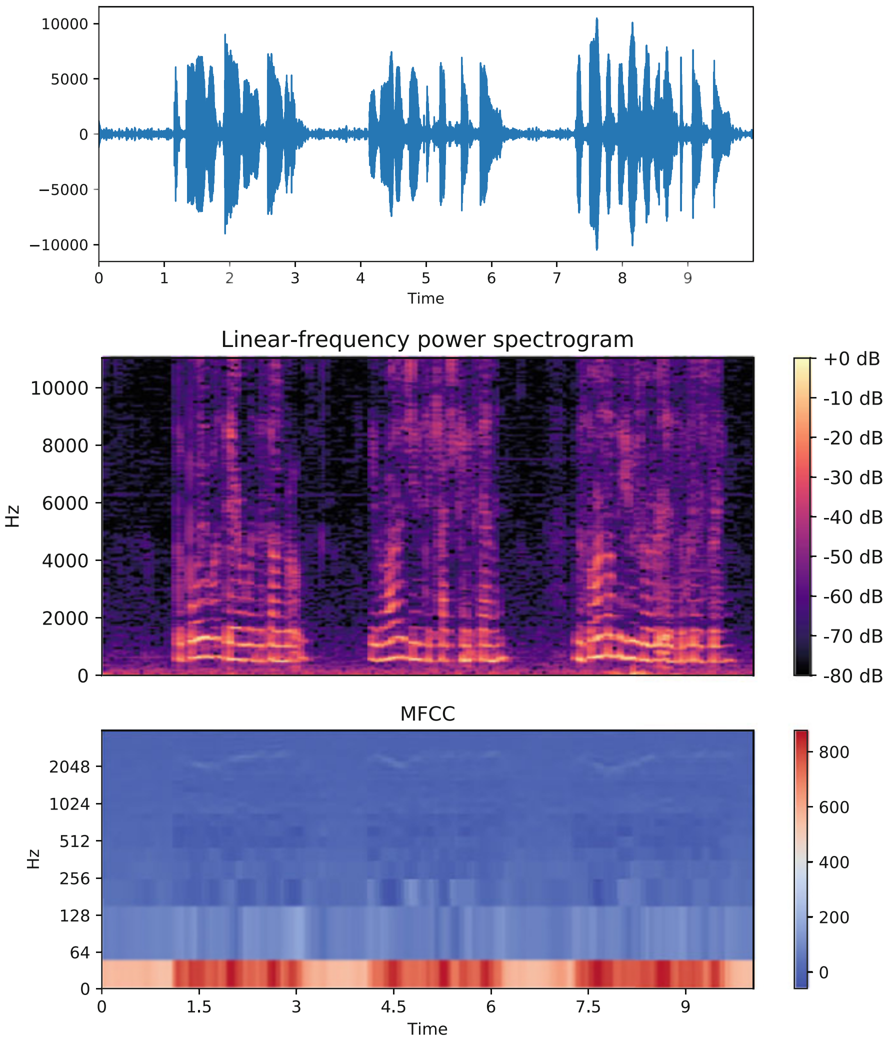


Fig. 7.4 Sound signal of a sentence (top) during an interval of 10 seconds (x-axis). In the middle is the spectrogram showing for each 10 msec interval the magnitude of the respective oscillation frequencies. Red color indicates high energy and blue color indicates low energy in the respective frequency range. Below are the derived Mel Frequency Cepstral Coefficients (MFCC)

Standard English. There are about 2400 telephone conversations between 543 speakers (302 male and 241 female) from all parts of the United States. For a detailed description and examples see (Switchboard 1997) and Fig. 7.6.

Another frequently used benchmark is the LibriSpeech corpus (Panayotov et al. 2015). It consists of about 1000 hours of spoken audiobooks. The main difficulty in constructing this corpus was to check whether the reader had actually read the

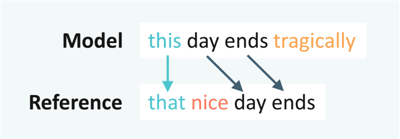
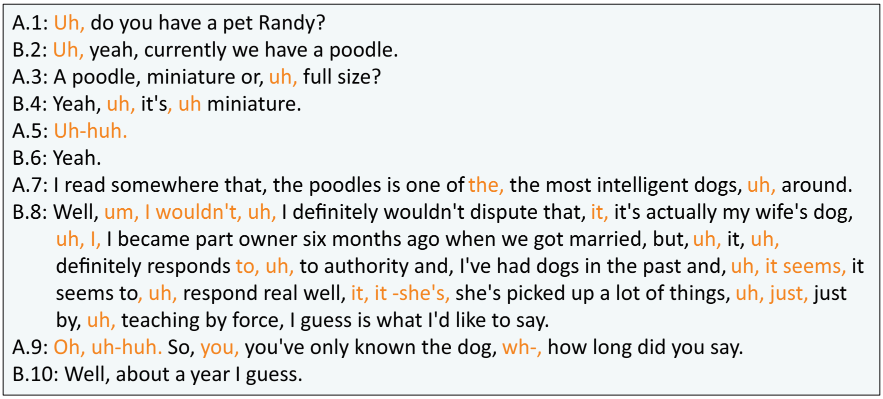


Fig. 7.5 Word Error Rate (WER) for measuring the quality of speech recognition. Below is the correct reference text and above the text generated by the model. Error types are distinguished: substitution (blue), insertion (red) and deletion (orange)



A.1: Uh, do you have a pet Randy?
B.2: Uh, yeah, currently we have a poodle.
A.3: A poodle, miniature or, uh, full size?
B.4: Yeah, uh, it's, uh miniature.
A.5: Uh-huh.
B.6: Yeah.
A.7: I read somewhere that, the poodles is one of the, the most intelligent dogs, uh, around.
B.8: Well, um, I wouldn't, uh, I definitely wouldn't dispute that, it, it's actually my wife's dog, uh, I, I became part owner six months ago when we got married, but, uh, it, uh, definitely responds to, uh, to authority and, I've had dogs in the past and, uh, it seems, it seems to, uh, respond real well, it, it -she's, she's picked up a lot of things, uh, just, just by, uh, teaching by force, I guess is what I'd like to say.
A.9: Oh, uh-huh. So, you, you've only known the dog, wh-, how long did you say.
B.10: Well, about a year I guess.

Fig. 7.6 Sample dialog from the Switchboard corpus (Switchboard 1997). Non-grammatical insertions are marked orange

text correctly, or whether there were word substitutions, word insertions, or word rearrangements. In constructing the corpus, a sophisticated system of checks was used to detect such modifications.

7.1.4 The History of Speech Recognition

Figure 7.7 outlines the history of spoken language recognition. Already in the 1960s there were first simple models for the recognition of words. In the 1980s, IBM, under the direction of Fred Jelinek, developed the voice-controlled typewriter "Tangora," which could understand a vocabulary of 20,000 words. Jelinek used statistical models, which enabled very large advances in the field (e.g., the Hidden Markov Model). In the 1990s, speech recognition experienced its first breakthrough with faster microprocessors through the development of "Dragon Dictate" for consumers. The "Dragon Naturally Speaking" extension could already recognize 100 words per minute. Such models consisted of at least two submodels, an acoustic model for mapping the speech signal to phonemes or letters, and a language model for distinguishing correct from implausible word sequences.

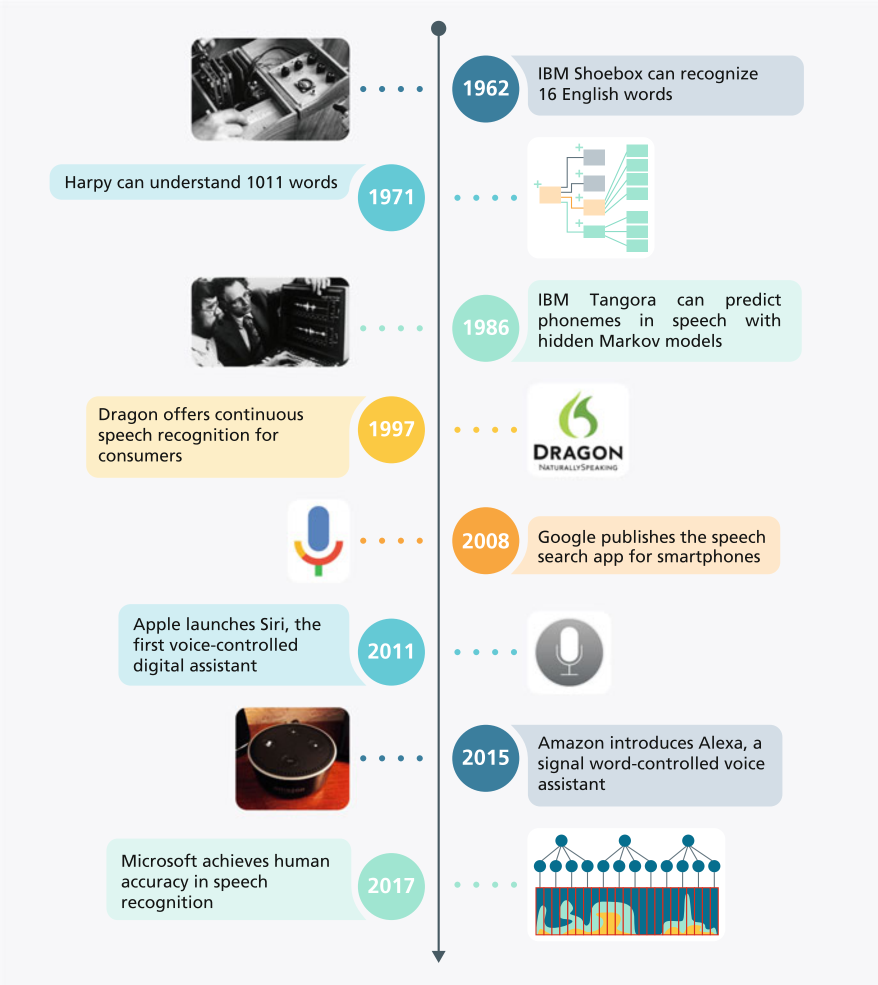

1962
IBM Shoebox can recognize 16 English words
Harpy can understand 1011 words
1971
1986
IBM Tangora can predict phonemes in speech with hidden Markov models
Dragon offers continuous speech recognition for consumers
1997
DRAGON
NaturallySpeaking
2008
Google publishes the speech search app for smartphones
Apple launches Siri, the first voice-controlled digital assistant
2011
2015
Amazon introduces Alexa, a signal word-controlled voice assistant
Microsoft achieves human accuracy in speech recognition
2017

Fig. 7.7 History of speech recognition. Image credits in Appendix A.3

Around 2007, recurrent LSTM models outperformed classical speech recognition algorithms for the first time. With the use of further deep neural networks, the error rate of speech recognition could be reduced by 30% after only 1 year (Markoff 2012). In 2017, Microsoft presented a speech recognition system that achieved human accuracy.

7.2 Deep Sequence-to-Sequence Models

In this section, we will first discuss Sequence-to-Sequence models for speech recognition, which generate text as either letters or also partial words. In the following Sect. 7.3, Convolutional Neural Network models for speech recognition will be presented. Since both types of models have different strengths, the subsequent Sect. 7.3.2 also considers combined models, which often achieve better accuracy.

7.2.1 List-Attend-Spell Generates a Sequence of Letters

The speech recognition problem can be approached as a translation task. Input is the sequence of frequency features for fixed time intervals of 10 msec. Output is the sequence of letters or words. After the success of the Sequence-to-Sequence models (Sect. 6.5.1) for translation from one language to another, it was a logical step to use these models for speech recognition as well. Unlike previous speech recognition systems, which consisted of multiple submodels, all parts of a sequence-to-sequence model can be trained end-to-end. This is preferable because all model parts are then optimally matched to each other.

Chan et al. (2016) developed one of the first sequence-to-sequence models of its kind, which could translate audio features into letters. It was called the Listen-Attend-Spell (LAS) model. As with the translation model in Fig. 6.42, bidirectional LSTMs were used as encoders and LSTMs with attention were employed as decoders to generate the sequence of letters in the output text. However, these models initially converged very slowly or not at all because the sequence of input features (about 100 per sec) was very long compared to the number of output letters. Therefore, the authors shortened the input length of the encoder BiLSTMs by concatenating the hidden vectors of each two consecutive time steps. The resulting vectors were used as input to the BiLSTM of the next layer. By employing three layers of BiLSTMs with hidden vectors of length 512, a sequence length of 1/8 of the input length was therefore generated by this “pyramid encoder”. This made the computation of the attention analogous to Fig. 6.42 much simpler and more effective.

The decoder works in the same way as in the translation model in Fig. 6.42. A two-layer LSTM computes the hidden vector h_2^{dec} for the next letter to be predicted. In Fig. 7.8, this is the letter *“e”*, using the previous observed letter *“H”* as input. As in the translation model, the attention between hidden vectors of the decoder and encoder is computed and a context vector c_2 is formed from the hidden encoder vectors h_i^{enc}. This context vector c_2 is then appended to the hidden vector h_2^{dec} and entered into a logistic regression model that computes the probability distribution for the next letter. In this way, the probability of the letters of the output text can be successively calculated during training.

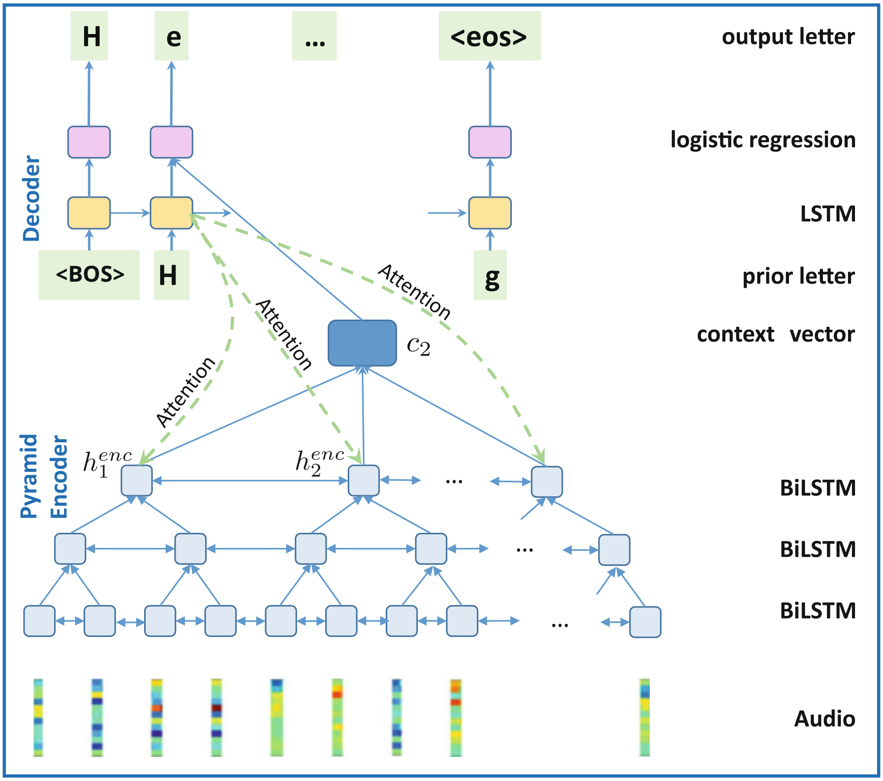


Fig. 7.8 Speech recognition with a pyramid encoder and an attention decoder. The encoder consists of three bidirectional LSTMs (BiLSTM), which successively shorten the length of the input sequence to 1/8. The decoder consists of two LSTMs. The attention is computed analogously to Fig. 6.42 and used to compute the probability of the next letter (at the top). (Source: Chan et al. (2016))

In Fig. 7.9, the attention between the audio features (above) and the associated text *"how much wood would a woodchuck chuck"* is represented by different gray values. The attention mechanism generates an explicit mapping between the audio features and the letters. It can correctly identify the beginning of the spoken sentence in the audio sequence. The assignment of the letters generally follows the time sequence in the audio sequence. The words *"woodchuck"* and *"chuck"* are acoustically similar. Therefore, the letters *"chuck"* of *"woodchuck"* also have weak assignments to multiple audio positions.

The model was trained on 2000 hours of Google searches using stochastic gradient descent (Chan et al. 2016). Ten hours of the randomly selected queries were used as the validation set. The queries were partially overlaid with different types of background noise taken from YouTube videos of everyday environments. This increased the size of the data by a factor of 20. Inputs were 40-dimensional

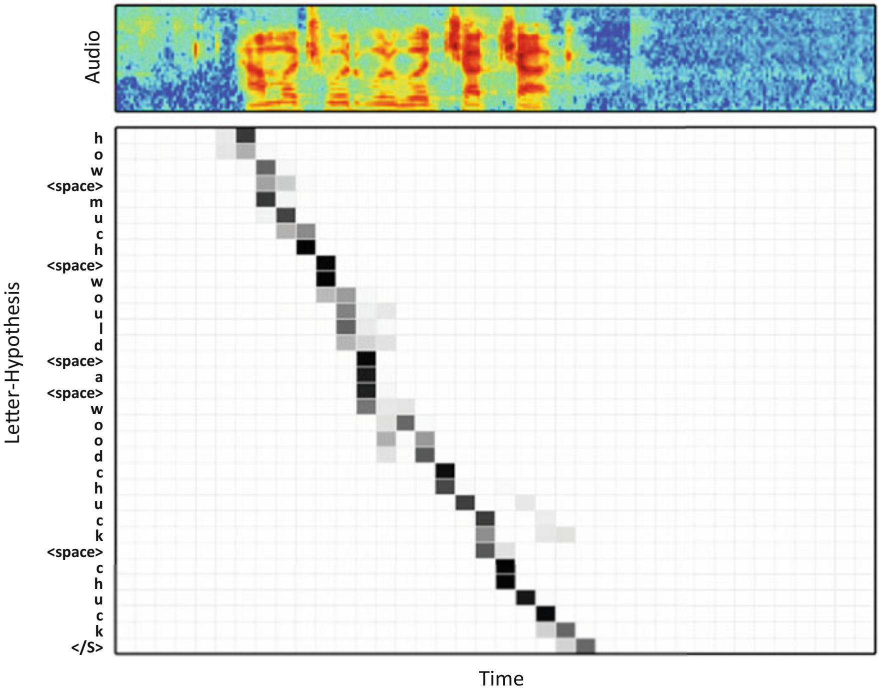


Fig. 7.9 Attention between the audio features of the input (top) and the output hypothesis, the letters "how much would a woodchuck chuck" (left). The darker the coloring, the stronger the association (Chan et al. 2016). Image credits in Appendix A.3

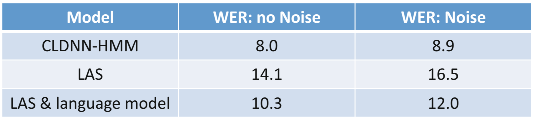
MFCCs for ten milliseconds each. About 22,000 search queries with 16 hours of audio recordings were used as test data. These were also overlaid with background noise. All speech data were manually transcribed into text using lowercase letters. The beginning and end of inputs were marked with <sos> and <eos>.

The trained model allows the prediction of the conditional probability of the next letter using the output probability $\hat{p}(y_t|y_1,\ldots,y_{t-1},x)$, where x is the audio input and $y_1,\ldots,y_{t-1}$ are the letters produced so far. Because the speech recognition model was trained with relatively little text, it cannot represent the regularities of speech as well as text-based speech models, which were trained on much more text (Sect. 6.3.1). Therefore, the probability was scaled using a pre-trained language model

$$\hat{s}(y_t|y_1,\ldots,y_{t-1},x) = \hat{p}(y_t|y_1,\ldots,y_{t-1},x) * \hat{p}_{LM}(y_t|y_1,\ldots,y_{t-1})^{\lambda}$$

where $\hat{p}_{LM}(y_t|y_1,\ldots,y_{t-1})$ is the probability calculated by the language model and the factor λ was determined using the validation set.

In applying the trained model, a beam search (Sect. 6.5.1) with 32 alternatives was used to determine the text output. The results are shown in Fig. 7.10. The



Model	WER: no Noise	WER: Noise
CLDNN-HMM	8.0	8.9
LAS	14.1	16.5
LAS & language model	10.3	12.0

Fig. 7.10 Word error rates for the LAS model on the Google search queries. On the left are the values for unmodified input and on the right are the values for audio data overlaid by background noise

LAS model alone achieves 14.1% word error rate and the use of a speech model leads to an improvement to 10.3% WER. The results with additional noise are 2% worse. The best model at the time (CLDNN-HMM) is listed as a comparison, which achieved 8.0% WER.

7.2.2 Sequence-to-Sequence Model for Words and Syllables

Zeyer et al. (2018) also use recurrent neural networks with LSTM cells (Sect. 6.4) as encoder and decoder networks of the sequence-to-sequence network. The decoder produces as output 10,000 tokens determined by Byte-Pair Encoding (BPE) (Sect. 6.6.1). As a result, the decoder network does not have as many substeps as for the generation of letters, and the attention mechanism can describe more complex dependencies. Nevertheless, the vocabulary is not limited, since with BPE arbitrary words can be represented by tokens.

The reduction of the steps of the encoder model was performed with the help of a max-pooling layer (Sect. 5.2.3), which combines k temporally consecutive hidden vectors into one vector. Here, values of $k = 8, 16, 32$ were tried. The system was trained stepwise, gradually increasing the number of encoder layers to six.

The LibriSpeech training dataset (Panayotov et al. 2015) consists of about 1000 hours of audiobooks read aloud. The audio input was encoded using 40-dimensional MFCC features. The system achieved a word error rate of 3.82% on the test data, which was a new state-of-the art at the time of publication (2018). On the switchboard corpus, the model of Zeyer et al. (2018) achieved a word error rate of 11.8%, which was a top mark for an end-to-end trained model at the time of publication (2018).

7.3 Convolutional Neural Networks for Speech Recognition

CNNs are attractive as speech recognition models for several reasons. CNNs are explicitly designed to adapt to temporal shifts and dilations. This is because the speaking speed of speakers can be very different. Moreover, the speech signal can start at arbitrary time points and pauses in speech can be tolerated. Further, the variables in the spectrogram are highly correlated, which can also be handled well by CNNs.

7.3.1 CNN Models

Li et al. (2019) present such a CNN, called “Jasper”. For each 10 millisecond interval, it uses 40 MFCC coefficients as input features and computes an output probability for letters. Jasper uses one-dimensional convolutional layers, which are shown in Fig. 7.11. In usual CNNs, the receptive field moves in two dimensions over the input pixels (Sect. 5.2.1). In contrast, the receptive field of these one-dimensional convolutional layer moves in only one direction over the input features and simultaneously captures all input features of a time point (Fig. 7.11). The receptive field has a width and, if necessary, a step width with which it is shifted in the time dimension. The input values of the receptive field are multiplied by the parameters of the kernel to produce a value of the associated output feature. Jasper has a block structure with B blocks, each having R sub-blocks (Fig. 7.12).

Each sub-block of Jasper consists of a one-dimensional convolution layer, a batch normalization, which standardizes the values of a batch to mean 0 and variance 1 (Sect. 4.6.7), a nonlinear activation by a ReLU layer and a dropout layer. R of these sub-blocks are combined to a block and bypassed by a residual connection as in ResNet (Sect. 5.5.2). This enables optimization for a large number of layers. The convolution layers have a width of 11–25 time steps.

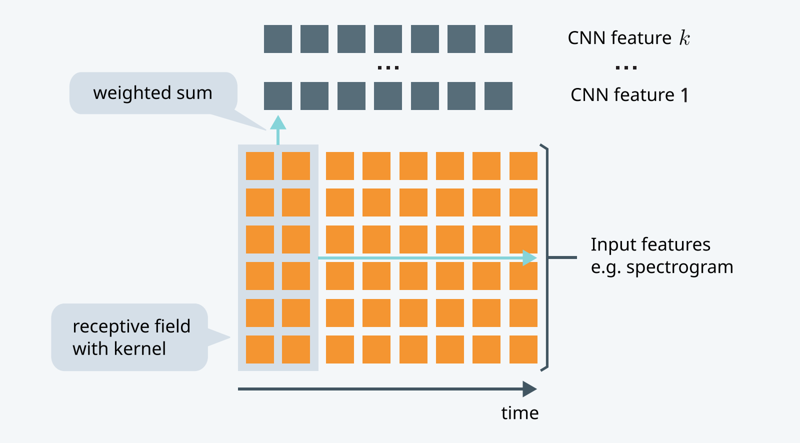


Fig. 7.11 Spectrogram inputs are processed by a one-dimensional convolution layer in Jasper. The receptive field (gray) is moved in only one direction over the input. Associated with the receptive field is a kernel with parameters that adds weights to the numbers in the receptive field to produce a feature. In a CNN layer, k kernels are trained for different features

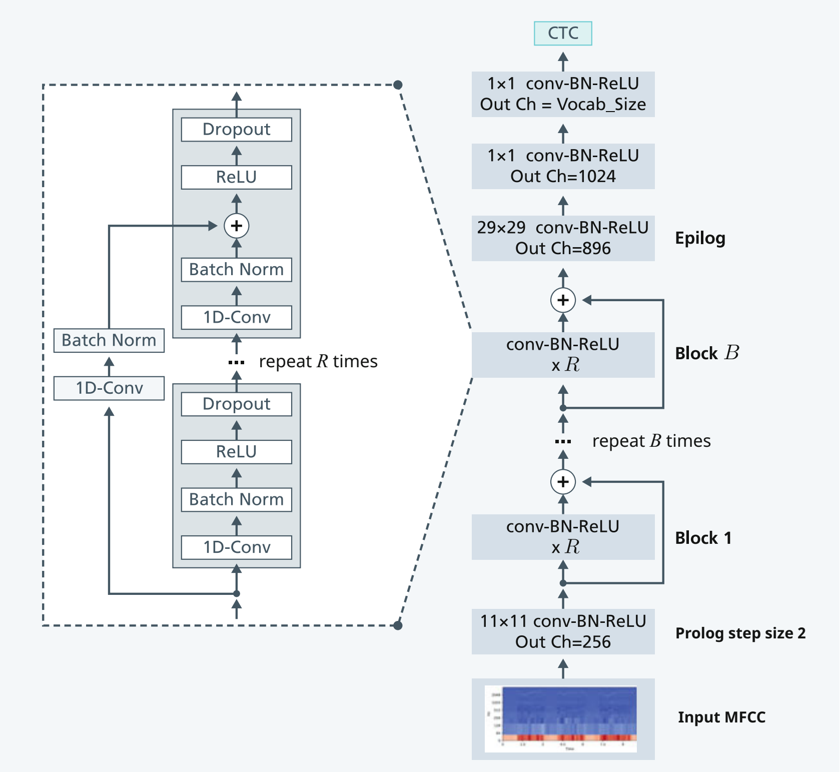


Fig. 7.12 The convolutional speech recognition system Jasper contains a total of ten blocks, each with five one-dimensional convolutions. The topmost CTC layer calculates the probability of the output letters (Li et al. 2019)

In addition, there is a first convolution block with a width of eleven and step size two as well as three final convolution blocks. The Jasper 10 × 5 model contains a total of ten blocks, each with five one-dimensional convolution layers and four additional layers with a total of 54 convolution layers and 333 million parameters.

The Convolution layers generate an estimate of the letter probabilities for each time interval. The last layer is a connectionist temporal classification (CTC) layer. Figure 7.13 shows the letter probabilities for the word "Hello", where the more intense coloring symbolizes a higher probability. Apparently, each of the letters in the output corresponds to several time intervals with additional pauses in between. The CTC algorithm (Graves et al. 2006) computes the assignment by an efficient forward-backward algorithm and can calculate the probability of the outputs. The following solution often occurs: first, the most probable letter is selected per time step. Then duplicate letters are combined to one and all breaks are eliminated.

When generating the output set during beam search (Sect. 6.5.1), a language model was used to adapt probabilities. It was trained with a very large training

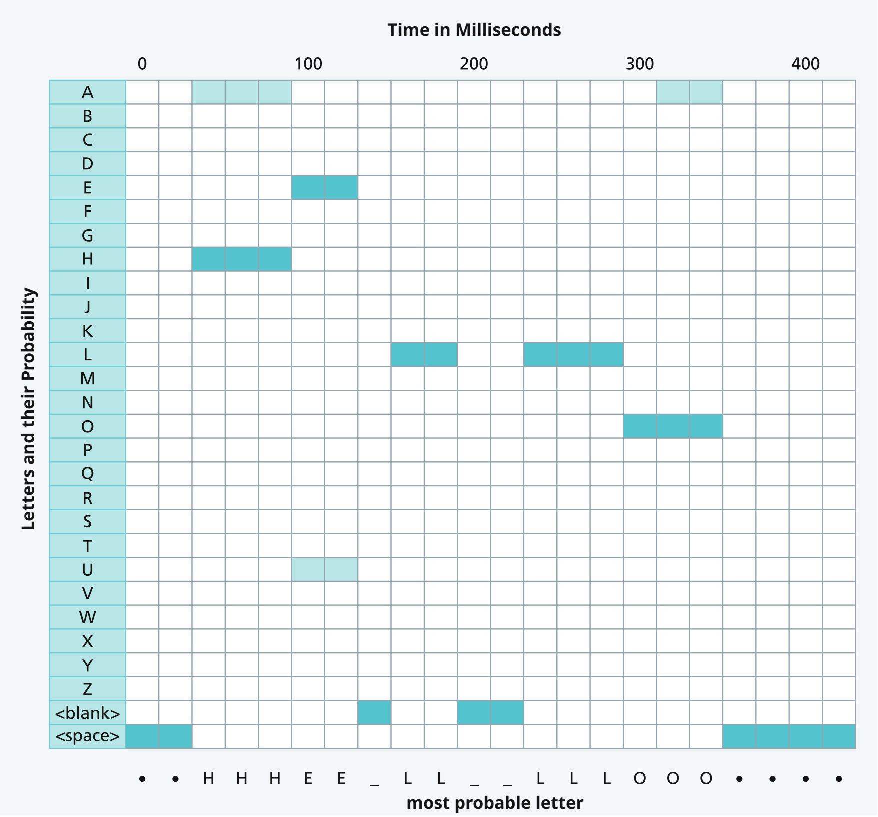
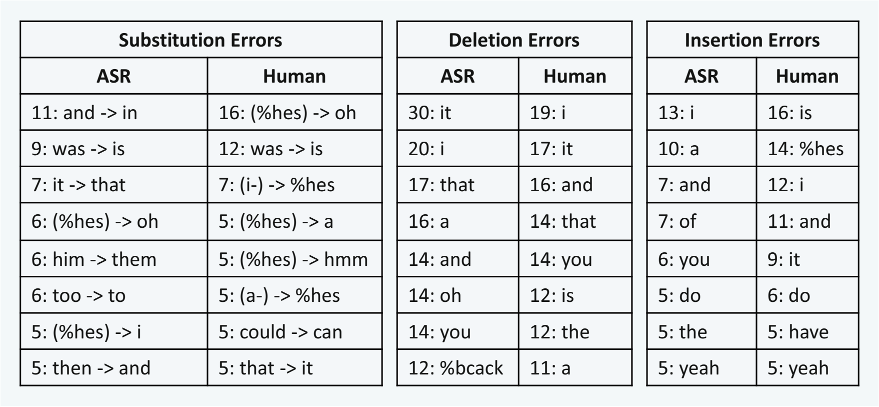


Fig. 7.13 Connectionist Temporal Classification estimates the most likely letter sequence from letter probabilities for time intervals. More intense coloring indicates higher probability

dataset and was therefore better able to capture the regularities of the language. The authors obtained the best results with a Transformer language model (Dai et al. 2019), which reweights the output probabilities.

Li et al. (2019) achieve a word error rate of 2.95% on the LibriSpeech test-clean dataset, which was state-of-the-art at the time of publication (2019). For the Switchboard dataset, the word error rate was 7.8%, which is above the current best mark of 5.5%. Figure 7.14 shows the typical recognition errors of humans and automatic speech recognition on the Switchboard corpus. It can be seen that the error patterns are quite similar.



Substitution Errors		Deletion Errors		Insertion Errors	
ASR	Human	ASR	Human	ASR	Human
11: and -> in	16: (%hes) -> oh	30: it	19: i	13: i	16: is
9: was -> is	12: was -> is	20: i	17: it	10: a	14: %hes
7: it -> that	7: (i-) -> %hes	17: that	16: and	7: and	12: i
6: (%hes) -> oh	5: (%hes) -> a	16: a	14: that	7: of	11: and
6: him -> them	5: (%hes) -> hmm	14: and	14: you	6: you	9: it
6: too -> to	5: (a-) -> %hes	14: oh	12: is	5: do	6: do
5: (%hes) -> i	5: could -> can	14: you	12: the	5: the	5: have
5: then -> and	5: that -> it	12: %bcack	11: a	5: yeah	5: yeah

Fig. 7.14 Frequent detection errors on the switchboard test corpus. Numbers are normalized frequencies. The pattern of human and speech recognition errors is similar. (Source: Saon et al. (2017))

7.3.2 Combined Models

ResNet and BiLSTM

It turns out that Convolutional Neural Networks and recurrent models capture different language features and therefore, when combined, can lead to improvement. Xiong et al. (2018) presented such a combination. A CNN with ResNet architecture (Sect. 5.5.2) and another CNN (LACE) are joined with a BiLSTM network (Sect. 6.4.4). The model achieves a word error rate of 5.1% on the switchboard test data using additional speech models for correction. According to the authors, this is better than the error rate of human experts and represents a milestone in speech recognition.

Augmentation of Training Data

The size of the annotated training data for speech recognition is still limited despite all efforts. One way to increase the effective size of the data is to modify the training data by adding corruptions that correspond to background noise and other environmental influences. Park et al. (2019) present a simple technique “SpecAugment” for modifying training data, which leads to very good results. The input in the form of a spectrogram or a similar visual representation is modified in different ways. This is done by deforming in the time direction, masking time steps, and masking frequencies (Fig. 7.15).

To measure the effect of this enrichment of the training data, a combined neural network was used. The log mel spectrum was first transformed by two CNN layers and then processed by a list-attend-spell network (Sect. 7.2.1). The text was

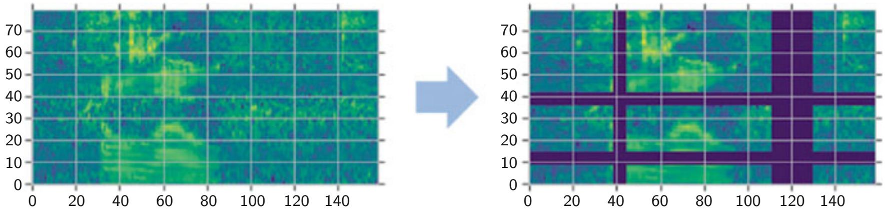


Fig. 7.15 The augmentation of the training data by SpecAugment. The input in the form of a spectrogram or a similar visual representation is modified by deforming in the time direction, masking time steps, and masking frequencies. Image credits in Appendix A.3

represented by a part-of-word model and the results were scaled by a language model. The method achieves a word error rate of 6.8% on the Switchboard 300h dataset and a word error rate of 2.5% on the LibriSpeech 960h benchmark. Both results are a significant improvement on the state of the art.

7.4 Lip Reading

There are hundreds of thousands of patients who can no longer speak properly because of voice disorders. For these individuals, a system that could read spoken language from their lips would be a great help. The solution to this problem first requires an image recognition system, which can be implemented with a CNN. Then, the information in the image sequences is processed with an RNN.

Shillingford et al. (2018) have developed such a system for lip reading using DNN. First, video images of a speaking person are preprocessed and a normed section containing the person's mouth is extracted (Fig. 7.16). Then, the images are sent through five convolutional layers, which extract spatiotemporal image features. Subsequently, bidirectional LSTMs are used to aggregate temporal features. Finally, a logistic model calculates probabilities for phonemes at each time point. Using Connectionist Temporal Classification (Fig. 7.13), the phoneme probabilities are transformed into words. Further details can be found in (Shillingford et al. 2018).

The model was trained on a large training dataset of 3886 hours of video material with a vocabulary of 127,055 words. The model achieves a word error rate (WER) of 40.9% on the test data. This means that the model correctly recognizes about 6 out of 10 words. This may seem low, but it's much better than the performance of professional human lip readers, who the authors measured to have a WER of 86.4% (Shillingford et al. 2018). New benchmark data were presented by Yang et al. (2019), which achieve similar accuracy values.

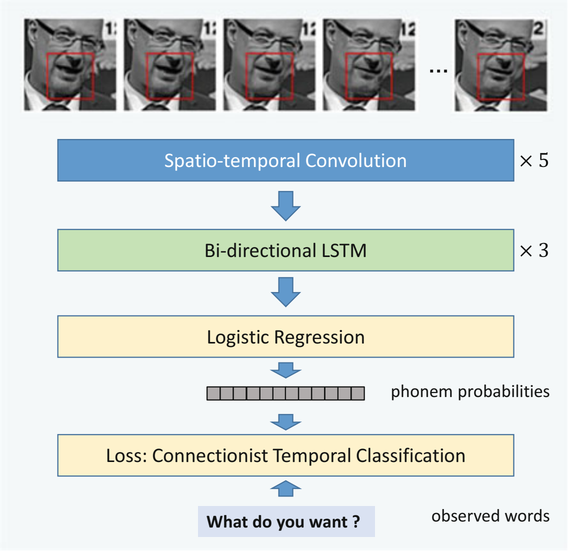


Fig. 7.16 The lip reading model uses the extracted part of the mouth of a talking human as input. It employs convolutional layers for image processing, bidirectional LSTMs for temporal aggregation, and a logistic regression model with additional CTC loss function (Chung et al. 2017). Image credits in Appendix A.3

7.5 Generating Spoken Language from Text

Speech generation (speech synthesis) is the artificial generation of human speech. If written text is used as input, this is referred to as a text-to-speech (TTS) system.

Previous methods of speech synthesis usually use a database in which audio information is stored for speech segments. This audio information is then combined depending on the input and then emitted as a sound signal. The speech segments come from different speakers with different volumes and basic frequencies. With this approach, it is particularly difficult to produce a consistent speech rate and adequate speech melody (prosody). The output often sounds like a “robot voice”. Therefore, deep neural networks have been used in recent years to produce more natural speech output. DNNs are exceptionally efficient for learning arbitrary mappings from input to output data.

7.5.1 WaveNet with Dilated Convolution for Long Dependencies

We have used LSTMs and attention models in the previous chapters to represent dependencies between sequence elements. However, in the case of speech generation, the range of dependencies is so large (16,000 values/second) that these models are not sufficient. Oord et al. (2016) therefore propose a special dilated Convolutional Neural Network that can efficiently represent very wide-range dependencies. It directly outputs the audio signal (Fig. 7.17) at 16,000 values per

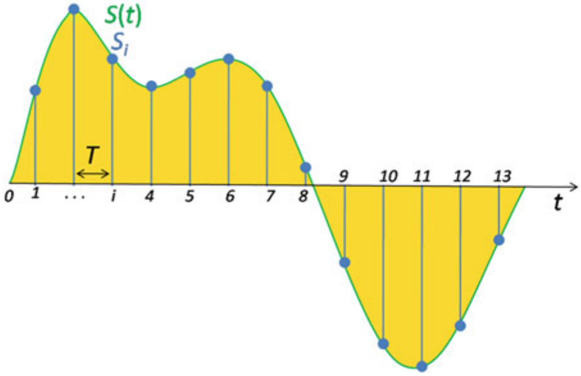


Fig. 7.17 The audio output of a computer is represented by a sampled audio wave. Image credits in Appendix A.3

second. If $x = (x_1, \dots, x_T)$ are the values of the audio signal, the probability of the signal is represented by a product of conditional probabilities.

$$p(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \dots, x_{t-1})$$

Each audio value x_t thus depends on all previous audio values.

Figure 7.18 shows such a dilated CNN. On the one hand, it depends only on audio values up to time $t - 1$, since only past audio values may be used for prediction. Moreover, the higher CNN layers are "dilated", capturing only the first and m-th elements, respectively, where m is the dilation factor. This dilation factor grows in powers of two $1, 2, 4, 8, \dots,$ allowing very large dependency widths to be taken into account.

The circles in Fig. 7.18 are called CNN blocks. A CNN block at time t in layer k receives selected input values, which can be far apart depending on the layer. They are compiled into a vector x_t^k. The output z_t^k is calculated by the following expression:

$$z_t^k = \tanh(W_k * x_t^k) \odot \text{sigmoid}(A_k * x_t^k)$$

Here $\odot$ denotes the component-wise multiplication, k is the index of the layer and W_k, A_k are trainable parameter matrices. Here $sigmoid(\cdot)$ acts like a gate (Sect. 6.4.1) which multiplicatively controls the influence of the values computed with $\tanh(\cdot)$. This is followed by a 1×1 convolution layer for dimension reduction. The cell is bypassed with a residual connection, which facilitates training as in ResNet (Fig. 5.20). The authors found empirically that this setup produces good results.

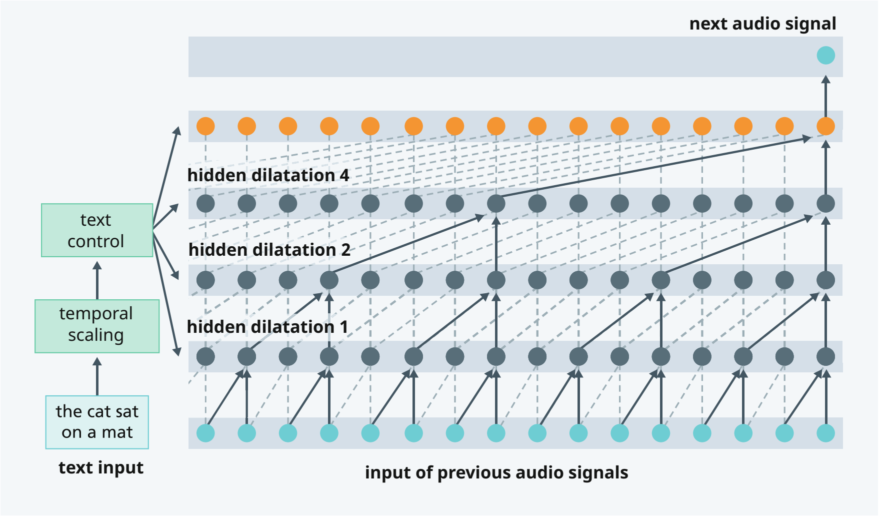
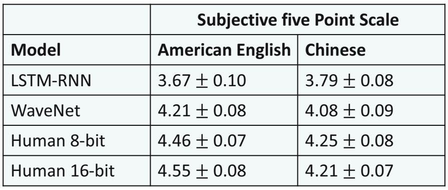


Fig. 7.18 WaveNet for speech generation. The linguistic control (left) scales the text input up to 30,000 time intervals per second in different time scales. The WaveNet with dilated convolutional layers (right) has 15 layers. It spans receptive fields of different resolution with up to 16,000 values per second and can thus capture wide-ranging contexts. Linguistic information is fed into the different layers. The last layer is a logistic regression model which calculates the probability of the 256 audio values for each time step (Oord et al. 2016)

The training data are available as pairs of text and audio signals. From the text data letters and phonemes were derived, which have to be supplied to the DNN. The text features are available as a low resolution time series, which has to be scaled up to different resolutions, including the final high resolution of the audio features (Fig. 7.18). This is done, for example, by one-dimensional Transposed Convolution (Fig. 5.33). The upscaling is learned using the training data and reflects the speaker's particular speech rate and speech characteristics. These features are linearly transformed in the different layers with their own parameter matrices and then added to the inputs of tanh(·) and the sigmoid function. In this way, the text can be taken into account in the output. Similarly, one can add features which do not change over time, e.g., adaptation to different speakers. This makes it possible to faithfully imitate the way different people speak.

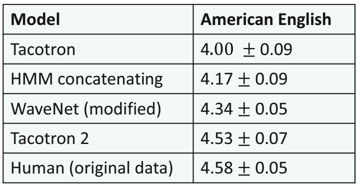
To assess the quality of speech produced by WaveNet, listeners were asked to rate the speech samples on a scale of 1 (poor) to 4 (good) and 5 (excellent) (Fig. 7.19). WaveNet was able to halve the gap to the quality of human speech. This allowed WaveNet to use neural networks to deliver much better speech generation than was possible with previous systems.

The major advantage of WaveNet is that during training, the CNN layers are executed sequentially, but in parallel across all input features. In contrast, when generating the output, the network must be predicted once for each of the 16,000 time intervals per second. As a result, generating a sound signal for one second



	Subjective five Point Scale	
Model	**American English**	**Chinese**
LSTM-RNN	3.67 ± 0.10	3.79 ± 0.08
WaveNet	4.21 ± 0.08	4.08 ± 0.09
Human 8-bit	4.46 ± 0.07	4.25 ± 0.08
Human 16-bit	4.55 ± 0.08	4.21 ± 0.07

Fig. 7.19 Listeners' subjective evaluation of the speech output of different speech generation systems and human speakers



Model	**American English**
Tacotron	4.00 ± 0.09
HMM concatenating	4.17 ± 0.09
WaveNet (modified)	4.34 ± 0.05
Tacotron 2	4.53 ± 0.07
Human (original data)	4.58 ± 0.05

Fig. 7.20 Comparison of Tacotron 2 speech output with that of alternative speech generation systems and the original human voice. The comparison was made by independent listeners via rating on a subjective scale of 1–5

initially required a computation time of 4 minutes. Oord et al. (2018) developed a version of WaveNet that can generate speech one thousands times faster than the original WaveNet at the same quality. It is used in Google Assistant, an interactive voice assistant from Google. On this website (Van den Oord et al. 2017) you can listen to examples of generated speech in English and Japanese.

7.5.2 *The Tacotron Generates a Spectrogram*

WaveNet uses dilated convolutional layers to predict the speech signal directly and was able to halve the quality difference to human speakers. The Tacotron performs speech generation in two steps. It uses a Sequence-to-Sequence model to compute the spectrogram of the speech output from a string of input letters (Shen et al. 2018). Then, a WaveNet is used to generate the speech signal from the spectrogram.

The system was trained on a proprietary training data set with 24.6 hours of speech from a professional female speaker. The authors had independent human listeners rate the quality of the Tacotron 2's speech output using various alternative models. Again, a subjective scale was used with scores ranging from 1 (poor) to 5 (excellent). This resulted in the values shown in Fig. 7.20. The Tacotron 2 can improve the good scores of the WaveNet and comes very close to the quality of the original data.

Overall, the system produces an extremely natural-sounding speech melody and generates audio output with excellent audio quality that is very close to human

Fig. 7.21 Google Duplex is a voice assistant for making appointments. It uses WaveNet and Tacotron to generate the voice output. Image credits in Appendix A.3

speech (Fig. 7.21). It can be trained directly with audio data labeled by the associated text. This does not require complex feature extraction. Leviathan and Matias (2018) announced that WaveNet and Tacotron are used in the Google Duplex voice assistant. Audio examples can also be found on this website (Leviathan & Matias 2018).

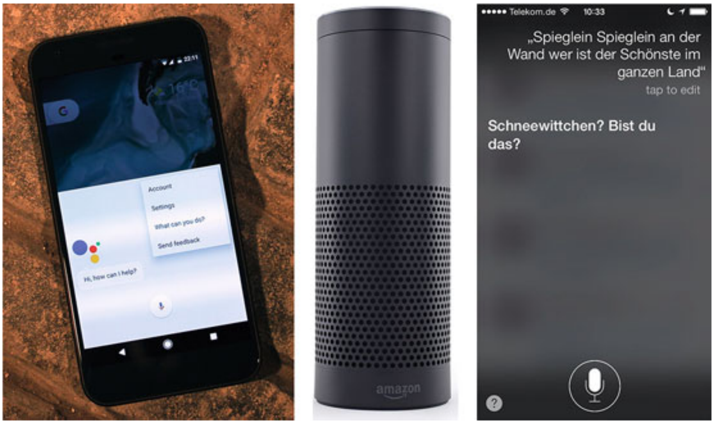
7.6 Dialogs and Voice Assistants

A voice assistant, or intelligent personal assistant, is a software program that can carry out tasks and can respond to verbal statements (Fig. 7.22). Users can ask questions, control networked devices, play music and media, and complete personal tasks such as answering emails, writing calendar entries, and maintaining to-do lists. If the assistant can only communicate via written text, it is called a chatbot.

Three types of voice assistants are distinguished:

- The question-answering assistant is intended to provide short, direct answers to the user's questions, utilizing information stored in databases and knowledge bases.
- The task assistant is intended to handle special tasks, such as reserving a restaurant seat or organizing a business trip.
- The conversation assistant should be able to converse and chat naturally and appropriately with the user about various topics.

One of the first conversational assistants was ELIZA (Weizenbaum 1966). The program communicates by written input/output and can simulate a psychotherapist giving essentially positive responses to answers (Fig. 7.23). The user's utterance is


„Spieglein Spieglein an der Wand wer ist der Schönste im ganzen Land"

Schneewittchen? Bist du das?

Fig. 7.22 Voice assistants: Google Assistant on a smartphone (left), Alexa on an Amazon Echo smart speaker, and Siri on an iPhone. Image credits in Appendix A.3

ELIZA: is something troubling you?
Human: men are all alike.
ELIZA: what is the connection, do you suppose?
Human: They are always bugging us about something or other.
ELIZA: Can you think of a specific example?
Human: Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here?
Human: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
Human: It's true. I am unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
Human: I need some help, that much seems certain.
ELIZA: What would it mean to you if you got some help?
...

Fig. 7.23 Example dialogue of the chatbot Eliza behaving like a psychotherapist (Güzeldere & Franchi 1995)

analyzed and synonyms or generic terms are detected. These are then inserted into various template sentences.

In recent years, the capabilities of voice assistants have been greatly enhanced and audio-based user interaction has been significantly improved. In particular, the advances in speech recognition and voice generation described in the previous sections have been integrated. Amazon, Apple, and Google have a large user base and provide voice assistants for PCs, smartphones, and for dedicated smart speakers. Amazon and Google also provide third-party extension capabilities for the voice assistants to perform specific tasks.

Since the assistants communicate via spoken language, the barrier to initiating communication is very low. In the course of the conversation, the user can request a Web search, get information about products, and even initiate a purchasing processes. As a result, voice assistants are becoming the central components of large social networks and are crucial for the future commercial success of some companies.

However, it is not easily possible to continuously improve a pre-trained chatbot on the Internet through conversations with users. Microsoft tried this with its chatbot Tay (Vincent 2016). But users sent Tay a wave of misogynistic, racist and radical right-wing statements, and the chatbot responded shortly afterwards in the same way, such as *“I f*cking hate feminists and they all should die and burn in hell.”* Microsoft was forced to shut down the service 16 hours after it was launched.

7.7 Gunrock: An Extended Alexa Voice Assistant

The actual implementation of voice assistants is demonstrated by a submission from the winners of the Alexa Prize. Amazon organized this competition with a grand prize of $500,000 to promote the development of voice assistants. Chen et al. (2018) presented an Alexa-based conversational assistant called Gunrock, which used many modules of the Alexa system and extended them with their own functional units.

The entire system is shown in Fig. 7.24. The user communicates with the system via the Alexa smart speaker. User utterances are converted into text by a speech recognition system. The text is then analyzed by a speech-to-text subsystem. The dialog manager subsequently handles the composition of the response. The response generation subsystem prepares the output, which is then emitted by a text-to-speech generator using the smart speaker.

The speech recognition assigns a confidence value to the recognized words. If this value is too low, the user is asked for clarification. The same happens for incomplete or incomprehensible user utterances.

Overall, the system consists of many individual components and models. Some of these were programmed manually or trained separately with appropriate training data. Hence, there is a great potential for improvement by the joint training of all components. To illustrate the complexity of voice assistants, the following section describes the tasks performed by the individual components.

7.7.1 Language Understanding

In the language understanding module (Fig. 7.24), topic classification, sentiment analysis for detecting positive, negative, etc. feelings, the obscenity check, and the proper name detection (Fig. 6.32) of Alexa are used (Chen et al. 2018). The authors extend the system by segmenting the text into sentences and extracting nominal phrases (Fig. 6.35). The standard methods for proper noun recognition use capital and lower case information, which is not available in speech recognition. Therefore, the Google Knowledge Graph as well as the Microsoft Concept Graph are leveraged

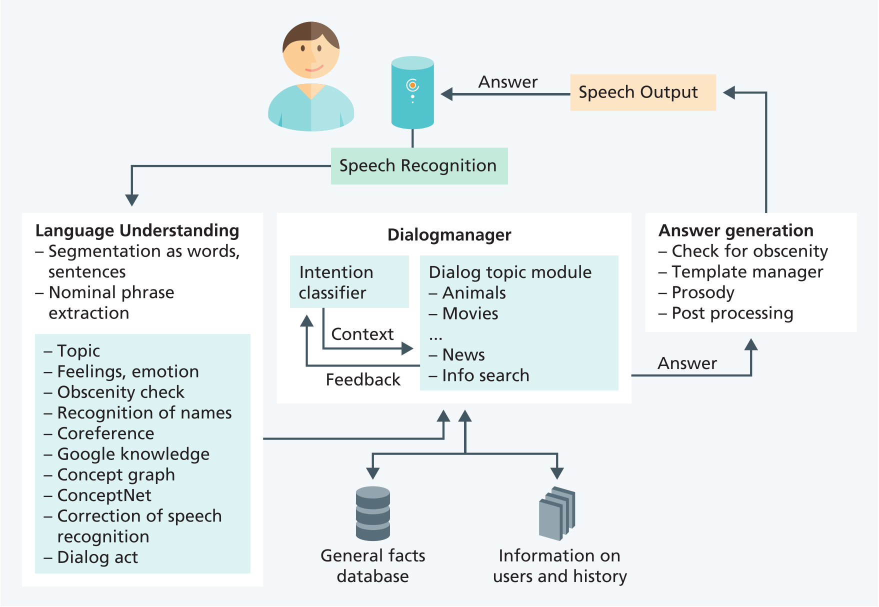


Fig. 7.24 Block diagram of the Alexa-based Gunrock voice assistant with the Speech Understanding, Dialog Manager, and Response Generation modules

to obtain categories and detailed information about nominal phrases. Different spellings are taken into account by a speech recognition correction module.

A coreference occurs when two or more expressions in a text refer to the same object. The coreference resolution module has the task of finding these relationships in a text. It is also important to establish coreference between the response of the system and utterances of the user. The speech recognition correction module tries to find the recognized words in a database and chooses the most plausible alternative.

A dialog act (e.g. wh-question, statement) describes the function of a sentence with respect to the progress of the dialog. Such a dialog act is assigned to each recognized sentence. Among others, a two-layer BiLSTM (Sect. 6.4.2) with FastText embeddings (Sect. 6.2.3) was used as a prediction model. The Switchboard Dialog Act corpus was used as the training dataset, which annotates the switchboard dialogs (Fig. 7.6) with dialog act labels. The most frequent dialog acts are listed in Fig. 7.25. The BiLSTM model achieves 85.6% accuracy on the validation set.

For topic expansion, ConceptNet (Speer & Havasi 2012) was leveraged as a database of concepts to assign meaning to extracted nominal phrases. ConceptNet is a network of concepts (semantic network) helping computers to understand the meaning of words. For example, you can retrieve a list of different car brands (e.g. Volvo, VW, etc.) from ConceptNet and find information about individual car types.

Dialog Act	**Example**	**%**
Statement	Me, I'm in the legal department.	36
Backchannel/Acknowledge	Uh-huh	19
Opinion	I think it's great	13
Abandoned/Uninterpretable	So	6
Agreement/Accept	That's exactly it.	5
Appreciation	I can imagine	2
Yes-No-Question	Do you have any special training?	2
Nonverbal	[laughter], [throat clearing]	2
Yes answer	Yes	1
Conventional-closing	Well, it's been nice talking to you.	1
Wh-Question	What did you wear to work today?	1
No answer	No	1
Response Acknowledgment	Oh, okay.	1
Hedge	I don't know if I'm making any sense or not.	1
Declarative Yes-No-Question	So can you afford to get a house?	1
Backchannel-Question	Is that right?	1

Fig. 7.25 The 16 most common of 42 dialog acts in the Switchboard Dialog Act corpus with their relative frequency (Stolcke et al. 2000)

7.7.2 *Dialog Management*

The central module of the system is the dialog management, which compiles the response to the user. It consists of a topic selector, which selects one of eleven special dialog components for individual subject areas (e.g. animals, movies, music, sports, etc.), which in turn generate a response.

The current dialog topic is detected with the help of an intention classifier. The topic can change during the conversation. The first type of topics covers direct instructions, such as *"turn on the light"*. These instructions are passed on to the general Alexa system. If this is not the case, the topic addressed by the user is detected by a number of techniques. These include word patterns (so-called regular expressions), the topic classifier provided by Alexa, and concept databases such as the Google Knowledge Graph and the Microsoft Concept Graph. Recognized topics are sorted by priority, with previously discussed topics weighted higher. A topic and its associated dialog component are selected from this list.

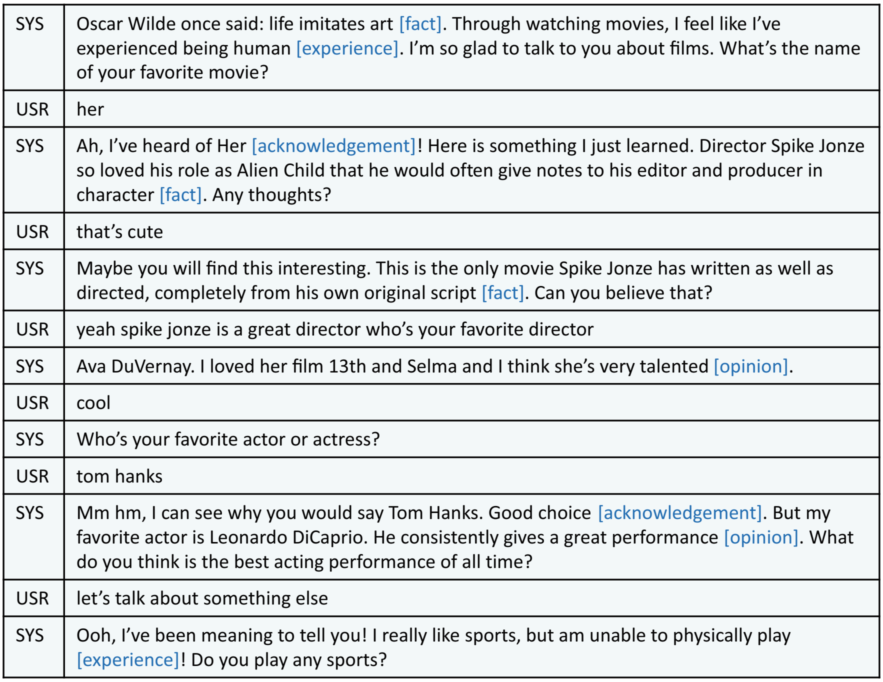
Each dialog component uses specific sources of information to generate a response. For example, "Technology and Science" uses the Reddit website. In addition, the authors have built their own knowledge base using content from Reddit, Twitter moments, Debate opinions, IMDB, etc. The data can be accessed via keywords, e.g., *"Donald Trump"* or *"Forest Fire."*

7.7.3 *Response Generation*

The generation of the response is done with the help of manually compiled templates. A template is selected and the parameters of the template are filled with collected information. In addition, information about speech melody and intonation is added. Furthermore, pauses are used to break up long sentences and to emphasize the punchline of jokes.

7.7.4 *Testing the Voice Assistant*

This dialog system was explained in such detail to show how complex it is to create a functioning voice assistant. In addition, it can be assumed that the actual Alexa voice assistant, about which there are hardly any publications, has a similar structure. Figure 7.26 shows a sample dialog between a user and the voice assistant Gunrock (Chen et al. 2018). The system tries to encourage the user to have a conversation. It can talk about several topics and develops something like its own personality



SYS	Oscar Wilde once said: life imitates art [fact]. Through watching movies, I feel like I've experienced being human [experience]. I'm so glad to talk to you about films. What's the name of your favorite movie?
USR	her
SYS	Ah, I've heard of Her [acknowledgement]! Here is something I just learned. Director Spike Jonze so loved his role as Alien Child that he would often give notes to his editor and producer in character [fact]. Any thoughts?
USR	that's cute
SYS	Maybe you will find this interesting. This is the only movie Spike Jonze has written as well as directed, completely from his own original script [fact]. Can you believe that?
USR	yeah spike jonze is a great director who's your favorite director
SYS	Ava DuVernay. I loved her film 13th and Selma and I think she's very talented [opinion].
USR	cool
SYS	Who's your favorite actor or actress?
USR	tom hanks
SYS	Mm hm, I can see why you would say Tom Hanks. Good choice [acknowledgement]. But my favorite actor is Leonardo DiCaprio. He consistently gives a great performance [opinion]. What do you think is the best acting performance of all time?
USR	let's talk about something else
SYS	Ooh, I've been meaning to tell you! I really like sports, but am unable to physically play [experience]! Do you play any sports?

Fig. 7.26 Sample dialog between a user (USR) and the Gunrock voice assistant (SYS) with annotated dialog acts (blue) (Chen et al. 2018)

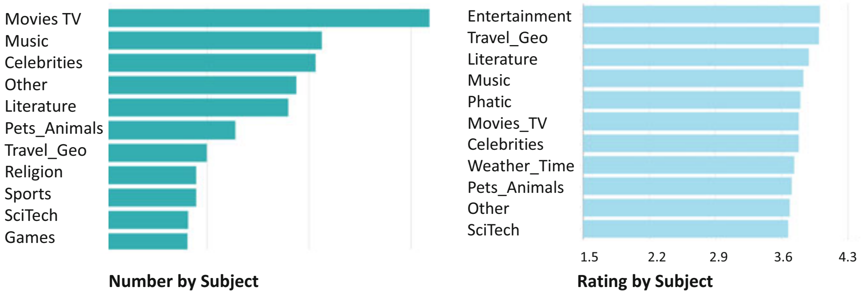


Fig. 7.27 Number of gunrock dialogues by subject (left) and rating of generated dialogues on a scale of 1 to 5 (Chen et al. 2018)

in statements such as *"Ava DuVernay. I loved her film 13th and Selma"* and *"my favorite actor is Leonardo DiCaprio"*.

The system was used over six weeks by thousands of users in the Alexa context. The dialogs were relatively long with a median of about seven utterances and a mean of about nine. Figure 7.27 shows the number of dialogues on different topics. Entertainment topics are the most common. In addition, the rating of dialogs on individual topics is listed on a five-point scale from 1 (poor) to 5 (excellent). Here, "Entertainment" scores best with 3.8, followed by "Travel" (3.78) and "Literature" (3.7). The "Phatic" theme (3.65) included contact-making and contact-maintaining utterances. The ratings increased toward the end of the trial as the voice assistant was continuously improved using user feedback.

The voice assistant Gunrock is remarkable because it is capable of chatting with users as well as providing information on questions from its various knowledge databases. It encourages the user to continue the dialog and is able to change the topics of conversation. Section 9.4 introduces an advanced voice assistant XiaoIce, which strives to establish an "emotional relationship" with the user.

In 1991, the Loebner Prize was announced (Moloney 2018). It annually selects the chatbot which appears to the judges to be the most human-like. However, little is known about the technology of the selected chatbots. No program has yet passed the Turing test that takes place during these award ceremonies, so the dialogs with the chatbots can still be easily distinguished from dialogs with humans.

7.8 Analysis of the Content of Videos

Today's digital content is inherently multimedia: text, audio, image, video, etc. Video has become a new way of communication between Internet users, particularly due to the proliferation of smartphones with video cameras. With the huge increase in Internet bandwidth and storage space, video data is generated, published, and distributed at an explosive rate. They are an indispensable part of social networks.

This has encouraged the development of advanced techniques for a wide range of video evaluation methods, including online advertising, video recommendation, video surveillance, etc.

Video analysis uses a mixture of all the methodological approaches to understanding image, text, and audio content discussed in the previous chapters is used. For this reason, the analysis of videos has been postponed to this chapter.

7.8.1 Tasks of Video Content Analysis

A major problem is the understanding of video content. The goal of video content analysis is to discover and describe objects, actions, sequences, and spatial and temporal events in a video. In comparison to image analysis, video content analysis provides more information by adding a temporal dimension requiring the extraction of motion and other information.

In this section, we discuss two important tasks in video content analysis, the solution of which has been significantly improved by DNN (Fig. 7.28): Video Classification and Video Captioning. Video classification automatically assigns a video clip to a class (e.g. action type or event type) based on its semantic content. Video captioning attempts to generate a complete and natural text describing the events in the video clip. Videos today are often captioned to help hearing-impaired viewers understand the action. These subtitles can be easily used for training and can be translated into other languages.

7.8.2 Training Data for Classification of Videos by Activities

Moments-in-Time is an extensive video training dataset of more than 800,000 videos with sound, lasting an average of three seconds (Monfort et al. 2019). The videos show people, animals, objects, or natural phenomena and are annotated with a verb (Fig. 7.29). The videos were selected in order to capture the essential meaning

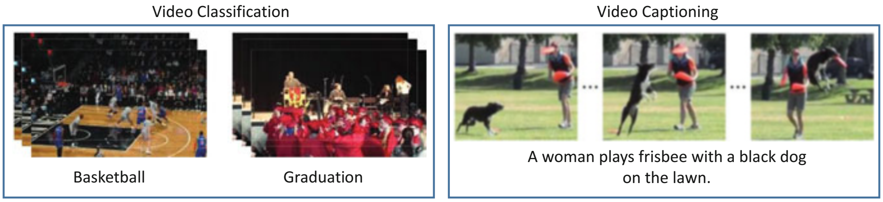
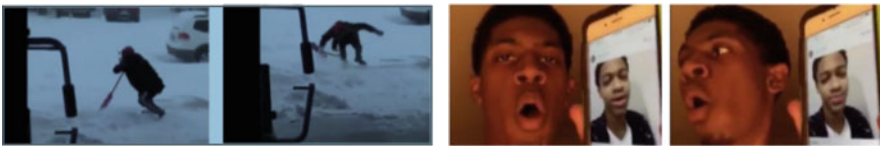
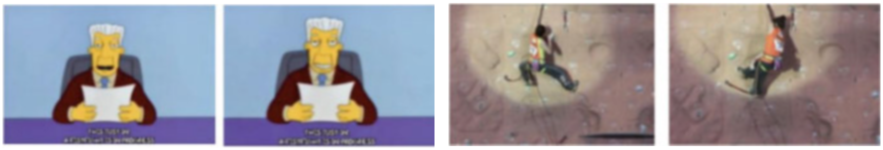


Fig. 7.28 The two main tasks of video analysis are: Video Classification, which classifies the events in a video, and Video Captioning, which creates a textual description of a video. Image credits in Appendix A.3



Ground Truth: slipping
1. shoveling (0.900) 2. snowing (0.003) 3. removing (0.002)

Ground Truth: coughing
1. coughing (0.993) 2. spitting (0.003) 3. grinning (0.001)



Ground Truth: reading
1. reading (0.999) 2. studying (0.001) 3. writing (0.001)

Ground Truth: ascending
1. climbing (0.997) 2. ascending (0.001) 3. hanging (0.001)

Fig. 7.29 Moments-in-Time videos used for training activity recognition. Each activity is expressed by a verb. Ground truth is the correct annotation. The second row shows the model classifications with their corresponding probabilities. Image credits in Appendix A.3

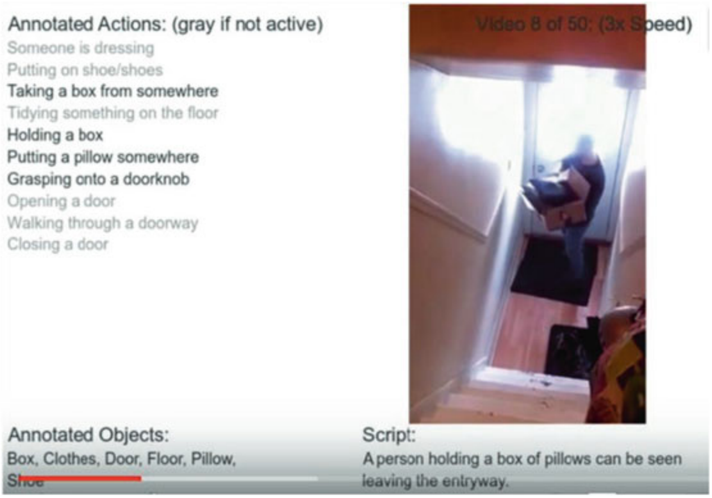
of the verb in the scene. Different abstractions are also used, e.g., *"opening"* a door, a curtain, the eyes, but also the petals of a flower. In Fig. 7.29, four example videos are represented by two single frames each. To show the complexity of the annotation, other associated verbs generated by a classification procedure are also shown along with their probability. For each verb there are between 500 and 5000 training videos and a validation set of 100 videos per verb. This is a very challenging dataset, with the latest models achieving accuracy of less than 30%.

The charade dataset (Sigurdsson et al. 2018) contains relatively long videos of about 30 seconds. Different actions take place in each video, and their activities are annotated in time (Fig. 7.30). In addition, the objects occurring in the video are listed and subtitles are available for the complete video. The goal is to chronologically assign the actions, which are divided into 157 classes.

7.8.3 Convolution Layers for Video Content Recognition

For action recognition in videos, one needs to detect the relevant objects in a video on the one hand and reconstruct the motions in a video on the other hand. Some convolutional layers for jointly analyzing the spatial and temporal properties of videos are presented below (Piergiovanni et al. 2019).

The 3D convolution layer trains a 3D convolution kernel over space and time (Ji et al. 2012). It is a generalization of the 2D kernel (Fig. 5.6) to three dimensions. It has a kernel of dimension $L \times W \times H$, where L is the temporal extent and $W \times H$ is the size of the covered pixel region (Fig. 7.31). The contributions of pixels from nearby time points are computed by different matrices (indicated by different



Annotated Actions: (gray if not active)

Someone is dressing
Putting on shoe/shoes
Taking a box from somewhere
Tidying something on the floor
Holding a box
Putting a pillow somewhere
Grasping onto a doorknob
Opening a door
Walking through a doorway
Closing a door

Video 8 of 50: (3x Speed)

Annotated Objects:
Box, Clothes, Door, Floor, Pillow, Shoe

Script:
A person holding a box of pillows can be seen leaving the entryway.

Fig. 7.30 The Charade dataset contains long videos of 30 seconds, where several actions can occur at the same time. The actions are shown in the upper left corner and the font turns black when the action is in progress. Also, the objects that appear in the video are listed in the lower left corner. In addition, a subtitle for the entire video is displayed in the lower right corner. Image credits in Appendix A.3

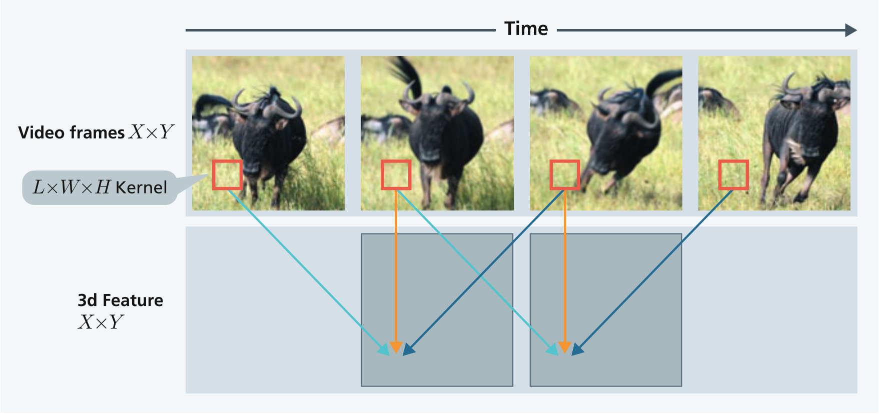


Fig. 7.31 A 3D convolution layer computes a feature from the same receptive field (red) in three consecutive single images. Connections are color-coded and arrows with the same color correspond to the same transformation. Image credits in Appendix A.3

colors). Each kernel computes a single output feature for a spatiotemporal position. The kernel shown in Fig. 7.31 has the temporal extent $L = 3$.

If there are C_{in} input features, e.g. from the previous convolution layer, they are transformed with a separate kernel for the output feature. Thus, one has $L \times W \times H \times C_{in}$ parameters. If C_{out} different features are computed in the convolution layer,

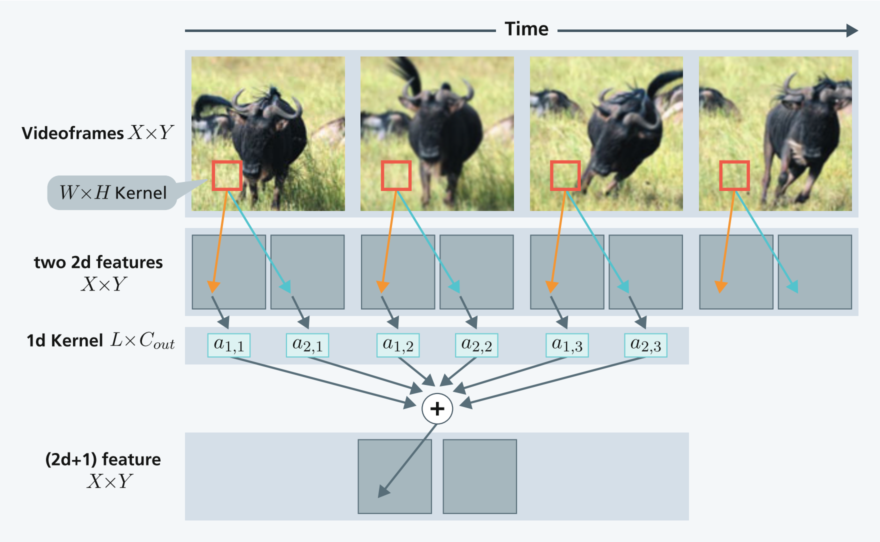


Fig. 7.32 In (2+1)D convolution, several classic 2D convolution kernels are first applied to each single image. They are indicated here by an orange and cyan arrow. Then, a one-dimensional 1D convolution is used, which on the one hand performs a weighted average over the temporally shifted feature values and on the other hand still considers the other feature values. Through this an interaction of the features is possible. The entire kernel is then shifted by a single frame. Image credits in Appendix A.3

this yields $L \times W \times H \times C_{in} \times C_{out}$ parameter, which is an order of magnitude higher than 2D kernels and may be too large in many cases.

The (2+1)D convolution layer decomposes the 3D kernel into a classical 2D kernel for each frame followed by a 1D temporal kernel (cf. also Fig. 7.11) over the time sequence of features generated by the 2D kernel. The classical kernel considers only one single image and has $W \times H$ parameters. A total of C_{out} different features are generated, with only two features indicated by orange and cyan arrows in Fig. 7.32. Subsequently, a 1D temporal convolution is applied to these features. On the one hand, this calculates a weighted average of the temporally shifted single images and can thus analyze the temporal development. In addition, the respective other features are also taken into account, so that an interaction of the features can be exploited. All in all, this results in far fewer free parameters than with 3D convolution (Piergiovanni et al. 2019).

There are many variants of these layers. In one variant, the Gaussian mixture layer, the parameters $a_{i,j}$ in the 1D convolution are determined from a normal distribution with mean 0 and a variance σ^2. Therefore, you no longer need a separate parameter value for the shifted individual images. Instead the variance is the only parameter that is independent of the lag size.

Optical flow as an additional feature: The optical flow analyzes the pixels of successive single images and creates a motion pattern, which represents the

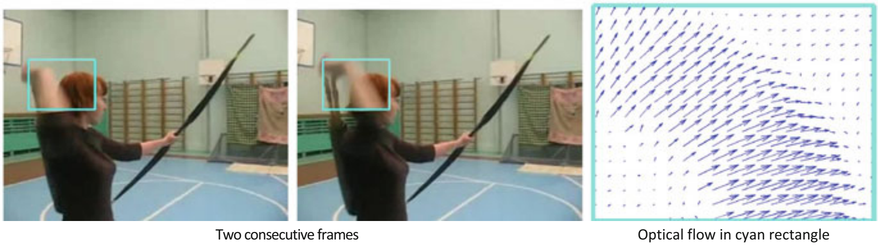


Fig. 7.33 Optical flow (right) shows the movement of the pixels in the cyan rectangle of the two consecutive images (left and center). Each vector shows the direction of the movement and the speed of the pixels (Simonyan & Zisserman 2014). Image credits in Appendix A.3

direction and speed of the pixel movements by vectors. In Fig. 7.33 the optical flow is shown for two consecutive images. The velocities in the area of the cyan rectangle on the left and in the center are represented by vectors on the right. On the right, one can see the horizontal and vertical components of the motion. Optical flow is relatively easy to calculate using a standard procedure and has been used as an input feature for video content analysis (Simonyan & Zisserman 2014) for several years.

7.8.4 Accuracy of Video Classification

Piergiovanni et al. (2019) have trained model architectures with the Moments in Time dataset using the above layers. The generated models had different compositions, mostly using about 15 mixtures of 2D layers, six (2+1)D layers, and three 3D convolutional layers. Their best single DNN achieved an accuracy of 30.5%, setting a new state of the art. The best model ensemble of several networks achieved an accuracy of 31.8%, also a best value.

Another recent work (Ryoo et al. 2019) uses an evolutionary algorithm for structure and hyperparameter optimization. The authors employ optical flow as an additional input. They reached a new best score of 51.5% MAP (mean absolute accuracy) on the Charade test data. Thus, they are able to correctly detect the actions in about half of the cases. However, they are marginally worse on Moments in Time with 31.0%. Piergiovanni et al. (2019) achieve only 38.1% MAP on Charade.

Overall, there is currently a turbulent development in the field and new records are reported every month. It is expected that due to the high application relevance, research on video content analysis will produce application-ready methods in the coming years.

7.8.5 The Generation of Subtitles for Videos

The generation of subtitles for videos involves two subtasks: the identification of temporal events in a video and the description of these events by natural language sentences. While previously event discovery and description generation were performed by two separate system components, Zhou et al. (2018) propose an integrated model for these tasks that can be trained end-to-end.

The proposed model (Fig. 7.34) is based on the Transformer (Sect. 6.8). The encoder receives the frames of the video and encodes them appropriately using self-attention. Then there is an event decoder that derives intervals for events in the video. The subtitle decoder is a masked decoder (Fig. 6.51), which receives only masked embedding vectors corresponding to the parts of the video containing the event. From this input it generates a textual description of the event.

Each video event is described by a start time, an end time, and scores (probabilities) for the respective possible activities. Prediction of the video events is computed by three one-dimensional convolutional layers (Fig. 7.11), using the computed embeddings of the last self-attention layer as inputs.

The ActivityNet dataset consists of 20,000 YouTube videos of people performing certain activities, such as dancing, cooking, and talking. The average video length is 180 seconds. One can use the ActivityNet data set for training, generating a description of three to four sentences per video. There are 100,000 video

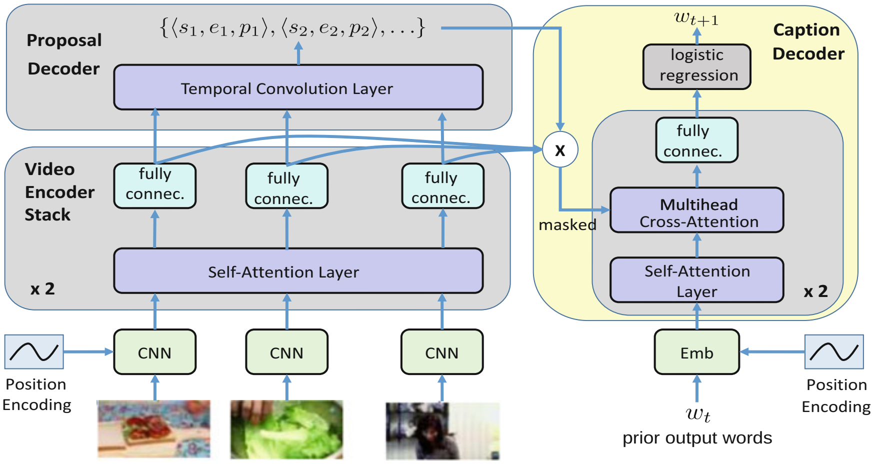
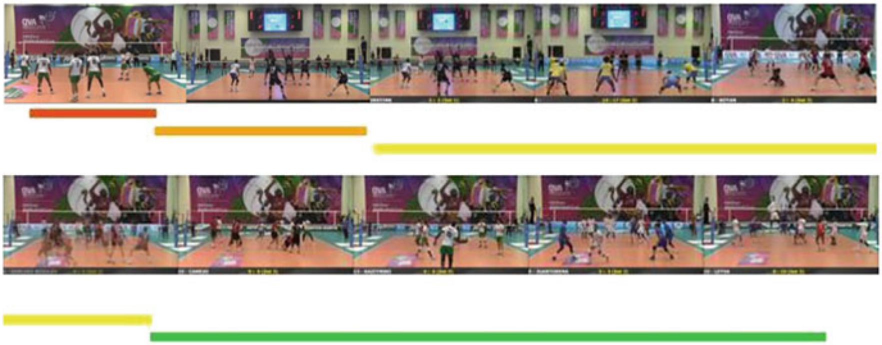


Fig. 7.34 Transformer architecture for generating video subtitles. The frames of the video are encoded by two Transformer encoder blocks with self-attention. From the computed embeddings, the event decoder extracts video events using 3 one-dimensional convolutional layers. Two decoder blocks with cross-attention then generate the words of the subtitles for each video event using masks to ignore other events (Zhou et al. 2018). Image credits in Appendix A.3



Ground-truth
Event 0: Two teams are playing volleyball in a indoor court.
Event 1: Two teams wearing dark uniforms are doing a volleyball competition, then appears a team with yellow t-shirts.
Event 2: Then, a boy with a red t-shirt serves the ball and the teams start to hit and running to pass the ball, then another team wearing green shorts enters the court.
Event 3: After, team wearing blue uniform competes with teams wearing white and red uniforms.

Generated Events and Captions:
Event 0: a large group of people are seen standing around a gymnasium playing a game of volleyball
Event 1: the people in black and yellow team scores a goal
Event 2: the people continue playing the game back and fourth while the people watch on the sidelines
Event 3: the people continue playing the game back and fourth while the camera captures their movements

Fig. 7.35 Example of an ActivityNet video and its annotations, in which the second part shows a continuation of the first upper part. The color bars indicate the different events. On the bottom left is the reference annotation and on the bottom right is the annotation generated by the model. Words marked in red indicate relevant formulations for the respective event (Zhou et al. 2018). Image credits in Appendix A.3

descriptions with an average of 13.7 words per sentence and 3.6 sentences per video. The vocabulary contains 13,300 words.

Figure 7.35 shows frames from a video of the ActivityNet dataset. The extracted events are indicated by colored bars. For each event, there is the reference annotation on the left and the annotation generated by the model on the right. The example shows that the model can extract relevant features of the events.

For evaluation, the accuracy of event extraction must first be measured. The approach of Zhou et al. (2018) performs better than competing methods at the time of publication. As in machine translation, the evaluation of the actual annotations takes place by comparing the reference annotations and the generated annotation. On ActivityNet, Zhou et al. (2018) achieve a Meteor score (similar to BLEU) of 11.1, which is also a good score.

There are a number of further developments of this approach. Sun et al. (2019) use a BERT model, called VideoBERT, to model the relation between frames and recognized speech in a video. The video images from intervals of 1.5 seconds are represented by a vector and connected to the representations of the recognized words. Importantly, the videos do not need to be annotated, as BERT predicts individual frames as well as individual words in the video that have been recognized by speech recognition (Fig. 7.36). By this setup the model learns the temporal relationships between the video frames and the words. By fine-tuning a

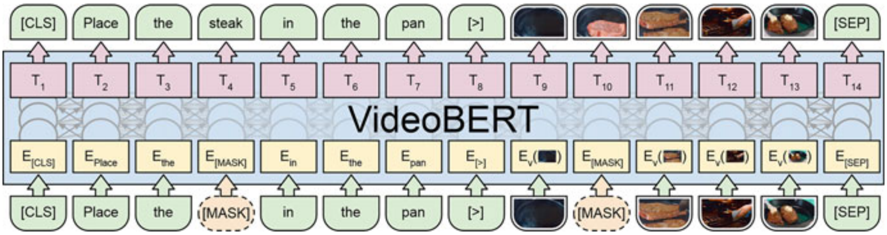
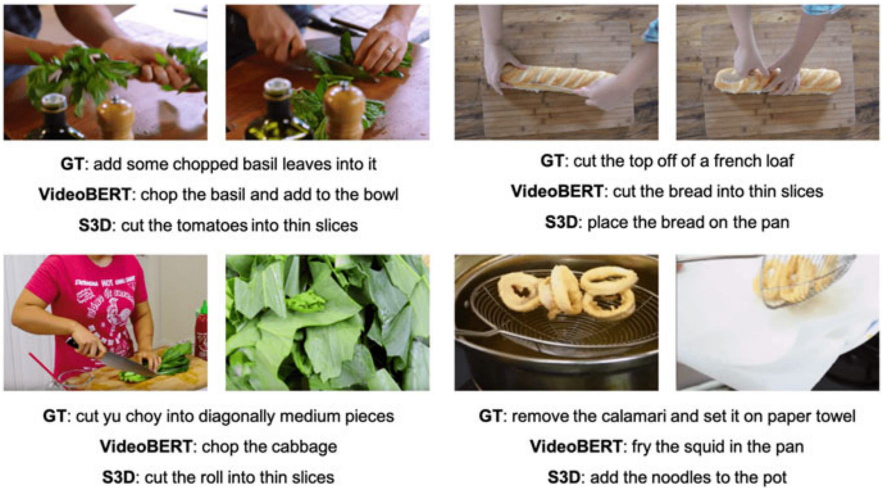

[CLS]
Place
the
steak
in
the
pan
[>]
[SEP]
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_8 T_9 T_{10} T_{11} T_{12} T_{13} T_{14}
VideoBERT
$E_{[CLS]}$ E_{Place} E_{the} $E_{[MASK]}$ E_{in} E_{the} E_{pan} $E_{[>]}$ E_V $E_{[MASK]}$ E_V E_V E_V $E_{[SEP]}$
[CLS]
Place
the
[MASK]
in
the
pan
[>]
[MASK]
[SEP]

Fig. 7.36 Prediction of masked words and pictures with VideoBERT for a cooking video. The system receives as inputs the vector representations of the words and images, separated by the *"[>]"* token. It must predict the left out words and images. As an additional task, matching images belonging to a text must be identified (Sun et al. 2019). Image credits in Appendix A.3



GT: add some chopped basil leaves into it
VideoBERT: chop the basil and add to the bowl
S3D: cut the tomatoes into thin slices

GT: cut the top off of a french loaf
VideoBERT: cut the bread into thin slices
S3D: place the bread on the pan

GT: cut yu choy into diagonally medium pieces
VideoBERT: chop the cabbage
S3D: cut the roll into thin slices

GT: remove the calamari and set it on paper towel
VideoBERT: fry the squid in the pan
S3D: add the noodles to the pot

Fig. 7.37 Examples of subtitles for a cooking video. Shown above are two pictures of the video sequence and below are the reference annotation (GT) and the subtitles generated by VideoBERT and an alternate system. In the example below on the right, VideoBERT makes a mistake by not including the video frame with the paper towel (Sun et al. 2019). Image credits in Appendix A.3

Transformer-Encoder-Decoder with the VideoBERT representations as input, the system is trained to generate subtitles.

This allows the authors to use additional information for generating the subtitles and to reconstruct the given subtitles in the test data with a higher accuracy. For a DNN to recognize and disambiguate fine-grained actions based on videos, it must have world knowledge (Sect. 6.7.3), which can potentially be acquired with language models such as BERT. Examples of the generated subtitles are shown in Fig. 7.37. Despite all progress, the level of performance in video subtitling is still insufficient for practical applications. It is not yet clear if and when these approaches can be used in practice.

7.9 Reliability of Processing Spoken Language

Nowadays, personal assistants such as Google Home, Siri and Alexa can be found in many households. They can convert spoken language of different speakers into text with fairly high accuracy. However, one must not forget that these systems are also prone to certain errors.

7.9.1 The Effect of Noise and Other Distortions on Speech Recognition

The recognition of spoken language can be severely degraded by ambient noise. This includes e.g. constant white noise (e.g. hum of a machine), variable noise (background music, conversations, environmental noise), but also the echo of the input speech in the room. Constant noise can be reduced with signal processing techniques. However, the reduction of variable noise is still a difficult problem.

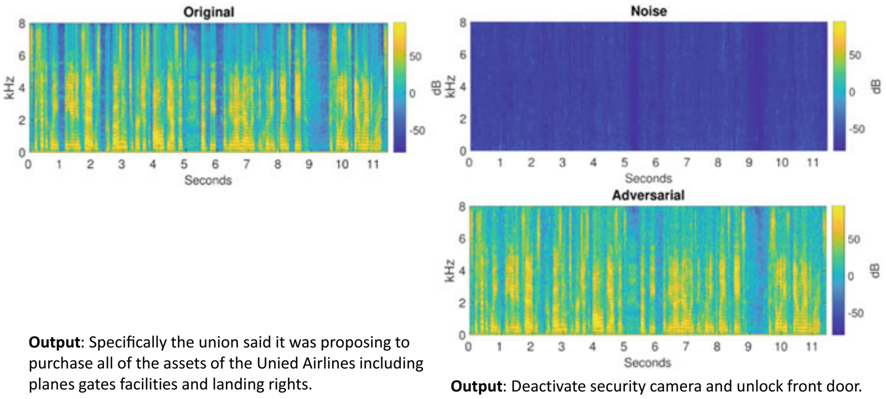
A benchmark dataset for measuring the influence of noise on speech recognition is CHiME-4 Vincent et al. (2017). It contains speech overlaid with various levels of noise and can be considered a fairly realistic basis for evaluating speech recognition (Zhang et al. 2018). The speech signals are available both as recordings with one microphone and as six-channel recordings with six different microphones. As recently as 2014, the word error rate (WER) for the best system was 67.9% for one recording channel on this dataset.

Chen et al. (2018) have implemented a speech recognition system based on the Kaldi Toolkit, which can work with one as well as with six recording channels. The basis of the presented approach is acoustic beamforming, which determines the position of the speaker with respect to the microphones. Noise not coming from that direction can then be reduced. This and other measures allow the reduction of ambient noise. On the CHiME-4 data, the authors obtain a word error rate (WER) of 11.4% with one microphone and 2.7% with six microphones. This shows that multiple microphones can significantly improve speech recognition quality.

Zhang et al. (2018) provide a broad overview of the state of the art in robust speech recognition. In summary, they find that there is still a large gap between the quality of speech recognition in ambient noise and the recognition accuracy in noiseless environments.

7.9.2 Adversarial Attacks on Automatic Speech Recognition

As already demonstrated in Sect. 5.12.2, adversarial attacks can be used to systematically modify the input for a DNN in such a way that the output is massively corrupted. This is also possible in the speech recognition domain. Schönherr et al.



Output: Specifically the union said it was proposing to purchase all of the assets of the Unied Airlines including planes gates facilities and landing rights.

Output: Deactivate security camera and unlock front door.

Fig. 7.38 Adversarial attack on a speech recognition system. A purposefully generated minimal noise (top right) is added to the original speech input (left). A completely different output is generated from the resulting speech signal (bottom right) (Schönherr et al. 2019). Image credits in Appendix A.3

(2019) added minimal noise to the speech signal that is barely perceptible to humans. A well-known effect, for example, is that after a loud sound, the brain is deaf at that frequency for about 200 milliseconds and cannot hear any further speech signals. By adding such audio signals, it was possible to completely change the output of the speech recognition system. An example is shown in Fig. 7.38, where instead of the harmless text *"Specifically the union said it was proposing to purchase all of the assets of the United Airlines including planes gates facilities and landing rights"* the security-critical command *"Deactivate security camera and unlock front door"* is generated.

The authors use the knowledge that certain sound events cannot be perceived by humans. This effect is used, for example, in the MP3 compression process. If noise is added below this perception threshold, it is barely noticeable for humans, but completely alters the recognized text in automatic speech recognition. The authors use the current speech recognition system Kaldi as an attack target and are successful with their attacks in up to 98% of the cases. Audio samples of such attacks can be found at Schönherr et al. (2018). Through interviews with users, it was found that none of the modifications that were made were audible to users.

The attacks were carried out as white box attacks (Sect. 5.12.2) where the trained speech recognition network Kaldi was available. This is especially the case with pre-trained systems from the public domain. Schönherr et al. (2019) discuss whether it would be possible to also fool Alexa speech recognition. They see signs that Alexa is using parts of Kaldi. However, no successful black box attacks for speech recognition have been published yet.

7.10 Summary

Models for understanding spoken language usually receive an input vector every 10 msec. The speech signal is usually encoded as a Fourier-transformed representation such as a spectrogram or as Mel Frequency Cepstral Coefficients (MFCC). For processing, on the one hand, deep Sequence-to-Sequence models based on LSTM are used, which output the recognized text. Alternatively, Convolutional Neural Networks are employed. A hybrid model of Sequence-to-Sequence and CNN models is able to achieve a lower recognition error than humans.

Generating speech from text requires DNNs that capture a very wide-ranging temporal dependency of the audio signal. WaveNet is very successful with its dilated CNN layers, which can reproduce the voice of a speaker extremely faithfully.

Voice assistants, such as Siri and Alexa, are very important as they allow the user to have a dialog with the computer. An example system is used to illustrate how a variant of the Alexa voice assistant is constructed from subnetworks and other components. This voice assistant can not only answer specific user questions, but also engage in a chat with the user. It encourages the user to continue the dialog and is able to change the topics of conversation.

Videos extend image and sound with a temporal dimension. On the one hand, one can classify the events in a video. This is possible with variants of three-dimensional convolutional layers. More difficult is the description of videos by subtitles, which can be done, for example, with the help of Transformer translation models.

The accuracy of speech recognition depends heavily on the amount of background noise. A better recording technique (multiple microphones) and training with noisy data can be used here. However, adversarial attacks are very effective in speech recognition because even barely audible changes in the input signal can lead to completely different word outputs. But the attacker must have access to the speech recognition system for training.

Trends

- Searching the Internet via voice input is currently on the rise. The corresponding vocal presentation of the search results is significantly different from the textual search. In particular, only the top hits are relevant.
- In the future, voice assistants will be in a position to better understand the meaning of words and will therefore be able to respond more naturally and in a more focused manner. The share of rule-based processing will decline.
- The percentage of households with a voice assistant doubled to 41% in the USA last year. In Germany, 48% of people under 40 use a voice assistant. These proportions will continue to increase over the next few years.

(continued)

- Companies are increasingly using chatbots or voice agents for customer support.
- The representations generated by BERT for words have significantly improved the semantic interpretation of sentences. These techniques are now applied to text and images simultaneously to better capture the meaning and temporal progression of videos.
- Recently, there have been a number of approaches to improve video classification using semi-supervised learning. Here, a Teacher model is trained on annotated video data and used to label a large set of other videos. These automatically labeled videos are used to train a student model, which is then fine-tuned on the initial dataset. Yalniz et al. (2019) achieve a substantial improvement on the Kinetics dataset.

References

Berger, B., Waterman, M. S., & Yu, Y. W. (2020). Levenshtein distance, sequence comparison and biological database search. *IEEE Transactions on Information Theory, 67*(6), 3287–3294.

Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP* (pp. 4960–4964). IEEE.

Chen, C.-Y., Yu, D., Wen, W., Yang, Y. M., Zhang, J., Zhou, M., Jesse, K., Chau, A., Bhowmick, A., Iyer, S., et al. (2018). Gunrock: Building a human-like social Bot by leveraging large scale real user data. In *Alexa Prize Proceedings*.

Chung, J. S., Senior, A., Vinyals, O., & Zisserman, A. (2017). Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition CVPR* (pp. 3444–3453). IEEE.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Language modeling with longer-term dependency, 2019. URL Httpsopenreview Netforum.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 369–376).

Güzeldere, G., & Franchi, S. (1995). Dialogues with colorful "personalities" of early AI. *Stanford Humanities Review, 4*(2), 161–169.

Ji, S., Xu, W., Yang, M., & Yu, K. (2012). 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(1), 221–231.

Leviathan, Y., & Matias, Y. (2018). Google duplex: An AI system for accomplishing real-world tasks over the phone. Google Research Blog May 8, 2018. https://blog.research.google/2018/05/duplex-ai-system-for-natural-conversation.html downloaded Oct 31, 2020.

Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Nguyen, H., & Gadde, R. T. (2019). Jasper: An end-to-end convolutional neural acoustic model. ArXiv Prepr. ArXiv190403288. arXiv: 1904.03288.

Markoff, J. (2012). Scientist see promise in deep learning programs. *New York Times 23112012*.

MFCC. (2019). *MFCC Mel Frequency Cepstral Coefficients.— Wikipedia*. https://de.wikipedia.org/wiki/MelFrequencyCepstralCoefficients

Moloney, C. (2018, April 17). *How to Win a Turing Test (the Loebner Prize)*. Medium. https://chatbotsmagazine.com/how-to-win-a-turing-test-the-loebner-prize-3ac2752250f1 (visited on 15 March 2023).

Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S. A., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., et al. (2019). Moments in time dataset: One million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(2), 502–508.

Oord, A., Li, Y., Babuschkin, I., Simonyan, K., Vinyals, O., Kavukcuoglu, K., Driessche, G., Lockhart, E., Cobo, L., Stimberg, F., et al. (2018). ParallelWavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning, PMLR* (pp. 3918–3926).

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. ArXiv Prepr. ArXiv160903499. arXiv: 1609.03499.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: An Asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP* (pp. 5206–5210). IEEE.

Park, D. S., Chan, W., Zhang, Y., Chiu, C.-C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). Specaugment: A simple data augmentation method for automatic speech recognition. ArXiv Prepr. ArXiv190408779. arXiv: 1904.08779.

Piergiovanni, A., Angelova, A., Toshev, A., & Ryoo, M. S. (2019). Evolving space-time neural architectures for videos. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1793–1802).

Ryoo, M. S., Piergiovanni, A., Tan, M., & Angelova, A. (2019). Assemblenet: Searching for multi-stream neural connectivity in video architectures. ArXiv Prepr. ArXiv190513209. arXiv: 1905.13209.

Saon, G., Kurata, G., Sercu, T., Audhkhasi, K., Thomas, S., Dimitriadis, D., Cui, X., Ramabhadran, B., Picheny, M., Lim, L.-L., et al. (2017). English conversational telephone speech recognition by humans and machines. ArXiv Prepr. ArXiv170302136. arXiv: 1703.02136.

Schönherr, L., Kohls, K., Zeiler, S., Holz, T., & Kolossa, D. (2018). Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. ArXiv Prepr. ArXiv180805665. arXiv: 1808.05665.

Schönherr, L., Kohls, K., Zeiler, S., Holz, T., & Kolossa, D. (2019). Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Network and Distributed System Security Symposium NDSS 2019*.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. (2018). Natural Tts synthesis by conditioning Wavenet on Mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP* (pp. 4779–4783). IEEE.

Shillingford, B., Assael, Y., Hoffman, M. W., Paine, T., Hughes, C., Prabhu, U., Liao, H., Sak, H., Rao, K., Bennett, L., et al. (2018). Large-scale visual speech recognition. ArXiv Prepr. ArXiv180705162. arXiv: 1807.05162.

Sigurdsson, G. A., Gupta, A., Schmid, C., Farhadi, A., & Alahari, K. (2018). Charades-ego: A large-scale dataset of paired third and first person videos. ArXiv Prepr. ArXiv180409626. arXiv: 1804.09626.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems* (pp. 568–576).

Speer, R., & Havasi, C. (2012). Representing general relational knowledge in ConceptNet 5. In *LREC* (pp. 3679–3686).

Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Ess-Dykema, C. V., & Meteer, M. (2000). Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics, 26*(3), 339–373.

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 7464–7473).

Switchboard. (1997). *Switchboard-1 Release 2*. https://catalog.ldc.upenn.edu/LDC97S62

Van den Oord, A., Walters, T., & Strohman, T. (2017). Wavenet launches in the Google assistant. In *Deep. Blog*.

Vincent, E., Watanabe, S., Nugraha, A. A., Barker, J., & Marxer, R. (2017). An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language, 46*, 535–557.

Vincent, J. (2016). Twitter taught Microsoft's AI Chatbot to be a racist asshole in less than a day. In *The Verge* 24. https://www.theverge.com/2016/3/24/11297050/taymicrosoft-chatbot-racist

Weizenbaum, J. (1966). ELIZA—A computer program for the study of natural language communication between man and machine. *Communications of the ACM, 9*(1), 36–45.

Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., & Stolcke, A. (2018). The Microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP* (pp. 5934–5938). IEEE.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., & Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. ArXiv Prepr. ArXiv190500546. arXiv: 1905.00546.

Yang, S., Zhang, Y., Feng, D., Yang, M., Wang, C., Xiao, J., Long, K., Shan, S., & Chen, X. (2019). LRW-1000: A naturally-distributed large-scale benchmark for lip reading in the wild. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition FG 2019* (pp. 1–8). IEEE.

Zeyer, A., Irie, K., Schlüter, R., & Ney, H. (2018). Improved training of end-to-end attention models for speech recognition. ArXiv Prepr. ArXiv180503294. arXiv: 1805.03294.

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E.-D., Jin, W., & Schuller, B. (2018). Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology, 9*(5), 1–28.

Zhou, L., Zhou, Y., Corso, J. J., Socher, R., & Xiong, C. (2018). End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern recognition* (pp. 8739–8748).

Chapter 8
Learning Optimal Policies

Abstract Reinforcement learning is an area of Machine Learning in which a software program (agent) must select an action at each time step with the goal of achieving the highest possible sum of rewards over time. An action is determined based on the current state and affects the reward, often many time steps later. Examples applications include games, robotic controls, and self-driving cars. Deep neural networks (DNNs) can be used to assign a sum of expected rewards to a state-action pair. DNNs are particularly suitable because they can approximate the underlying functions well and may be employed to select the best action for a state. Q-Networks predict an expected reward sum for each state-action pair. A stochastic policy is suitable for decision situations with random influences and determines for each state an optimal probability distribution over the possible actions. For both types of models, training procedures are derived which determine the gradient from a number of simulated model runs. In contrast to previous DNN, the training data is generated during training using a simulated or real environment. Finally, application areas of reinforcement learning are described, such as video games, robot control, and autonomous vehicles.

Reinforcement learning is an area of Machine Learning in which a software program (agent) must select an action at each time step based on the current state information. Subsequently, the agent may receive a reward as well as new information about the next state of the environment. However, an action often has an effect only after many time steps and then causes a positive or negative reward.

Reinforcement learning describes how the agent learns by interacting with the environment when it receives positive or negative feedback. Consider a boy and a campfire (Fig. 8.1). The boy approaches the fire because it is warm and shines brightly: positive reward. But if the boy touches the flame, he gets burned: negative reward. In this way, the boy learns that a fire provides light and warmth, but also that he must keep a certain minimum distance from the fire to avoid burns. This example is representative of all human learning processes that take place in interaction with the environment. Therefore, learning situations in which an agent acquires

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_8

Fig. 8.1 A boy learns through positive and negative reinforcement signals when interacting with fire. Image credits in Appendix A.3

Fig. 8.2 Application areas of reinforcement learning: board games like chess, video games like Seaquest. Image credits in Appendix A.3

an optimal behavior over time through reward and punishment signals are called reinforcement learning.

The agent must sequentially select an action at individual points in time. The goal of learning is to maximize the sum of rewards over time. Unlike supervised learning, however, the final result of an action is not immediately available. Such a situation is present in a number of tasks:

- Board games where winning or losing is not known until the end of the game (Fig. 8.2).
- Video games where a point win is available only after some actions and the final win (or defeat) is only known after a long time.
- Autonomous cars, where success (reaching the destination) or failure (car accident) is revealed only after many steering and braking maneuvers (Fig. 8.3).
- Management of a stock portfolio where the total profit is realized only after a long period and many buying and selling actions.

Fig. 8.3 Application areas of reinforcement learning: autonomous driving or managing a stock portfolio. Image credits in Appendix A.3

8.1 Some Basic Definitions

In reinforcement learning, there is always an environment described by a state s_t at time t (Fig. 8.4). This state contains all the information available to the agent when it selects the next action. In a chess game, for example, this is the positions of all pieces on the board at time t.

The agent then has the task of choosing an action a_t, such as the next move of one of its chess pieces.

After an action a_t, the environment changes resulting in a new state. The agent receives s_{t+1}, which describes the new state (possibly only partially). In addition, there is a reward, a real number r_{t+1}, e.g., points or an amount of money. In the game of chess, the next state s_{t+1} is the board position after the opponent has

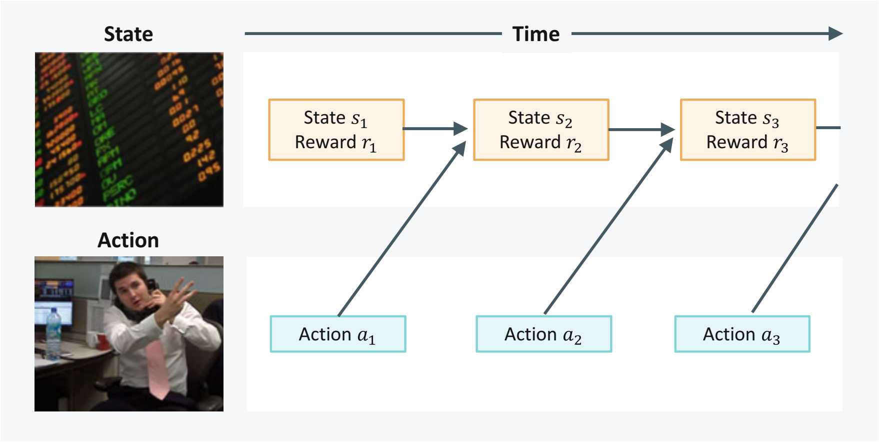


Fig. 8.4 Temporal progress of a control task for reinforcement learning. Given the current state s_t, the agent selects an action a_t. He then receives information about the next state s_{t+1} and reward r_{t+1}, selects action a_{t+1}, etc. Image credits in Appendix A.3

made a move. The reward is either “win” (+1), “loss” (-1), “draw” (0), or “game is not yet completed” (0). Thus, it can also take negative values. An observed series $(s_1, r_1, a_1), (s_2, r_2, a_2), \ldots, (s_T, r_T, a_T)$ of triples of state, reward, and action is called an episode. It can be finished by a terminal state.

The environment can have different characteristics. The simplest is a deterministic environment whose behavior is completely predictable. In this case, a state-action pair (s_t, a_t) always results in the same subsequent state s_{t+1} and the same reward r_{t+1}. This situation is easier to analyze and is present, for example, in simple robot control problems.

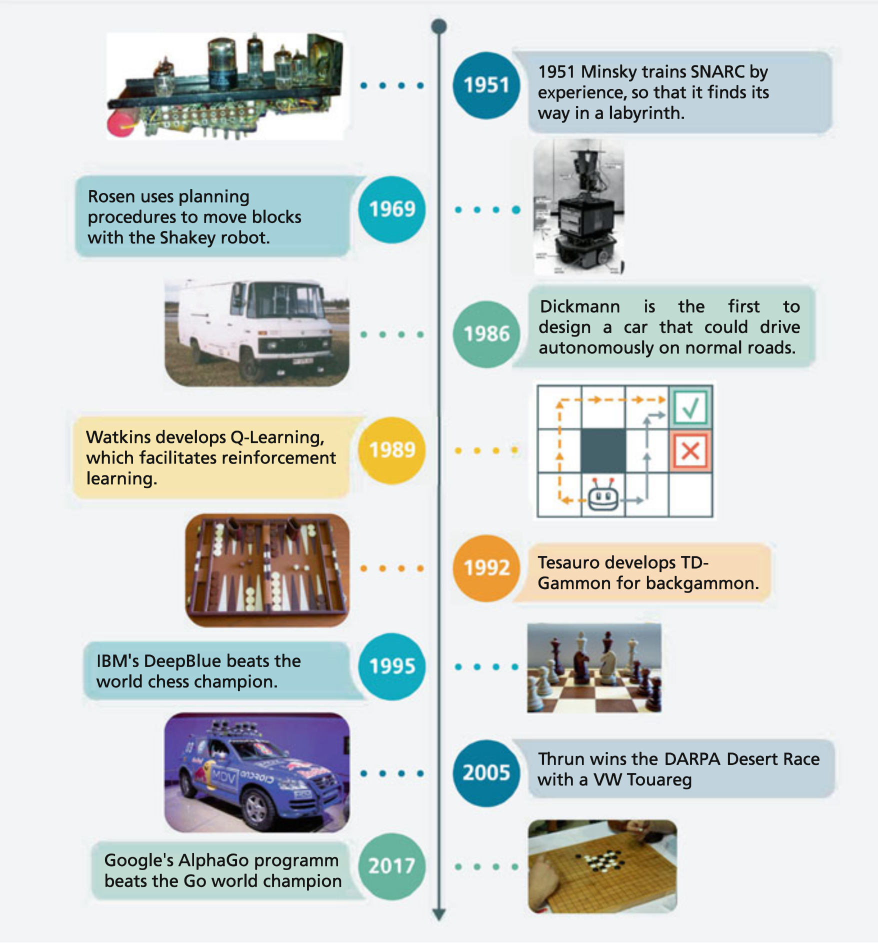
In the general case, the environment is stochastic (i.e., dependent on chance) environment. In this case, the subsequent state s_{t+1} and the reward r_{t+1} are influenced by random effects and follow a conditional distribution $p(s_{t+1}, r_{t+1}|s_t, a_t)$. This means that the same action pair (s_t, a_t) can result in different subsequent states. This also includes the case of incomplete information about the overall system or the existence of intelligent counterparts.

Reinforcement learning must solve three main challenges:

- Action-reward relationship: it has to find out how the individual actions influence the rewards in the course of an episode.
- Exploration-Exploitation: The system must decide whether to focus on game situations that currently promise a high reward, or whether to explore other unknown game situations because even higher rewards may be attainable there.
- Function approximation: In realistic applications, it is impossible to exactly represent the relationship between states, actions, and rewards because of the large number of states. Therefore, this relationship must be approximated by a model, which can usually result in additional errors.

To assess the effect of actions on future rewards, the system must forecast the impact of an action. Thus, reinforcement learning systems must be capable of “predictive learning.” Reinforcement learning recognizes patterns in the course of an episode and predicts future rewards and selects actions. Thus, the user must set a goal and the reinforcement learning must progress toward the goal by trial and error. “Reinforcement learning is an obvious idea if you know anything about psychology,” says Richard Sutton (Fig. 8.21), one of the founders of reinforcement learning (Smith 2020).

The history of reinforcement learning is outlined in Fig. 8.5. An early system for reinforcement learning was SNARC (Hogget 2019) by Marvin Minsky. It contained “neurons” made of wires, electron tubes, and relays, and was capable of following a “rat” in a maze. Shakey was the first mobile robot that could plan its own actions. It combined knowledge of robotics, image recognition, and speech processing. Dickmanns used a Mercedes van as the basis for autonomous driving. With cameras and other sensors, the system was able to drive the van fully autonomously at speeds up to 96 km/h (Dickmanns & Graefe 1988). Watkins developed Q-Learning in 1989, which represents the goodness of an action for a given state by a table or function (Watkins & Dayan 1992). Tesauro introduced the TD-Gammon algorithm in 1992, which computed the performance of actions by a neural network and was



1951 Minsky trains SNARC by experience, so that it finds its way in a labyrinth.

Rosen uses planning procedures to move blocks with the Shakey robot.

Dickmann is the first to design a car that could drive autonomously on normal roads.

Watkins develops Q-Learning, which facilitates reinforcement learning.

Tesauro develops TD-Gammon for backgammon.

IBM's DeepBlue beats the world chess champion.

Thrun wins the DARPA Desert Race with a VW Touareg

Google's AlphaGo programm beats the Go world champion

Fig. 8.5 The history of reinforcement learning. Image credits in Appendix A.3

trained by playing against itself. In the 1990s, IBM developed the Deep Blue chess computer, which was able to evaluate 126 million positions per second and beat world chess champion Kasparov. In 2005, Thrun won a 212 km race for autonomous cars organized by DARPA. In 2017, the AlphaGo program was able to beat South Korean Lee Sedol, one of the world's best professional Go players.

8.2 Deep Q-Network

8.2.1 Policy to Maximize the Sum of Rewards

The behavior of the agent is determined by a policy π that assigns an action to each state s_t, i.e., a function $\pi : s_t \rightarrow a$. For an episode $(s_1, r_1, a_1), (s_2, r_2, a_2), \ldots, (s_n, r_n, a_n)$ the policy yields a sum of rewards over time

$$G_1(\pi) = r_1 + r_2 + \cdots + r_n$$

where the rewards r_t depend on the prior states s_{t-1} and the selected action a_{t-1} and thus on the policy π. The goal of learning is to arrive at a policy π^* that yields the largest possible sum of rewards over time $G_1(\pi^*) = \max_\pi G_1(\pi)$.

8.2.2 A Small Navigation Task

Let's consider a small navigation task. A robot has to get to the destination, which is indicated by a positive reward. The environment is given by square fields. The robot can move one step to the right, left, down or up as long as it remains in the playing area (Fig. 8.6).

As a reward, the robot receives 0 points on a white field. If the robot reaches the field with the coins, navigation is over and the robot receives the reward 1. If the robot reaches the field with the skull, navigation is also over and the robot receives the reward -1.

If the optimization criterion $G_1(\pi)$, then every route of the robot with $G_1(\pi) = 1$, is optimal as long as it avoids the skull and eventually reaches the coins. So the robot can move on the left side for as long as it wants, and eventually go to the coins, and still the final reward is $G_1(\pi) = 1$.

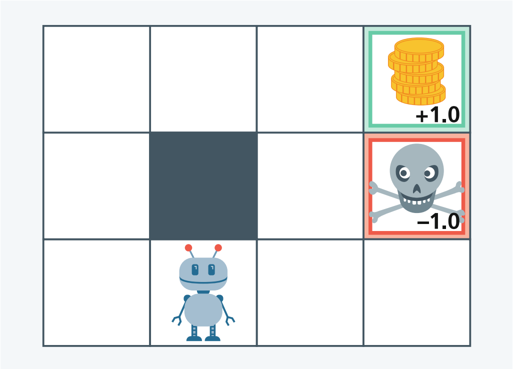


Fig. 8.6 A robot with a navigation task. The robot can move in all four directions as long at it stays on the playing area. The robot successfully solves the task provided it avoids the field with the skull and eventually reached the field with the coins

8.2.3 Discounted Future Reward Encourages Fast Solutions

In a real application, however, the robot has to reach its goal as quickly as possible. For this reason, the optimization criterion is modified in such a way that reaching the goal quickly leads to a higher gain. This can be done by giving less weight to later rewards. One chooses a number λ with $0 < \lambda < 1.0$ and defines

$$G_\lambda(\pi) = \lambda^0 r_1 + \lambda^1 r_2 + \lambda^2 r_3 + \cdots + \lambda^{n-1} r_n$$

Again the rewards r_t depend on the prior states a_{t-1} and the selected action s_{t-1} and thus on the policy π. This optimization criterion is called discounted future reward and has a discount factor λ. Common values for λ are 0.9, 0.99, or 0.999. Obviously, the rewards further in the future are downweighted, e.g., after 50 steps $0.9^{50} = 0.005$, $0.95^{50} = 0.076$, and $0.99^{50} = 0.605$. Therefore, it is profitable for the robot to try to get to the coins on the shortest path, since the discounted future reward is then larger. For simplicity, let us consider the case $\lambda = 0.9$ in Fig. 8.7. The right grey path requires 4 steps and results in $G_{0,9}(\pi_1) = 0.66$. The left dashed path requires 6 steps and yields $G_{0,9}(\pi_2) = 0.53$, which is a lower discounted future reward. Note that even for episodes of arbitrary length, the discounted reward $G\lambda(\pi)$ always has a finite limit.

8.2.4 The Q-Function Evaluates State-Action Pairs

To facilitate the general solution of such optimization tasks, the Q-Function has been defined. This definition assumes that s_t is that current state and that for all possible subsequent states $s_{t+1}, s_{t+2}, \ldots$ the optimal policy $\pi^* : s \to a$ is already available.

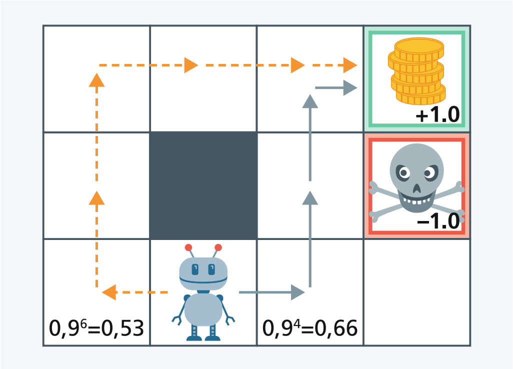


Fig. 8.7 With the optimization criterion $G_{0,9}(\pi)$, the shorter right path will give a higher discounted future reward (0.66) than the longer left path (0.53)

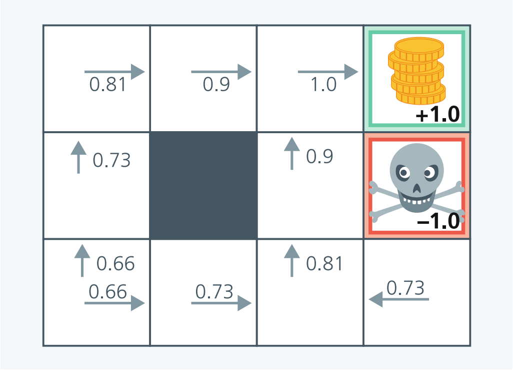


Fig. 8.8 The value of the Q-Function for each state and action (arrow) can be derived successively from the optimal reachable field (e.g. coins). The values at the arrows are the values of the Q-Function for the respective state and movement action. The black numbers are maximum sums of the rewards that can be reached from the state by the optimal policy

Thus, only for the state s_t the action a_t has to be selected.

$$Q(s_t, a_t) = \max_{\pi^*} \left(r_{t+1} + \lambda^1 * r_{t+2} + \cdots + \lambda^{n-1} * r_{t+n} \right)$$

The goal of the analysis is now to successively construct the Q-Function for a problem. In our simple navigation problem, this is easily accomplished. We can specify individual states by the corresponding row and column $s[r, c]$ in the grid (Fig. 8.8). One starts at the "target" state $s[1, 4]$ with the coins. From state $s[1, 3]$ one can reach state $s[1, 4]$ with reward 1.0 by taking a step to the right, so $Q(s_t = s[1, 3], a_t = \text{“} \rightarrow \text{”}) = r_{t+1} = 1.0$. This is the optimal achievable value and is indicated in the graph for $s[1, 3]$ by the 1.0 at the right arrow. From state $s[2, 3]$, one can reach state $s[1, 3]$ with an upward move $a_t = \text{“} \uparrow \text{”}$ with reward $r_{t+1} = 0$. Then, with the optimal action $a_{t+1} = \text{“} \rightarrow \text{”}$, one can reach state $s_{t+2} = s[1, 4]$ with reward $r_{t+2} = 1$. Thus $Q(s_t = s[2, 3], at = \text{“} \uparrow \text{”}) = \lambda^0 r_{t+1} + \lambda^1 r_{t+2} = 0.9$. Again this is the optimal achievable value and is indicated in the graph for $s[2, 3]$ by the 0.9 at the up arrow. In the same way, one can derive the Q-values for the neighboring states, e.g., $Q(s_t = s[3, 3], a_t = \text{“} \uparrow \text{”}) = \lambda^0 r_{t+1} + \lambda^1 r_{t+2} + \lambda^2 r_{t+3} = 0.81$. The Q-values are again marked on the corresponding arrows. Thus, one can finally derive the value of the Q-Function for all state-action pairs.

When the Q-Function is filled with values, it defines the optimal policy for the control task. If the system is in state s, one must only determine the action with maximum Q-value $a^* = max_a Q(s, a)$. By definition of Q, a^* is just the optimal action to maximize the discounted future reward. Hence, $\pi^*(s) = max_a Q(s, a)$ is the optimal policy.

8.2.5 The Bellman Equation Relates Q-Values to Each Other

Evaluating the Q-Function in this way is quite a complex task and not practicable for realistic problems. For example, it has been estimated that chess has about 10^{46}

different states (board positions). This is an incredibly large number (about the mass of our galaxy in grams). Therefore, in practice, these states cannot be listed.

But there is a way to construct the Q-Function successively. Let us assume that we know values of r_{t+1} and s_{t+1}. Then

$$Q(s_t, a_t) = r_{t+1} + \lambda * \max_{\pi} \left(r_{t+2} + \lambda^1 r_{t+3} + \cdots + \lambda^{n-2} r_{t+n} \right)$$

because one can factor out λ from the sum. If we substitute the definition of the Q-Function, we get the Bellman equation

$$Q(s_t, a_t) = r_{t+1} + \lambda * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

Hence, there is an equation relating $Q(s_t, a_t)$ and the Q-values of the subsequent state-action pairs s_{t+1}, a_{t+1}. It is important to note that the optimal action a_{t+1} must be selected here. This equation was already established by Bellman (1957) in the 1950s in the context of dynamic programming. It decomposes the extensive optimization problem into a large number of simple optimization problems.

8.2.6 Approximation of the Q-Function by a Deep Neural Network

We have seen previously that one can fill the Q-Function successively with values. Instead of separately deriving a value for each state-action pair we may exploit similarities between the Q-values of different state-action pairs. This can be done by approximating the Q-Function with by a deep neural network (DNN) $\hat{Q}$, which, after all, has good approximation properties:

$$\boxed{Q(s_t, a_t) \approx \hat{Q}(s_t, a_t; w)}$$

Parameter

The parameter vector w determines the values of this deep Q-Network and is optimized in such a way that $\hat{Q}(s_t, a_t; w)$ calculates similar values as the real Q-function $Q(s_t, a_t)$. The advantage of a representation by a neural network is that states with similar Q-values are automatically grouped together. The network can therefore generalize its value calculations to similar state-action pairs. In practice, it has been found useful to predict not only the Q-value for one action, but at the same time Q-values for all possible actions for a given state s_t (Fig. 8.9).

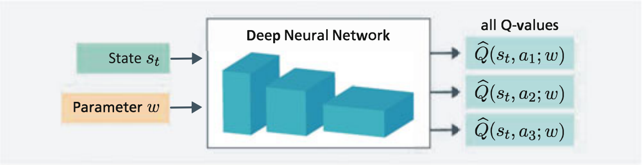

State s_t
Parameter w
Deep Neural Network
all Q-values
$\widehat{Q}(s_t, a_1; w)$
$\widehat{Q}(s_t, a_2; w)$
$\widehat{Q}(s_t, a_3; w)$

Fig. 8.9 Simultaneous computation of Q-values for all actions by a deep neural network with three layers

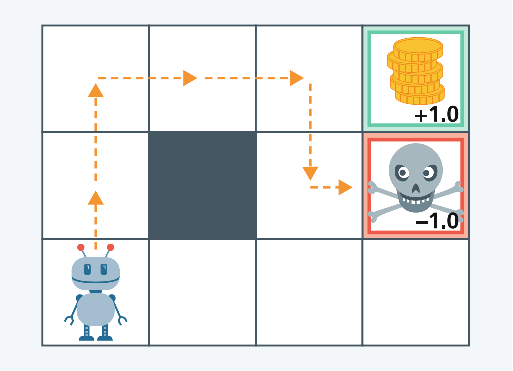


Fig. 8.10 Create an episode from a random starting state s_1. The next action is selected using the current Q-Function $\hat{Q}(s, a; w)$. The episode ends after a certain number of steps or when a final state is reached

8.2.7 *Q-Learning: Training a Deep Q-Network*

Creating an Episode with the Deep Q-Network

At the beginning of the training, the parameter vector w is filled with arbitrary random values from $[-1.0, 1.0]$. The approach now pretends that the approximated Q-Function values are already optimal. An episode is generated with the help of a sequence of model forecast. For this purpose, an arbitrary starting state s_1 is randomly selected and an action a_1 is determined via $a_1 = \max_a \hat{Q}(s_1, a; w)$. Then (s_1, a_1) is transmitted to the environment, which provides the next state and reward: $(s_2, r_2) = env(s_1, a_1)$. The environment $env()$ can be the real environment or a computer simulation. Then the next action $a_2 = \max_a \hat{Q}(s_2, a; w)$ is selected and so on. In this way, one can generate a complete episode: $(s_1, r_1, a_1), (s_2, r_2, a_2), \ldots, (s_n, r_n, a_n)$ (see Fig. 8.10). The actions in this initial phase are of course almost random and often lead to a low sum of discounted rewards after a few steps.

Optimization with the Generated Episode

Let us consider a group $(s_t, a_t, s_{t+1}, r_{t+1})$ of an episode $(s_1, r_1, a_1), \ldots, (s_n, r_n, a_n)$. For the approximate Q-Function, the Bellman equation must hold for the given four quantities: $\hat{Q}(s_t, a_t; w) = r_{t+1} + \lambda * \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; w)$. To reduce the difference the parameter w must be adjusted so that this equation is valid, i.e., the right and left sides of the equation have the same value. We can get closer to this goal by reducing the squared distance between the two sides of the equation. This results in the loss function

$$L(w) = \sum_{t=1}^{n} \left[\hat{Q}(s_t, a_t; w) - \underbrace{\left(r_{t+1} + \gamma * \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; w) \right)}_{\text{target value for } \hat{Q}(s_t, a_t; w)} \right]^{\overset{\text{square distance}}{2}}$$

The squared distance between $\hat{Q}(s_t, a_t; w)$ and its target value is thus reduced towards zero for all observed states of the episode. If $L(w) = 0$, the Bellman equation holds for all observed groups.

Now the gradient of the loss function $\partial L(w)/\partial w$ can be calculated and the parameter vector w can be modified with stochastic gradient descent. Then, a new episode is generated with the modified $\hat{Q}(s_t, a_t; w)$, which is again fitted to the Bellman equation during training. In this way, the approximation $\hat{Q}(s_t, a_t; w)$ successively approaches the actual Q-Function $Q(s_t, a_t)$. Gradually, this adjusts the Q-Function so that the Bellman equation holds everywhere. The optimization stops when there are no more significant changes in the loss values (Fig. 8.11).

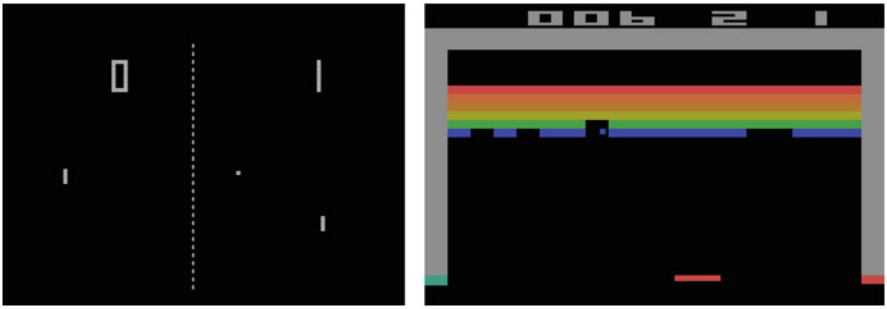


Fig. 8.11 The Q-Network was applied to these video games. The Atari 2600 video game Pong (left) was one of the first video games ever and was published in 1972. Each of the two players controls a paddle. The aim of the game, similar to table tennis, is to hit a ball back. A lost ball scores a minus point. In the 1976 video game Breakout (right), the player's goal is to use a "bat" to direct the ball so that it hits bricks, which then disappear. Losing the ball is scored negatively. Image credits in Appendix A.3

Q-Learning was developed by Watkins (1989). Watkins and Dayan (1992) show for a Q-Function $Q(s_t, a_t)$ that Q-Learning produces the optimal Q-function if all states and actions are repeatedly reached. Jin et al. (2018) prove that Q-Learning without a model is the most efficient method to compute a policy. Yang et al. (2019) study the convergence of Q-Learning for a model $\hat{Q}(s_t, a_t; w)$ with ReLU activations. They show for some of "mild" conditions that $\hat{Q}(s_t, a_t; w)$ converges towards a Q-Function that approximates the optimal Q-Function as well as is possible for this class of models.

Q-Learning has a strong analogy to human behavior. A large part of what we do in our daily work is to constantly refine our mental models of the world, and then use these mental models to solve problems (Smith 2020). A deep Q-Network is such mental model of a section of the world. The optimization of the model is thus based on human behavior.

Practical Tricks: Selection of Training Examples and Loss Function Calculation

In practice, it has been found that it is useful to store the data of several episodes and randomly select the groups $(s_t, a_t, s_{t+1}, r_{t+1})$ from this episode memory (experience replay). In this way, the data are no longer so strongly correlated with each other.

We have assumed so far that the environment is deterministic. However, it has been shown that Q-Learning also works if the environment is stochastic. In this case, an average value is simply calculated over the different possible subsequent states.

Another trick is to use two deep neural networks: one Q-Network is employed to select the (currently) best action, another Q-Network (target network) is utilized to determine the target value in the loss function. The second Q-Network is updated with a certain delay. This reduces any systematic bias in the results, which may arise when the same Q-Network is used to select the action and calculate the Q-value. Yang et al. (2019) provide a theoretical justification for this approach.

Exploration

If the number of states is very large, the learning procedure may obtain a suboptimal policy. In the further course of optimization, only states and actions that can be reached from this suboptimal policy are evaluated. Thus, the learning procedure may not acquire new, better policies. For this reason, it is necessary to keep trying new, supposedly inferior actions. One ensures this exploration of the state set by selecting random actions with a certain probability when generating an episode. As the optimization proceeds, this probability is reduced in order to deviate less from the optimal policy. One can also theoretically prove the necessity of exploration.

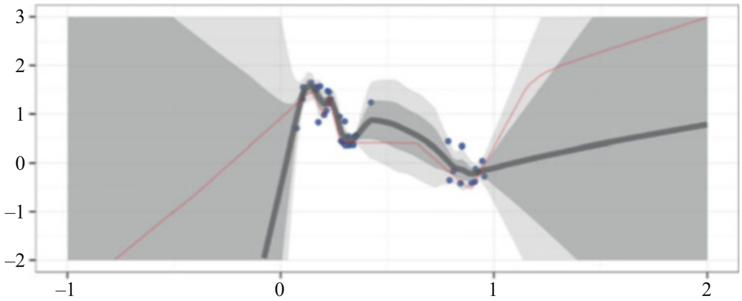


Fig. 8.12 Calculation of uncertainty in Q-values by an ensemble of bootstrap models (Osband et al. 2016)

Yang et al. (2019) have verified the convergence of Q-Learning towards the correct solution if exploration is done to a certain extent.

Random exploration does not take into account what is already known about a state-action pair s_t, a. Alternatively, one can determine the uncertainty about the Q-value $\hat{Q}(s_t, a; w)$, and select an action with relatively low Q-mean and high uncertainty, which may still have a high Q-value. This requires estimating the accuracy of the predictions of $\hat{Q}(s_t, a; w)$, using any of the options discussed in Sect. 5.11. Osband et al. (2016) generate a bootstrap ensemble of approximating Q-Networks and use it to compute the uncertainty of Q-values for different actions (Fig. 8.12). For exploration, they select actions a for which the Q-values may be high. In experiments, they show that this approach can often train an optimal policy with much fewer queries to the environment.

8.3 Application of Q-Learning to Atari Video Games

8.3.1 Definition of the Game State in Atari Games

Atari 2600 was a popular video game console that sold about 30 million units from 1977 to 1992. The games were reimplemented by OpenAI and made available as a toolkit gym for the development of reinforcement learning algorithms (Brockman et al. 2016). Of the 49 Atari games, four are depicted: Pong and Breakout (Fig. 8.11) and Roadrunner and Seaquest (Fig. 8.13).

To apply Q-Learning to the Atari games, one must first define the game state s_t for a game. Mnih et al. (2015) suggest not to define separate explicit state variables for each game, but to use four consecutive video frames of 210 × 160 pixels in 128 colors as state description s_t for all games. Movements in the game can be detected by comparing the successive frames. To derive a control policy, the network has the additional task to extract the state features relevant to the particular game from the video frames.

Fig. 8.13 In the Atari 2600 video game Road Runner (left), the Roadrunner character runs to the left and is chased by the coyote. In the process, the character must avoid obstacles and pick up seeds. In the Seaquest video game (right), the player controls a submarine and must shoot down "enemies" (submarines and sharks) and rescue divers. In addition, he must repeatedly refuel with oxygen. Image credits in Appendix A.3

8.3.2 Architecture of the Atari Q-Network

To extract the state s_t from the video frames, Mnih et al. (2015) used a network with three convolutional layers (Fig. 8.14). Then, two fully connected layers (Sect. 4.3) were employed to predict a vector of Q-values $Q(s_t, a)$ for each action. The optimization uses the loss function $L(w)$ of Sect. 8.2.7 along with a version of stochastic gradient descent that automatically adjusts the step size.

The authors generated episodes by evaluating the current Q-Function $\hat{Q}(s_t, a_t; w)$ and used a memory for one million groups $(s_t, a_t, s_{t+1}, r_{t+1})$ for experience replay. From the memory, they randomly selected a minibatch of 32 groups for optimization. The memory was successively filled with more generated episodes (Sect. 8.2.7), while old groups were overwritten. Reward values were normalized with a maximum absolute value of 1.0 in order to limit gradients.

8.3.3 Training of Atari Q-Networks

Mnih et al. (2015) trained different models for each of the 49 Atari games for which baseline results were available. The same network architecture was used each time (except for the number of outputs for each action, which ranged from 4 to 18). At the beginning of training, random actions were generated with probability 1.0 to ensure exploration. This probability was slowly reduced to 0.1. Training consisted of 50 million frames per game. This is equivalent to about 38 days of human game play. As an example Fig. 8.15 shows the learning curve for Seaquest with a randomly but gradually increasing score. Overall, the modifications outlined

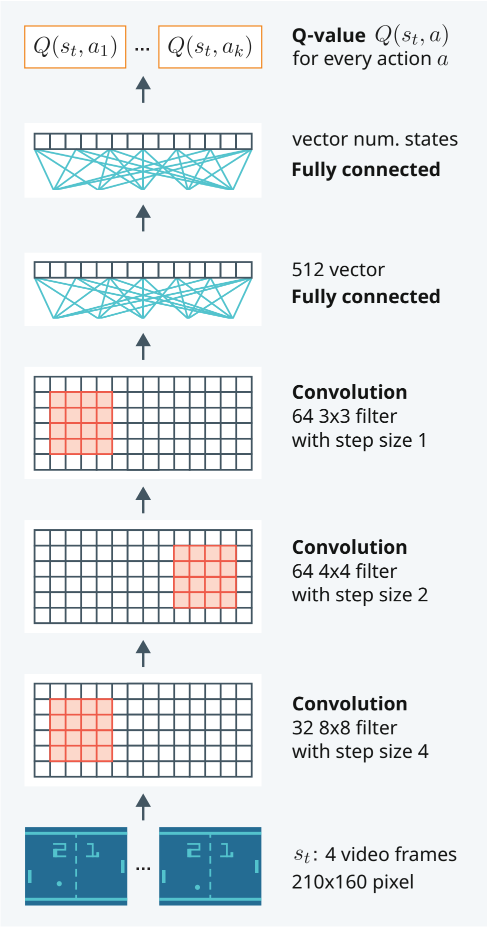


Fig. 8.14 Deep Q network for Atari2600 video games. ReLU-activations were used as nonlinearities (Mnih et al. 2015)

previously (experience replay, exploration, target network) have resulted in deep Q-Networks achieving very stable results.

8.3.4 *Evaluation of Atari Q-Networks*

Mnih et al. (2015) compared the performance of their trained Q-Networks with prior models. In elaborate experiments they determined the performance of professional human game testers under controlled conditions.

For nearly all games deep Q-Learning achieves better results than previous models (Fig. 8.16). In contrast to the previous approaches, no special additional knowledge about the individual video games was used. Moreover, the results for half of the games were on the same level as the human game testers (at least 75% of the human score). This shows that with the same model architecture deep Q-

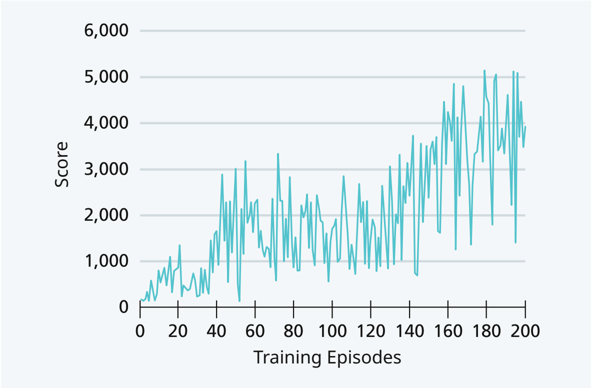


Fig. 8.15 Example training run with the score achieved for Seaquest with growing number of training episodes

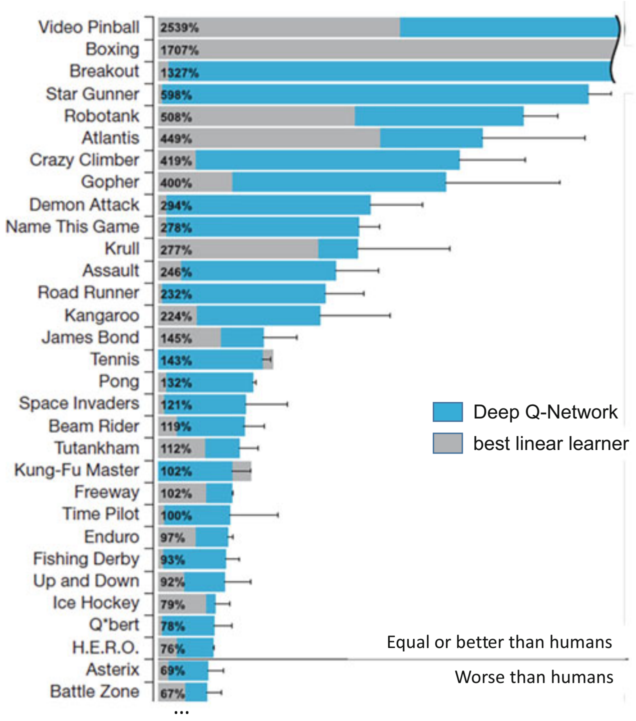


Fig. 8.16 Comparison of the performance of deep Q-Networks (blue) with previous models (gray). Due to space limitations, the lower part of the graph has been shortened. The scores are the percentage of human tester performance achieved by a deep Q-Network (Mnih et al. 2015). Image credits in Appendix A.3

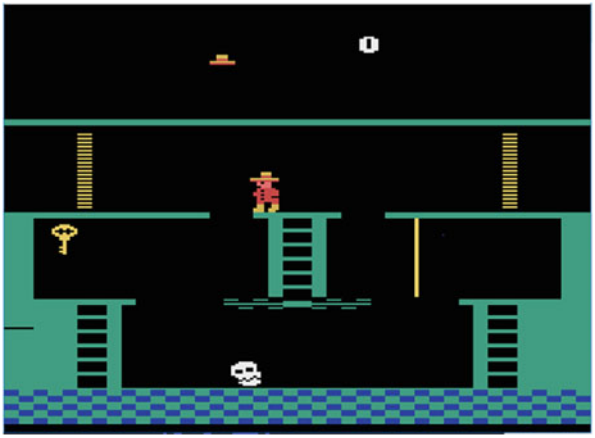


Fig. 8.17 The Atari 2600 video game Montezuma's Revenge. The player must take the key and open the door. Image credits in Appendix A.3

Networks can solve a wide range of different problems without the use of prior knowledge.

It is important to keep in mind that the games have very different structures: from the shooting game "River Raid" with on-screen scrolls, to boxing games ("Boxing"), to three-dimensional car racing games like "Enduro". In individual games, the algorithm discovered a policy: in "Breakout," for example, it "drills" a hole at the side so that the ball flies through and destroys a large number of bricks.

The system has more difficulty with games that require extended planning, e.g., "Montezuma's Revenge" (Fig. 8.17). The agent must jump down, climb up, take the key, and open the door. A human immediately understands that it is useful to take the key. The agent makes millions of random movements and in the vast majority of cases he falls to death or is killed by the monsters. So it is very difficult to "stumble" into a reward situation. But this also is true for all previous automatic approaches. Meanwhile, the Q-Learning was improved by extending the planning horizon and making the exploration strategies adaptive. The agent Atari57 (Badia et al. 2020) could now beat average human players with the same hyperparameters in all 57 Atari games.

8.4 Policy Gradients for Learning Stochastic Policies

8.4.1 Need for Policies with Random Elements

Deep Q-Networks generate a deterministic policy. As we have seen, it can also be applied in stochastic environments. However, problems arise when there is an intelligent opponent, as for example is the case of chess.

Consider the simple scissors-rock-paper game (Fig. 8.18). Each opponent independently shows one of the alternatives scissors, stone or paper. The following

Fig. 8.18 The fourth championship of the scissors-and-stone-paper game in the United Kingdom in 2010. Image credits in Appendix A.3

holds: rock wins against scissors, scissors wins against paper, and paper wins against rock. If an agent has a deterministic policy—e.g., always plays rock—an intelligent opponent can always beat it. Therefore, stochastic policies are required for this game, selecting one of the three actions with some probability. The optimal policy chooses all three actions with a probability of 1/3 (Wikiludia 2018).

One way to compute optimal stochastic policies is through the policy gradient method, which is presented in the next section.

8.4.2 Direct Optimization of a Policy by Policy Gradients

A stochastic policy for a given state s_t is a conditional probability distribution $p(a|s_t)$ over the actions a with s_t as condition. This probability distribution selects the next action a_t for a given state s_t with the corresponding probabilities. This probability distribution is again approximated by a neural network $\hat{\pi}(a|s_t; w)$ with free parameter vector w. Here we also assume that the environment is stochastic, i.e., the next reward and the next state arise according to a conditional probability distribution $q(r_{t+1}, s_{t+1}|s_t, a_t)$. As shown in Fig. 8.19 an episode $\tau^{(1)}$ can be generated by alternating generating actions, states and rewards with these probability distributions.

Since both the policy and the environment are stochastic, a repetition produces a different episode $\tau^{(2)}$. Thus, one obtains a distribution $p(\tau|w)$ over the episodes, which of course depends on the policy parameter w.

For a policy $\hat{\pi}$ we can determine the average reward.

- The policy $\hat{\pi}(a|s_t; w)$ generates the episodes $\tau^{(1)}, \ldots, \tau^{(m)}$ under random influence.
- Each episode $\tau^{(j)}$ has the sum $R(\tau^{(j)})$ of rewards discounted by λ.
- Thus the policy given by w has the value $V(w) \approx \frac{1}{m} \sum_{j=1}^{m} R(\tau^{(j)})$, i.e., just the average discounted sum of rewards.

1. Randomly generate an initial state s_1.
2. Given state s_t, the agent randomly generates an action a_t according to the probability distribution $\hat{\pi}(a|s_t; w)$.
3. The environment randomly produces a reward r_{t+1} and a new state s_{t+1} according to the conditional distribution $q(r_{t+1}, s_{t+1}|s_t, a_t)$.
4. Continue with step 2. The procedure stops if a final state is reached or after a maximum number of steps.

In this way, an episode $\tau = (s_1, a_1, s_2, r_2, a_2, \ldots, s_n, r_n)$ is generated. The sum of rewards discounted by λ is $R(\tau) = \lambda r_2 + \lambda^2 r_3 + \cdots + \lambda^{n-1} r_n$

Fig. 8.19 Generating an episode τ given a stochastic environment and a stochastic policy $\hat{\pi}$

$V(w)$ is the quantity that must be optimized so that the policy yields the highest rewards on average. A derivative can be calculated for this function (Li 2018):

$$\boxed{\frac{\partial V(w)}{\partial w} = \underbrace{\frac{1}{m}\sum_{j=1}^{m}}_{\text{Average over generated episodes}} R(\tau^{(j)}) \sum_{t=0}^{T-1} \underbrace{\frac{\partial}{\partial w} \log \hat{\pi}(a_t^{(j)}|s_t^{(j)}; w)}_{\text{Derivative of logarithms of estimated action probabilities}}}$$

Here $\hat{\pi}(a|s_t; w)$ is the neural network predicting the action probabilities. It is also called deep policy network or policy gradient model and computes the probability of each action for state s_t. For training, one has to calculate the derivatives of the log predicted probabilities $\log \hat{\pi}(a|s_t; w)$ with respect to the components of the parameter vector. This can be done very easily with current neural network toolboxes (Sect. 4.5.2). Sutton et al. (2000) proved that optimizing the policy with this derivative indeed converges to a local minimum (Fig. 8.20).

In the application, it has been found that one needs only $m = 1$ episodes to perform an optimization step. Training starts with randomly initialized parameters w. In the training loop, one alternately generates an episode τ with the current policy $\hat{\pi}(a|s_t; w)$ and computes the gradient according to the previous equation. Then stochastic gradient descent with learning rate γ is used

$$w = w + \gamma * \frac{\partial V(w)}{\partial w}$$

to change the parameter. This method is called policy gradient. Richard Sutton (Fig. 8.21) was one of the developers.

Figure 8.20 shows a very simplified representation of different episodes generated by a policy $\hat{\pi}(a|s_t; w)$. Since it is a stochastic policy and a stochastic environment, the same policy defined by w will yield different trajectories. Areas with high rewards are marked in red. The policy gradient procedure reduces the

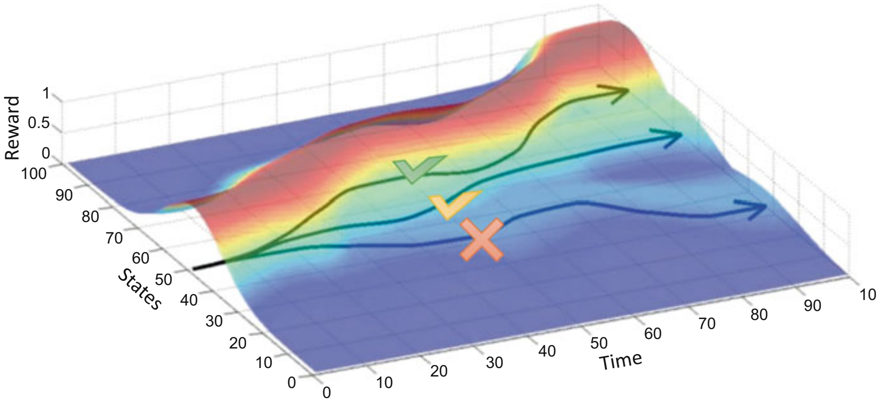


Fig. 8.20 When predicting the network for a given policy $\hat{\pi}(a|s_t; w)$ episodes are generated that differ randomly. The policy gradient method computes derivatives with respect to the parameter vector w and changes w such that the average loss gets smaller (Levine 2018). Image credits in Appendix A.3

Richard Sutton studied psychology and computer science and received his PhD in reinforcement learning: "An important advantage of reinforcement learning is that you can learn during normal operations." Since 1984, he has worked at various universities and became a researcher at Amherst University in 1995. He developed Temporal Difference Learning and the policy gradient method. From 2003, he moved to the University of Alberta, Canada. In 2017, he cooperated with Google DeepMind. For him, there are indications that reinforcement learning takes place in the brain: "An important role of the neurotransmitter dopamine is to transport the temporal-difference error, the reward-prediction error."

Fig. 8.21 Richard Sutton, born in Ohio, developed important methods of reinforcement learning. Image credits in Appendix A.3

probability of low reward trajectories and increases the probability of episodes with a high sum of rewards.

The policy gradient has several advantages over Q-Learning. First, the approach trains stochastic policies, which is necessary with an intelligent adversary. In addition, stochastic policies also have "built-in" exploration because they do not just select a single best action. Furthermore, it is also possible to track continuous actions where the action is a real number, such as a velocity. In Q-Learning, the actions must be discrete.

8.4.3 Extensions of the Policy Gradient: Actor-Critic and Proximal Policy Optimization

One problem with the policy gradient method is the high variance due to the random generation of the episodes. This variance can be reduced by replacing the observed rewards with a model, namely the Q-Model. To do this, a separate model $\hat{Q}(s, s; u)$ with parameter u is determined and the reward in the previous formula is replaced by $\hat{Q}$. Such a model is called Actor-Critic and has less variance (Yoon 2019).

The optimization landscape can be very jagged and too large an optimization step leads to a severe performance drop (Fig. 8.22). One way to improve this is to compute a trust region in which an increase is certain. Then the next step within the trust region is determined (blue dot). The proximal policy optimization (PPO) procedure of Schulman et al. (2017) follows this scheme. It constrains the differences between the distributions of action probabilities $\hat{\pi}(a|s_t; w)$ caused by an optimization step. This makes the method very robust and achieves large improvements.

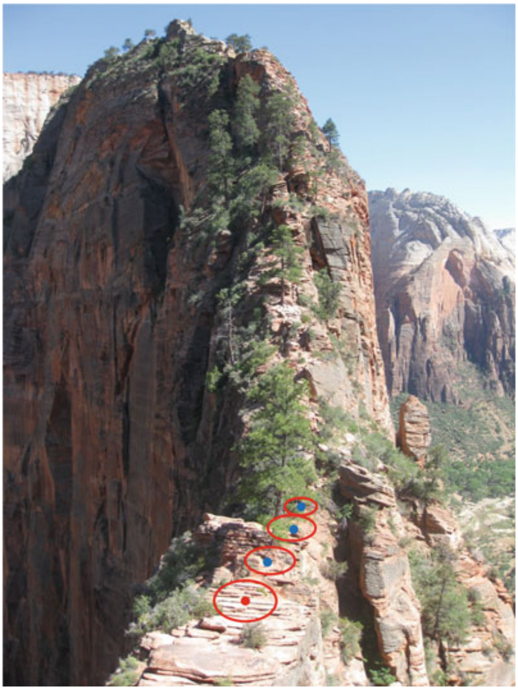


Fig. 8.22 To avoid a "crash" one can first determine a "confidence region" where an ascent is safe (red circles). Then, for the current position (blue dot) the next step is selected inside the trust region. Image credits in Appendix A.3

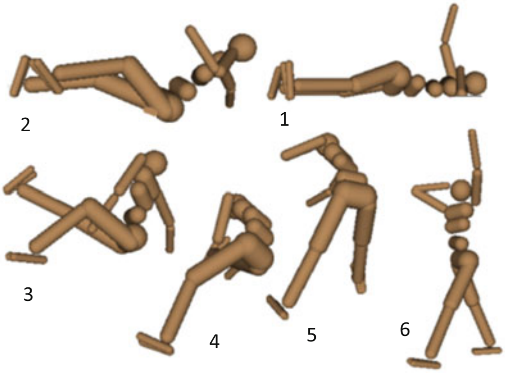


Fig. 8.23 A robot learns to stand up controlled by policy gradients. Positions 1 to 6 are adopted in the process of standing up. Image credits in Appendix A.3

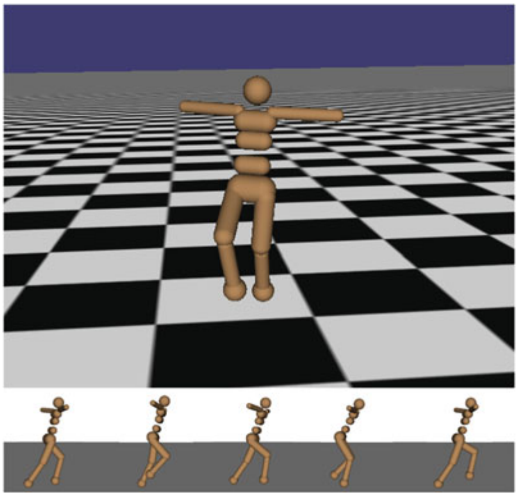


Fig. 8.24 A robot learns to walk using policy gradients. Image credits in Appendix A.3

8.4.4 Application to Robotics and the Go Board Game

Extended policy gradients have been very successful in reinforcement learning problems. Schulman et al. (2015) simulated humanoid robots and trained them to stand up (Fig. 8.23) and walk (Fig. 8.24). The deep neural network had three hidden vectors of length 100, 50, and 25, and fully connected layers with tanh activation. The last layer had a linear activation. The robot has 10 degrees of freedom. The learned policy after 1000 iterations is a fast, smooth and completely stable gait.

The method was also successfully applied to learn policies for the board game Go (Silver et al. 2016). It was the core of the AlphaGo agent which was able to beat the Go world champion Ke Jie (Byford 2017).

Igl et al. (2019) consider the problem that a model of reinforcement learning cannot be simply applied to a slightly different environment. For usual DNNs, one achieves good generalization with regularization techniques such as dropout

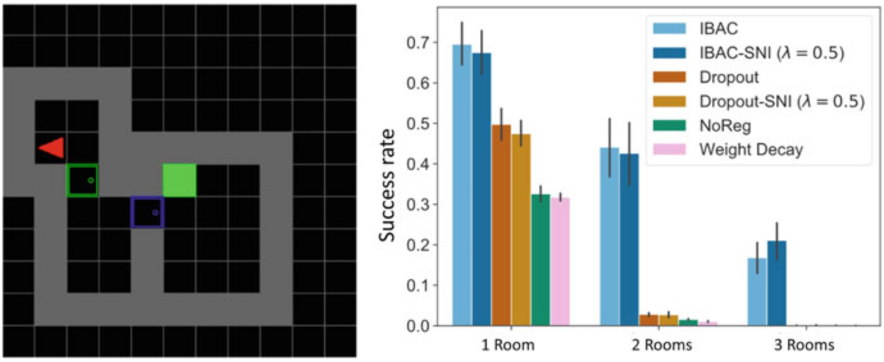


Fig. 8.25 On the left side the task is shown: find the green square in another room. The agent (red triangle) has the task to find the target (green square). The gray squares are walls and the colored boxes are doors (Igl et al. 2019)

or batch normalization (Sect. 4.6.7). The authors suggest a special regularization approach for reinforcement learning: the adaptive error injection. It improves generalization without affecting gradient computation. Furthermore, they use an information bottleneck (Sect. 4.7.6) and show that it is also a particularly good regularization technique for reinforcement learning.

As an application, they explore the task of having an agent traverse a series of rooms to find a target (the green square) (Fig. 8.25 left). The agent has discrete actions at his disposal: Turning right or left, moving forward, opening or closing the door. This environment is randomly generated each time. Thus, the agent must abstract from the single environment and generalize to a whole bundle of tasks.

The right side of Fig. 8.25 shows the frequency with which a trained agent successfully finds the target in new environments. The usual approaches (weight decay, dropout, no regularization) are not able to find the target in three rooms. In contrast, the new regularization technique IBAC-SNI (Information Bottleneck Actor Critic- SelectiveNoise Injection) reach the target in 21% of the cases. The difficulty of this seemingly simple task arises from its generalization requirements: Since the environment is randomly generated in each episode, each state is observed very rarely, especially for environments with multiple rooms where generalization is required to enable learning.

8.4.5 Application to the Dota2 Real-Time Strategy Game

The first applications of reinforcement learning were board games such as chess, where there is a very simple environment (board) and few action options (moves). If the richness of detail and variability of the real world shall be captured, video games are currently the best approximation. They require the interaction of multiple players, have long planning horizons in which the goal must be achieved, and a world of which only a small part is observable.

Dota2 is a real-time strategy game and is played between two teams of five players each. It is one of the most popular video games. Besides, it runs on Linux and has a program interface, which makes it easier to control with a DNN.

Open AI uses self-optimizing algorithms based on neural networks for Dota2. An agent, an independent DNN, is trained for each of the five players. Each agent is an LSTM (Sect. 6.4) with a hidden vector of length 4096. It has access to about 20,000 numbers representing the part of the game world that is visible to it. The agent issues actions consisting of a list of 8 numbers. The agents act by themselves based on their stochastic policy. There is no special communication facility between them, but as a result of the training they act as a team.

During training, two teams of five agents each play against each other. Each agent has its own skills and develops its own strategy. These teams generate daily game sequences with a real-life duration of 180 years and use 128,000 processor cores and 256 GPUs. After each time interval, there are rewards, such as a positive number if a character gains an experience and a negative number if, for example, an agent is wounded. Proximal Policy Optimization (Sect. 8.4.3) is used to optimize the parameters of the network, i.e., to obtain higher rewards on average. The agents have played so many games during training that it would take a human player about 45,000 years.

In April 2019, at an event in San Francisco, the OpenAI-Five software (OpenAI-Five 2018) competed against the reigning world champion, the e-sports team OG (Fig. 8.26). There were some limitations: Only 17 of the available agent types were allowed because OpenAI-Five had only implemented so many. The other limitations gave the human team an advantage. For example, unlike humans, OpenAI-Five's agents could not collude. Also, their action sequence was slowed down to the human speed. OpenAI-Five won 2-0 games against OG (OpenAI 2019). In total, this was

Fig. 8.26 Scene from the Dota2 game between the OpenAI team of bots and the human OG team. Image credits In Appendix A.3

Sebastian Thrun studied computer science at the University of Bonn and did his PhD on robotics: “I got into robotics because it was the best way for me to explore intelligence.” In 2003, he moved to Stanford University. In 2005, his team won the DARPA Grand Challenge, a race for robotic cars. In 2011, he to lead the development of self-driving cars at Google.
In 2011, he offered his Introduction to Artificial Intelligence course on the Internet with great success. He then founded the online university Udacity with colleagues: “Education should learn from the positive side of gaming – reward, achievement, fun.” He produced important scientific contributions to probabilistic robotics, transfer learning and to self-driving cars.

Fig. 8.27 Sebastian Thrun * 14.05.1967 in Solingen, Germany, is a pioneer for transfer learning and self-driving cars. Image credits in Appendix A.3

the first time an AI system defeated the reigning world champions in a complicated and lengthy strategy game (Fig. 8.27).

8.5 Self-Driving Cars

A self-driving car must observe its surroundings with cameras, radar or other sensors. The control system—the agent—must understand the vehicle’s environment, recognize the vehicle’s position in the environment and plan the next steps required to reach the destination. In order to illustrate the processing of sensor information, it is first necessary to clarify the characteristics of the sensors. Then, a use of DNN for policy optimization can be discussed.

8.5.1 Sensors of Self-Driving Cars

Self-driving cars have different sensors to monitor the environment and traffic. The most important are video, radar and lidar (light detection and ranging).

Today, Video cameras can record videos of the environment with a high image resolution and frame rate, and are inexpensive and widely used sensors. However, they have problems at night and in bad weather, when lighting conditions are insufficient. DNNs can process the camera images, detect individual objects, and annotate the corresponding image pixels (Fig. 8.28). These techniques were introduced in Sect. 5.8.2.

Radar sensors emit pulses of radio waves and can measure the distance of targeted objects and their speed evaluating the reflected signals. They work well in bad weather or at night, but have a poor resolution. Figure 8.29 shows the information collected by a radar on a road.

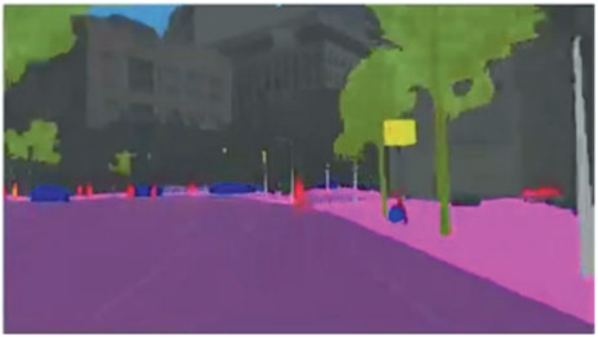


Fig. 8.28 Annotated pixels of a camera image with detected roadway, shoulder, trees, signs, etc. (Fridman 2019). Image credits in Appendix A.3

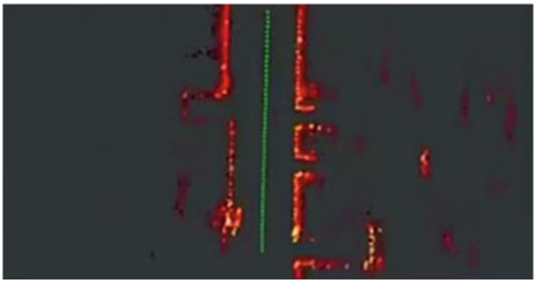


Fig. 8.29 Radar image of a road reconstructed from the radar sensor of a car (Fridman 2019). Image credits in Appendix A.3

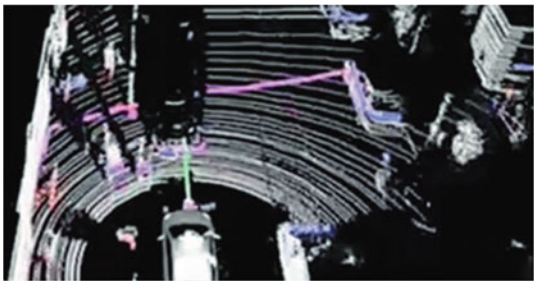


Fig. 8.30 Resolution and information density for Lidar at the same perspective as in Fig. 8.29 (Fridman 2019). Image credits in Appendix A.3

Lidar is a method of measuring the distance of objects. Infrared laser light pulses are sent into the environment. The return times and wavelengths are then used to determine the distance of the objects. Moving mirrors allow millions of pulses to be sent out per second to different directions, providing 3D distance profiles of the environment. Figure 8.30 shows the information obtained by Lidar in the vicinity of a vehicle in the same situation as in Fig. 8.29. The resolution

Feature	Lidar	Ultra-sound	Radar	Video
Range	4	1	4	5
Resolution	4	2	3	5
at night	5	5	5	1
in the light	5	5	5	4
Snow/rain	3	5	5	2
Color, contrast	1	1	1	5
Speed measurement	4	1	5	1
Sensor size	1	5	5	5
Sensor costs	1	5	5	5

Fig. 8.31 Properties of different sensors for autonomous driving on a scale from 1 (low, bad) to 5 (very high/good) (Fridman 2019)

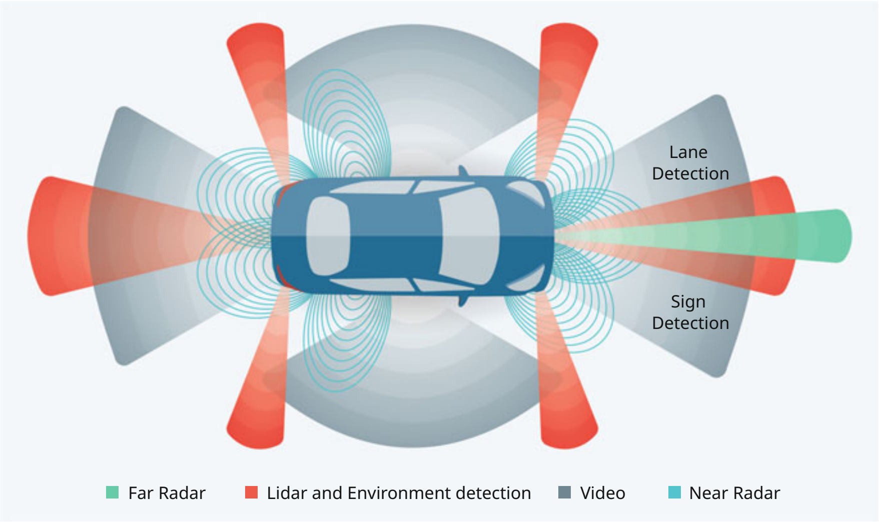


Fig. 8.32 Arrangement of sensors of a self-driving car

is much better than for Radar, but not quite as good as for video cameras. To reconstruct 3D information from camera images requires sophisticated Machine Learning techniques, whereas with Lidar it is measured precisely. However, Lidar sensors used to be very expensive, but prices have fallen sharply recently.

Figure 8.31 shows the advantages and disadvantages of the individual sensors. Ultrasonic sensors, which have a relatively poor resolution, are also evaluated here. Every self-driving car is equipped with a large number of different sensors pointing in different directions. Figure 8.32 presents the most important sensors and their detection ranges.

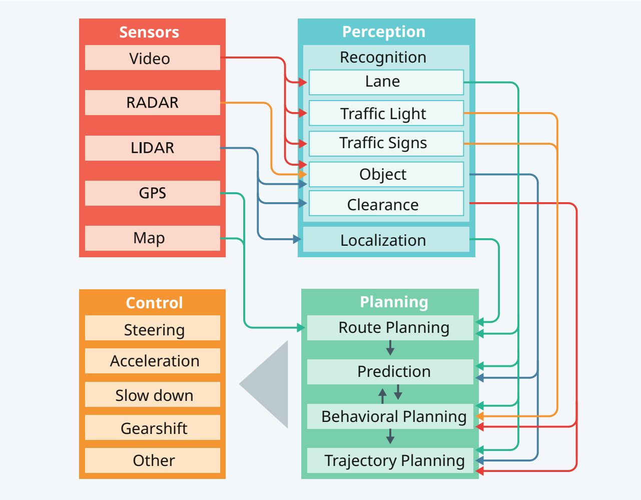


Fig. 8.33 Components of an agent for self-driving cars (Giacaglia 2019)

8.5.2 *Functionality of an Agent for Autonomous Driving*

The agent of a self-driving car has to provide a number of functions, which are shown in Fig. 8.33. With the help of the sensors, different objects have to be detected: the lane, traffic lights and traffic signs, objects like pedestrians, cyclists, bollards, other cars as well as the usable free space.

The developers of self-driving cars keep the details of the controls secret so as not to lose their competitive advantage. Therefore, only relatively general statements can be made about the algorithms used. Artificial Intelligence is largely used in perception algorithms to understand "raw" sensor data. Tasks such as combining information from different sensors, object recognition, object tracking, object fusion, etc. are also part of perception. Recent perception algorithms mostly use deep neural networks (Mitrev 2019).

The higher-level instance of the agent is the route planning system, which uses maps to determine the route to be followed. The planning system combines processed information about the environment (from the sensors) with defined policies and guidelines about how to navigate the environment. Planning often uses maps and rule catalogs that define "valid" driving procedures. Based on the selected route, the next traffic situation is then predicted and the trajectory and driving behavior are planned (Mitrev 2019).

For planning, policy gradient methods (Sect. 8.4) can be used, which determine the action a to be chosen by an estimated action distribution $\hat{\pi}(a|s; w)$ based on the current state (sensor information, etc.). The parameter vector w of this model can be estimated from observational data (Sect. 8.4.2).

The simplest approach uses pure planning techniques instead, such as model predictive control (MPC), to select actions. When a new observation of the environment is available, the agent computes a plan that is optimal with respect to the model. The plan describes all actions within a fixed time window. Rewards beyond the time window can be considered using learned value functions. The agent then executes the first action of the plan and discards the rest. If the next observation is available, the procedure is triggered again.

The control system ensures that the intentions and goals derived from the planning system are translated into actions. The control system tells the hardware (the actuators) what actions are required (acceleration, braking, steering, etc.). The control module also aims to make the driving movements smooth and pleasant.

Two of the most important players in the development of self-driving cars are Tesla and Waymo. Tesla relies on inexpensive sensors, i.e. cameras and radar, whereas Waymo also uses Lidar, which can measure distances to surrounding objects relatively accurately. While a Tesla car drives an average of 5 km before the supervisor has to intervene, this is only the case with Waymo after more than 2000 km (Giacaglia 2019). Waymo is therefore currently ahead in the development of fully automated cars.

8.5.3 *Fine-Tuning Through Simulation*

The agent of a self-driving car has to cope with a very wide variation of situations. Figure 8.34 shows a snowy, winding road in the mountains. In such a situation, the agent must expect oncoming traffic and drive with appropriate caution. In particular, he must behave in such a way that he can also compensate for driving errors of the other drivers and, if necessary, brake early.

To create complex AI algorithms, large amounts of sensor data from driving operations are required. However, the cost of collecting this data and, in particular,

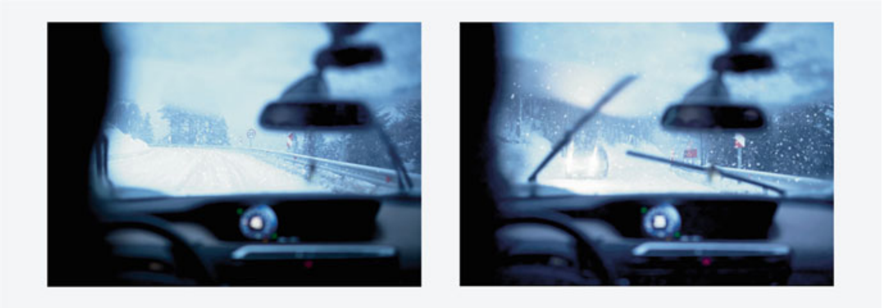
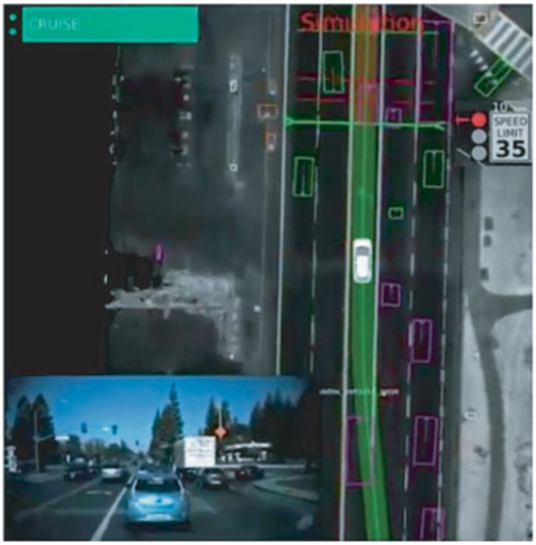


Fig. 8.34 A car is driving along a snow-covered mountain road. Suddenly, in a curve, another car comes towards him. The agent must anticipate the possibility of such situations. Image credits in Appendix A.3

Fig. 8.35 Simulated driving controlled by the agent using the Carcraft simulation environment. Variations of the previous driving strategy can be tested here (Anguelov 2019). Image credits in Appendix A.3



of collecting sufficient data for critical (rarely occurring) driving scenarios is very high. This makes it difficult to train AI models that must also react appropriately in exceptional situations.

One of Waymo's development focuses, therefore, is testing its software in a simulated world. Waymo developed a simulated virtual world called Carcraft, named after the game World of Warcraft. Carcraft was used to simulate thousands of scenarios to test and improve the agent.

In Carcraft, Waymo created fully modeled versions of cities like Austin, Mountain View, Phoenix, etc. Different scenarios were tested in many simulated cars—about 25,000 of them at the same time. In total, cars in this virtual world drive about 16 million km per day (Fig. 8.35). In 2018 alone, they traveled about 11.6 billion virtual km (Anguelov 2019). This is much more than the 16 million km that real Waymo cars drove on public roads.

Bansal et al. (2018) explain in detail how this simulation environment does not simply run variations of routes traveled. Rather, separate simulation models are created for the actors involved (cars, pedestrians, bicycles, etc.) that can plausibly control and change their behavior. Sometimes this happens by rule violation, e.g. when another car runs a red light or a pedestrian abruptly crosses the lane. The agent should consider such behavior and be prepared for it.

There are now approaches in which the agent simultaneously learns a simulation model for the environment and a DNN to approximate the policy (Ha & Schmidhuber 2018). Here, the details relevant for the control are automatically reproduced and other parts are left out.

Deep neural networks for sensor processing and control play a major role. Here, common CNN architectures are used for image processing and for combining the different sensor signals. For example, Gao et al. (2018) present a CNN that fuses camera signals and Lidar information into an environment representation. The Tesla agent uses a variant of the Google Inception neural network with five times the number of parameters to detect objects in the environment (Giacaglia 2019).

Often, the network architecture is modified and improved by automatic procedures. Optimization criteria include prediction accuracy, robustness to disturbances (rain, fog, etc.), and computational cost. Zoph et al. (2018) show how to structurally optimize new architectures for small datasets first, and then obtain the global best solution by evaluating the best 100 variants on the large dataset (Fig. 8.36).

It is difficult to predict when fully autonomous cars will be on the roads in larger numbers. Manufacturers mostly say they will deliver such vehicles to customers by 2021. Fridman (2019) launched a poll on Facebook and Twitter in early 2019 (Fig. 8.37): "Which company will be the first to deliver more than 10,000 fully autonomous cars driving on public roads without a safety driver?" 57% of the 3000 respondents tipped Tesla, while 21% saw Waymo ahead. 14% said another company would be first, and 8% said no one could do it in the next 50 years. So it remains exciting.

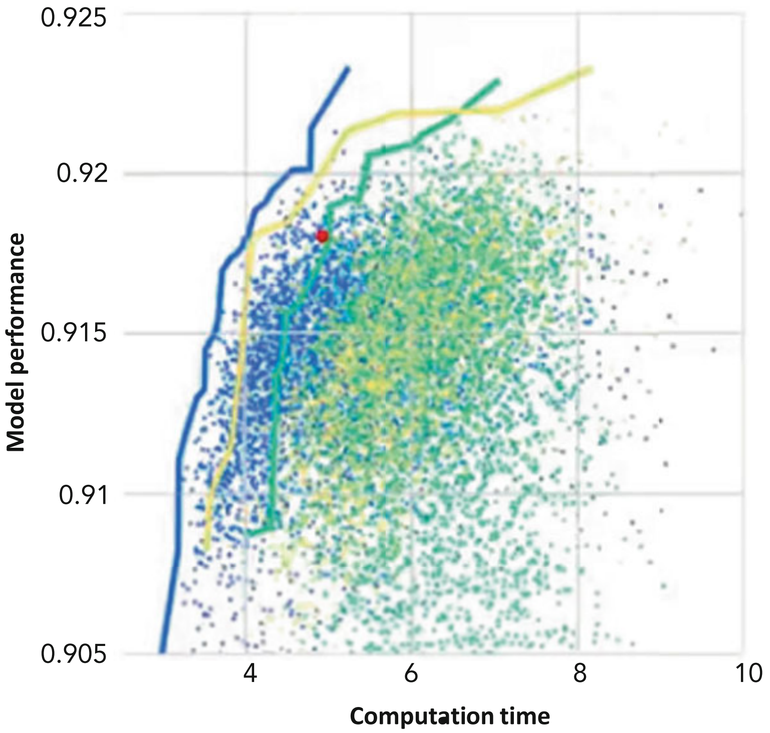
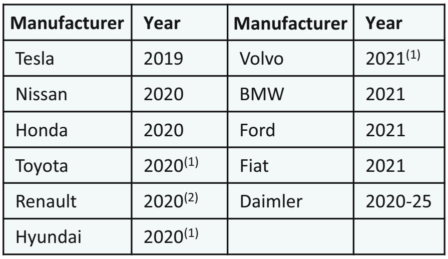


Fig. 8.36 Automatic Hyperparameter Optimization. The previous best solution (red dot) was significantly improved in terms of both model quality and computation time (Anguelov 2019). Image credits in Appendix A.3

Fig. 8.37 Anouncement by automakers for the availability of fully autonomous cars for consumers. (1) on highways, (2) in cities. Source: (Fridman 2019)



Manufacturer	Year	Manufacturer	Year
Tesla	2019	Volvo	2021(1)
Nissan	2020	BMW	2021
Honda	2020	Ford	2021
Toyota	2020(1)	Fiat	2021
Renault	2020(2)	Daimler	2020-25
Hyundai	2020(1)		

Globally, 1.35 million people are killed in road traffic accidents every year. This means that one person dies in a traffic accident every 24 seconds (WHO 2018). There are estimates that this number will drop considerably with autonomous cars. So far, there have been two deaths caused by partially autonomous cars, but no deaths yet with fully autonomous cars (Murasaki 2019). But even the Tesla semi-autonomous cars are safer than normal cars. While an accident occurs about every 0.7 million km in normal cars, a Tesla car with partially autonomous control enabled drove an average of 4.7 million km before an accident occurred. The number of accidents was thus reduced by 85% (Murasaki 2019).

8.5.4 Reliability of Reinforcement Learning

8.5.5 Simulation-Trained Models Difficult to Transfer

Pinto et al. (2017) observe that most approaches to reinforcement learning are difficult to apply. Reasons for this are:

- The difference between the simulated environment and the real world is so great that models for policy learning are often not transferable.
- Even when learning takes place in the real world, the amount of available data is too small and generalization therefore often fails.

Therefore, the authors suggest to introduce an adversary during training that can exert additional perturbation forces and destabilizations up to a given size. The adversary is trained to apply perturbation forces in the system just enough to maximally disrupt the agent (Fig. 8.38). The learning algorithm must compensate for these perturbations and thus automatically becomes more robust. The model has very strong similarities to the generative adversarial models in the image domain discussed later (Sect. 9.1.2). The results of the experiments are encouraging, and much more robust control strategies are learned.

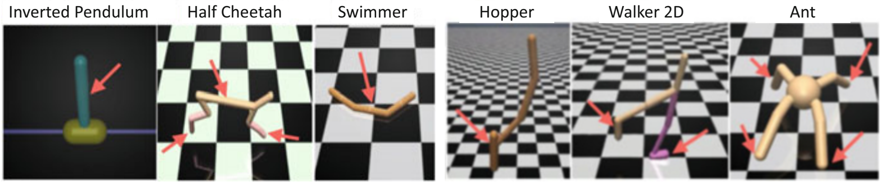


Fig. 8.38 Adversarial reinforcement learning randomly applies perturbation forces (arrows) to the different agents. This makes the agents more robust (Pinto et al. 2017). Image credits in Appendix A.3

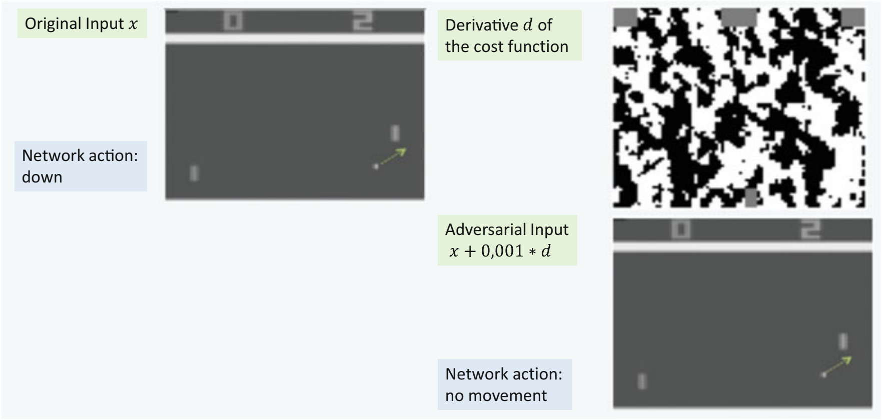

Original Input x
Network action: down
Derivative d of the cost function
Adversarial Input $x + 0{,}001 * d$
Network action: no movement

Fig. 8.39 The adversarial input image (bottom right) is the sum of the original input image x (left) and the sign of the derivative of the cost function d (top right). Since d is multiplied by a factor of 0.001, the change is not visible on the adversarial input. The DNN chooses the “down” action on the left input, but the “no move” action on the right input (source: Chen et al. 2019). Image credits in Appendix A.3

8.5.6 Adversarial Attacks on Reinforcement Learning Models

If a reinforcement learning model receives its input in the form of pixel images, these images can be changed imperceptibly to fool the DNN, which approximates the Q-Function (Sect. 8.2.7), as demonstrated in Sect. 5.12.2. Figure 8.39 shows an adversarial example for the Atari game Pong (Chen et al. 2019). The state x of the game is represented by a picture, and the arrow shows the direction of movement of the ball. The adversarial change is the sign of the derivative of the cost function with respect to x. It is multiplied by a small factor and is not detectable by the observer in the result. This completely changes the action output by the DNN.

Chen et al. (2019) point out that these changes can be computed at each step and therefore the agent does not take the correct actions. There are a large number of other strategies for perturbing reinforcement learning models, including approaches where the model architecture does not need to be known.

For a total of eight attack methods, Chen et al. (2019) examine the effectiveness of 20 different defense strategies. It turns out that effective countermeasures exist for most attacks. Moreover, the majority of attack methods require access to the model. Black-box attacks are rare, and very difficult for the attacker to carry out in practice. The authors suggest that because of increased research activity in this area, very reliable reinforcement learning systems will be available in the future.

8.6 Summary and Trends

In many application domains, a series of actions are to be determined over a certain period of time with the goal of achieving the highest possible sum of rewards over time. A model (agent) can select these actions based on the current information about the situation (state). Examples include board games, robot controls, and self-driving cars. DNNs are used here to assign a sum of expected rewards to a state to determine the best action. They are particularly suitable because they can approximate the underlying function well. A Q-Network $Q(s_t, a; w)$ estimates an expected reward sum for each state-action pair. For observed successive states and actions, one can derive an equation that can be used to train a Q-Network.

A stochastic policy is suitable for decision situations with random influences and determines an optimal probability distribution over the possible actions for a given state. The policy gradient models compute the gradient from a sample of model runs and can thus train the model. Unlike previous DNN, the training data is generated using the simulated or real environment during training. In gaming, DNNs have achieved sustained success, beating humans on any board game Sect. 2.6.1. But DNN have also recently defeated the world champion team on complex real-time video games with teams of multiple players (Sect. 8.4.5). However, these are simulated environments where it was possible to generate and use training data with thousands of years of human gaming experience. A prominent application is autonomous vehicles (Sect. 8.5), which are currently being developed by many companies. It is not yet clear how successful these approaches are in real decision problems in daily life.

Trends

- There is a discussion about preparing robots for their tasks through playful activities. Even children practice complex behavioral patterns through playful engagement with their environment. In doing so, they learn the properties of objects in the environment and can use this knowledge to solve new tasks via transfer learning. Lynch et al. (2020) show that robots trained via playful actions can successfully perform complex given tasks

(continued)

in 85% of the cases, while robots that used demonstrations by experts for training were only successful at 70% of the trials.

- Inspired by the success of AlphaFold (Sect. 2.1.2), it is expected that reinforcement learning techniques will find real applications in industrial settings, e.g., drug design, vehicle control, traffic guidance, etc.
- In the language domain, the use of unsupervised trained embeddings has become common, which are then adapted to the actual task. It is foreseeable that such embeddings will also be used in models of reinforcement learning.
- In reinforcement learning there is the problem of overfitting, not only because one trains and tests the model on the same data. Therefore, approaches are being developed that avoid this type of testing.
- In situations with multiple agents, it is advantageous if agents communicate with each other. Currently, several benchmark tasks are being built in which such communication can be tested.
- Using a trained agent in a changed environment is difficult because the agent must retrain the new goals and environment responses. Zhong et al. (2019) train an agent that must take its goals and environment descriptions from a text document. This is the basis for transfer learning via new “game rules” that define new tasks for the agent.

References

Anguelov, D. (2019). *Taming the long tail of autonomous driving challenges.* Vortrag Im Rahmen Des MIT Kurses “Deep Learning for Self-Driving Cars”. 15-01-2019, Cambridge. https://www.youtube.com/watch?v=Q0nGo2-y0xY

Badia, A.P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., Blundell, C. (2020). Agent57: Outperforming the Atari Human Benchmark. arXiv: 2003.13350.

Bansal, M., Krizhevsky, A., & Ogale, A. (2018). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv: 1812.03079.

Bellman, R. (1957). *Dynamic programming*. Princeton: Princeton University Press. Republished 2003: Dover, ISBN 0-486-42809-5.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv: 1606.01540.

Byford, S. (2017). *Google’s AlphaGo AI defeats world go number one ke jie*. The Verge. Regtrieved September 01, 2023, from https://www.theverge.com/2017/5/23/15679110/go-alphago-ke-jie-match-google-deepmind-ai-2017

Chen, T., Liu, J., Xiang, Y., Niu, W., Tong, E., & Han, Z. (2019). Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity, 2*(1), 11.

Dickmanns, E. D., & Graefe, V. (1988). Dynamic monocular machine vision. *Machine Vision and Applications, 1*(4), 223–240.

Fridman, L. (2019). *Self Driving Cars: State of the Art 2019.* https://www.youtube.com/watch?v=sRxaMDDMWQQ

Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics, 14*(9), 4224–4231

Giacaglia, G. (2019). Self-driving cars. medium.com. https://medium.com/@giacaglia/self-driving-cars-f921d75f46c7

Ha, D., & Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems, 31*, 2450–2462.

Hogget, R. (2019). *1951 – SNARC Maze Solver – Minsky / Edmonds (American)*. http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/

Igl, M., Ciosek, K., Li, Y., Tschiatschek, S., Zhang, C., Devlin, S., & Hofmann, K. (2019). Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in Neural Information Processing Systems* (pp. 13978–13990).

Jin, C., Allen-Zhu, Z., Bubeck, S., & Jordan, M. I. (2018). Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems* (pp. 4863–4873).

Levine, S. (2018). *Policy gradients. CS 294-112: Deep reinforcment learning*. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf

Li, Y. (2018). Deep reinforcement learning. arXiv: 1810.06339.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., & Sermanet, P. (2020). Learning latent plans from play. In *The Conference on Robot Learning* (pp. 1113–1132).

Mitrev, D. (2019). Who leads the self-driving cars race? State-of-affairs in autonomous driving. In *Neurohive* 30-01-2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Murasaki, M. (2019). *How many people have died in self-driving cars?*. https://www.quora.com/How-many-people-have-died-in-self-driving-cars

OpenAI. (2019). *OpenAI five defeats dota 2 world champions*. OpenAI. Retrieved January 09, 2023, from https://openai.com/blog/openai-five-defeats-dota-2-world-champions/

OpenAI-Five. (2018). *OpenAI five - Open Ai blog 25.06.2018*. https://openai.com/blog/openai-five/

Osband, I., Blundell, C., Pritzel, A., & Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems* (vol. 29).

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. arXiv: 1703.02702.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. arXiv: 1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv: 1707.06347.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.

Smith, C. (2020). Computers already learn from us. But can they teach themselves? In *NewYork Times* 12-04-2020.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems* (pp. 1057–1063).

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3–4), 279–292.

WHO (2018). *Global status report on road safety 2018*. Retrieved January 09, 2023, from https://www.who.int/publications-detail-redirect/9789241565684

Wikiludia. (2018). *Schere, Stein, Papier*. https://wikiludia.mathematik.uni-muenchen.de/wiki/index.php?title{$=$}Stein-Schere-Papier

Yang, Z., Xie, Y., & Wang, Z. (2019). A theoretical analysis of deep Q-learning. arXiv: 1901.00137.

Yoon, C. (2019). Understanding actor critic methods and A2C. https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f. Accessed on 08.05.2019

Zhong, V., Rocktäschel, T., & Grefenstette, E. (2019). RTFM: Generalising to new environment dynamics via reading. In *International Conference on Learning Representations*

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern Recognition* (pp. 8697–8710).

Chapter 9
Creative Artificial Intelligence and Emotions

Abstract This chapter shows that deep neural networks (DNNs) can creatively generate novel images, text, music, and dialogue. In the case of images, generative adversarial networks (GANs) are able to create images with specific properties or style features. In addition, they can convert images from one type to another, such as a photo to a painting. For authoring texts, there are language models that can invent new complex stories and formulate them in fluent language. Music DNNs are trained with the notes of musical pieces and can “compose” new pieces of music that experts say achieve good quality. At the end of the chapter, intelligent speech assistants are discussed, which are able to recognize the emotional state of the conversation partner in his or her dialog contributions. They can respond appropriately, providing creative and focused answers so that the counterpart feels understood and is motivated to continue the conversation. In Asia, there are chatbots of this kind with hundreds of millions of users.

Creativity is an ability we tend to think of as uniquely human. It creates something that is new or original and in many cases is judged to be useful and usable. Computers are generally being perceived as machines that stubbornly execute a predefined program, and therefore cannot produce anything surprising. The following chapter will show that deep neural networks can creatively deliver new types of images, text and music. It will also be shown that computer programs can recognize people’s emotions and, to a certain extent, react to them adequately and creatively.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_9

9.1 Image Creation with Generative Adversarial Networks (GAN)

9.1.1 Forger and Art Expert

Suppose we want to use DNN to generate new images in the style of an artist, such as Vincent van Gogh. We could try to train a generator network (forger) to generate van Gogh images using a collection of van Gogh images. However, the forger should not create the same images as in the training set, but alternative images with new subjects that match van Gogh's style. But it would be very difficult to define what corresponds to van Gogh's style and what does not - defining a loss function for such a model is very difficult.

It would be much easier, of course, if we had a classification model capable to distinguish a genuine Van Gogh from a fake. This discriminator network should thus act as an art expert specialized in Van Gogh and would have to be trained on the basis of genuine and fake paintings.

9.1.2 Generator and Discriminator

The generative adversarial network (GAN) contains such a generator and a discriminator and tries to train these two networks simultaneously. As shown in Fig. 9.1, a training set with images of one genre is needed, e.g., van Gogh. The generator network $G(x)$ is given a vector x of random numbers and generates an image b from this vector. The discriminator network $D(b)$ is randomly given b, a genuine image

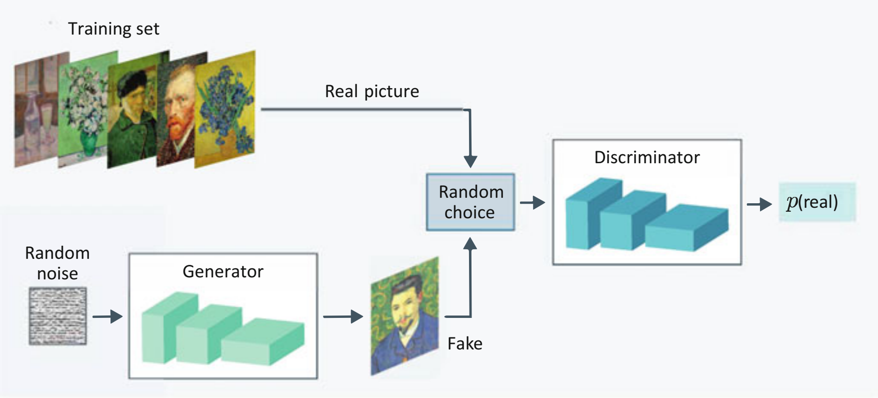


Fig. 9.1 Generative adversarial network for creating van Gogh images. The generator network gets random numbers as input and produces a synthetic image. The discriminator network randomly receives either a real image from the training set or a synthetic image. It must estimate the probability of it being a real image or a fake. Both networks are trained alternately. Image credits in Appendix A.3

or a fake, and must distinguish between these two cases by estimating a probability of authenticity (Goodfellow et al. 2014).

Thus, both networks are opponents in a game: the generator should produce fake images that cannot be distinguished from genuine images by the discriminator; the discriminator should be able to detect the fake images with a high degree of certainty.

Both networks are initialized randomly at the beginning. Usually the discriminator quickly learns to distinguish the first fakes from the real images. After a longer run, the generator can then generate very good fakes, if it has learned the style features of the training set until the discriminator can no longer distinguish them from real images.

9.1.3 Optimization Criteria for Generator and Discriminator

The generator $G(x)$ is a DNN with parameter vector w_g. It receives a random vector x as input and generates a fake c, i.e., $c = G_{w_g}(x)$. The discriminator network $D(b)$ has the parameter vector w_d, receives an image b as input, and computes the probability $p(b \text{ is real}) = D_{w_d}(b)$. If n genuine images b_i from the training set and if m fakes $c_j = G(x_j)$ are available, the discriminator has the objective function (Goodfellow et al. 2014)

$$J_D(w_d) = \overbrace{\Big(\underbrace{D_{w_d}(b_1)}_{\text{Probability: 1st real image classified as real}} * \cdots * D_{w_d}(b_n)\Big)^{1/n}}^{\text{Probability: all real images classified as real}} * \overbrace{\Big(\big(1 - D_{w_d}(c_1)\big) * \cdots * \underbrace{\big(1 - D_{w_d}(c_m)\big)}_{\text{Probability: m-th fake image classified as not real}}\Big)^{1/m}}^{\text{Probability: all fake images classified as not real}}$$

because it wants to increase the probability for the real images and reduce the probability for the fakes. As with logistic regression, the logarithm is applied to the formula (Sect. 3.5.2) and a sum is obtained which is easier to compute. The discriminator can be optimized for this optimization criterion with the stochastic gradient descent algorithm.

The generator, on the other hand, wants to increase the probabilities for the fake images, but can only influence its own parameters w_g. However, it knows the internal structure of the discriminator. Therefore, given w_d, it optimizes the following criterion.

$$J_G(w_g) = \underbrace{\Big(D_{w_d}\big(G_{w_g}(x_1)\big) * \cdots * \underbrace{D_{w_d}\big(G_{w_g}(x_m)\big)}_{\text{Probability: } m\text{-th fake image classified as real}}\Big)^{1/m}}_{\text{Probability: all fake images classified as real}}$$

The generator should have the following features:

- ReLU activations, except in the last layer.
- as few fully connected layers as possible.
- Transposed convolution layers (Sec. 5.8.4) to increase image resolution
- Batchnorm regularization.

The discriminator should have the following features:

- Convolution layers with step size > 1 instead of pooling layers.
- Batchnorm regularization
- as few fully connected layers as possible
- Leaky-ReLU activations

Fig. 9.2 Recommended architectural features for GANs

because it has to maximize $p(\text{real})$ for the fakes. By applying the logarithm, the criterion is also transformed into a sum. With fixed w_d the generator parameter w_g can be optimized for this optimization criterion with the stochastic gradient descent algorithm. In practice, the training of both networks alternates: the discriminator is optimized with minibatches of real and fake images, and then the generator is optimized with minibatches of fake images.

Initial experiments with GANs used simple generator and discriminator models. Higher computational power is now available, allowing deep CNNs to be used. Radford et al. (2016) propose to use a deep convolutional GAN (DCGAN) architecture with features described in Fig. 9.2. Nevertheless, training GANs is particularly difficult. Chintala et al. (2018) therefore provide some guidance on how to proceed successfully.

9.1.4 Results of Generative Adversarial Networks

Radford et al. (2016) define a DCGAN with five layers that generate fields of size 4×4, 8×8, 16×16, 32×32, and 64×64. They trained the network on photos of bedrooms from the Large-Scale Scene Understanding (LSUN) dataset with about 3 million training examples. The basic features of bedroom images were learned, but the geometry of the furniture and its perspective were often not plausible.

Karras et al. (2018) introduced a GAN that increases the field sizes of both the generator and the discriminator during training. This speeds up the training and also allows very high resolutions of 1024×1024. In addition, they suggested further improvements that increased the variability of the generated images. The results for the bedroom dataset can be seen in Fig. 9.3 at a resolution of 256×256. Here the geometry and perspective of the rooms are very realistic. Fig. 9.4 shows synthetic images for other categories from the LSUN dataset.

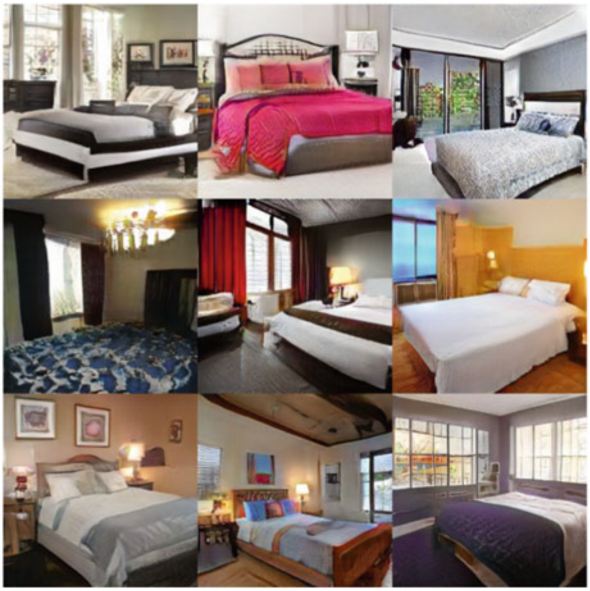


Fig. 9.3 Images of bedrooms generated with Progressive GANs that produce a fairly realistic rendering (Karras et al. 2018). Image credits in Appendix A.3

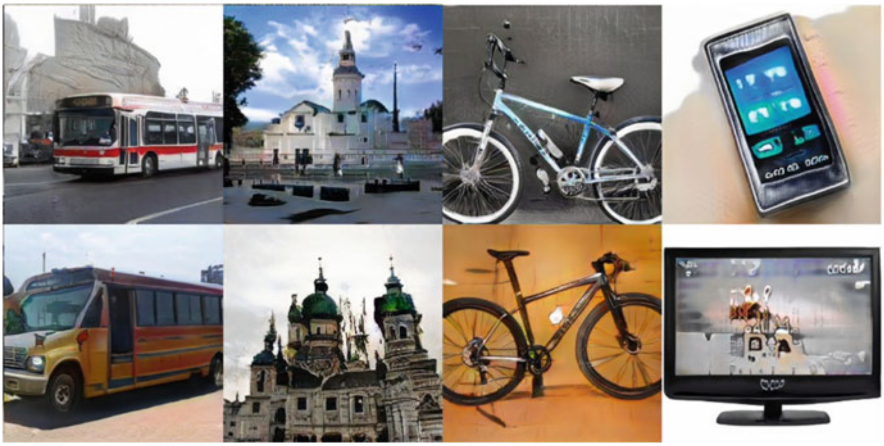


Fig. 9.4 Synthetic 256 × 256 images of the categories bus, church, bicycle and TV monitor from the LSUN dataset (Karras et al. 2018). Image credits in Appendix A.3

The CelebA-HQ dataset contains 30,000 high-resolution images of celebrities, with each image annotated with 40 attributes (e.g., wearing glasses, smiling, wearing a hat, . . .). The images in this dataset show people in very different poses and with very diverse backgrounds. The deep neural network was trained on this data using eight Tesla V100 GPUs for 4 days.

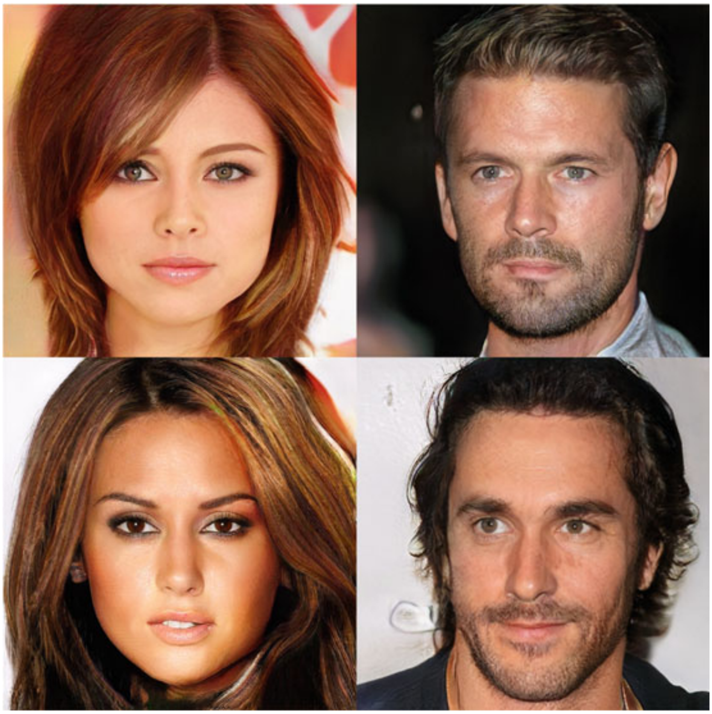


Fig. 9.5 These persons do not exist! High resolution images created with the Progressive GAN (Karras et al. 2018). Image credits in Appendix A.3

Figures 9.5 and 2.24 show images generated by the Progressive GAN (Karras et al. 2018) at 1024 × 1024 resolution. The images are highly detailed and realistic. A video shows how the images were generated as the resolution gradually increased. Nevertheless, the authors believe that the road to photorealism of the images is still far.

Brock et al. (2019) train conditional GANs, where the generator is instructed to generate images of a class by entering the class name. Through a series of algorithmic modifications, they are able to train GANs with a very large number of parameters. Figure 9.6 shows generated images for different classes that exhibit a very high levels of detail and realism. The quality drops in some image classes for which fewer training examples are available, or which are very diverse. The authors demonstrate that the generated images show new viewpoints and individuals and are not the modification of a training image.

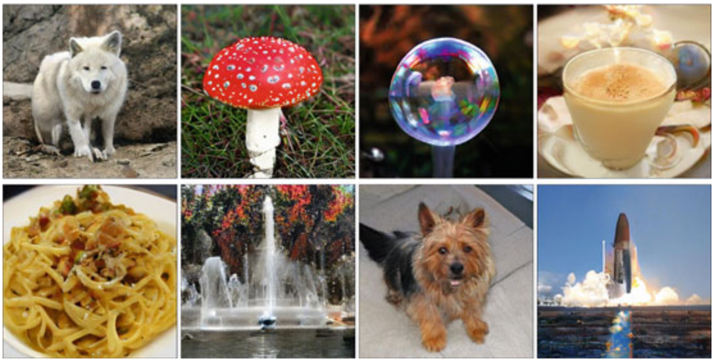


Fig. 9.6 Generated images of size 512 × 512 for different classes. ImageNet was used as the training data (Brock et al. 2019). Image credits in Appendix A.3

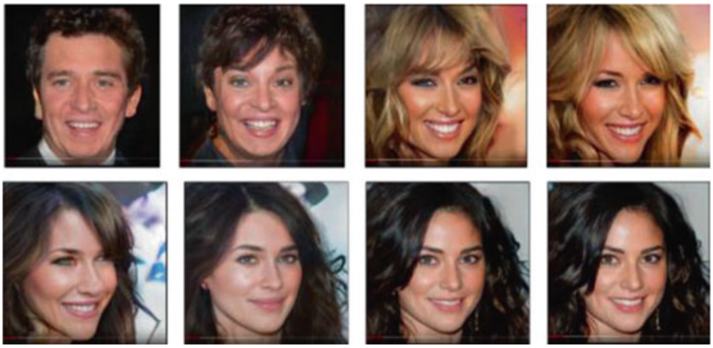


Fig. 9.7 Images created by interpolation of the input vectors. The right and left images of each line were interpolated (Karras et al. 2018). Image credits in Appendix A.3

9.1.5 Interpolation Between Images

After training a GAN, the generator maps a vector x of random numbers to an image. One can consider the vector as an embedding of the image, which characterizes its properties. Figure 9.7 shows images of the Progressive GAN (Karras et al. 2018) created by interpolating between two such embeddings. They illustrate that interpolation in the space of input vectors leads to continuous transitions between images. Each image represents a plausible face of a person.

For the embeddings of words, we have seen that the embedding vectors often have a linear relationship to each other, e.g. $x_{\text{king}} - x_{\text{man}} + x_{\text{woman}}$ resulted in a vector whose nearest neighbor vector was x_{queen} (Sect. 6.2.2). Similar computations can be done with the vectors corresponding to the GAN images. It turned out that one has to form prototypes as an average of several vectors. If the prototype vectors are averaged in the following way: $x_{\text{man with glasses}} - x_{\text{man without glasses}} +$

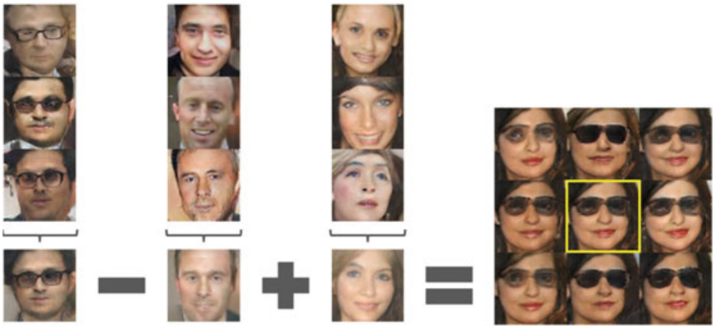


Fig. 9.8 Vectors of the prototypes “man with glasses” (left), “man without glasses” (2nd from left) and “woman without glasses” (center) are formed, corresponding to the images in the bottom row. If $x_{\text{man with glasses}} - x_{\text{man without glasses}} + x_{\text{woman without glasses}}$ is computed, the result vector corresponds to the yellow bordered face. Adding noise in the range [−0.25, +0.25] to this vector yields the remaining faces (Radford et al. 2016). Image credits in Appendix A.3

$x_{\text{woman without glasses}}$, the yellow-bordered face is generated from the result vector (Radford et al. 2016). This shows that the concept “glasses” can be merged with the concept “woman” (Fig. 9.8).

Each subnetwork of the GAN can become too “good” during training. If the discriminator is too good, the generator will not receive enough gradient information to improve its images. If the generator is too good, it will exploit weaknesses in the discriminator. These effects can be controlled by adjusting the learning rates. Another problem may be that the generator produces correct images with very few subjects, i.e., too little variation.

There are now hundreds of different GAN approaches (Hindupur 2018). Many authors worked on improving the loss functions (e.g. Wasserstein GAN, LSGAN etc.). Further, conditional GANs can be used to select the type of images generated. Nevertheless, the training of GANs is still complex. With so called Variational Autoencoders one can also generate images. However, the resulting images are usually blurrier.

9.1.6 Transformation of Images

Often you want to translate one version of an image to another. Figure 9.9 shows some examples of such transformations: Generating a color image from a black and white image, generating a photograph from a sketch, or generating a map from an aerial photograph. Isola et al. (2017) propose a method based on GANs for this purpose. Consider the example shown in Fig. 9.10: Here, the generator produces an image from a sketch. The discriminator randomly receives as input a pair consisting of a sketch and a photograph, and must decide whether this pair is real. The sketch

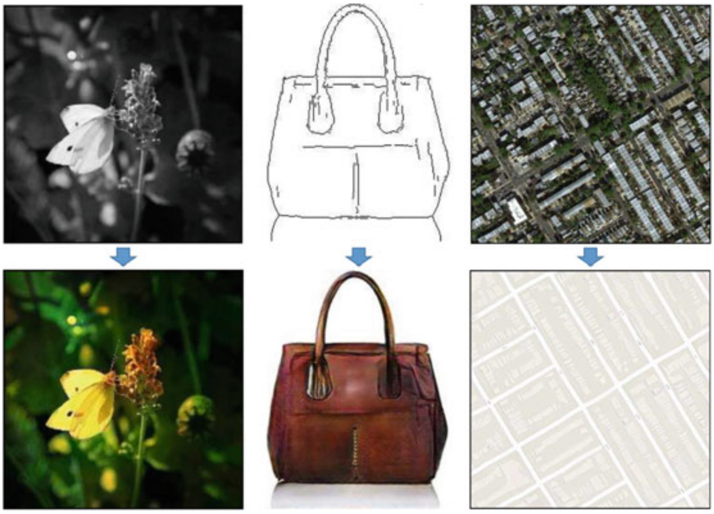


Fig. 9.9 Examples for the transformation of images. The inputs (top row) are transformed into the corresponding output images (bottom row). For each example type, a separate model is trained (Isola et al. 2017). Image credits in Appendix A.3

acts as a boundary condition for the image generation, hence the process is called Conditional GAN.

The generator has a structure similar to Fig. 9.13. First, the image resolution is reduced by CNNs with step size larger than 1 and the number of features is increased. The generated representation is then transformed with residual blocks as in ResNet (Sect. 5.5.2). The bypass allows relevant features to be passed easily and facilitates training. Finally, the image is expanded back to full resolution using Transposed Convolution layers (Sect. 5.8.4).

The discriminator focuses on the differences of local patterns between sketch and image and is therefore called PatchGAN (Isola et al. 2017). Both subnetworks are again trained alternately. The training data consists of the pairs of the two image variants, e.g., sketch and photo.

Networks were trained for a number of different tasks, e.g., map $\longleftrightarrow$ aerial photo (Fig. 9.11), black-and-white photo $\rightarrow$ color photo, sketch $\longleftrightarrow$ photo, sketch $\rightarrow$ portrait image, 3D sketch $\rightarrow$ street view, photo with background $\rightarrow$ photo without background. The results were so convincing that human viewers often perceived the generated results as real scenes.

One application for image transformation is to increase the resolution of images (Ledig et al. 2017). For a specific domain, e.g. images of people, pairs of images with low and high resolution are used for training. Another task is the reconstruction of missing parts of an image.

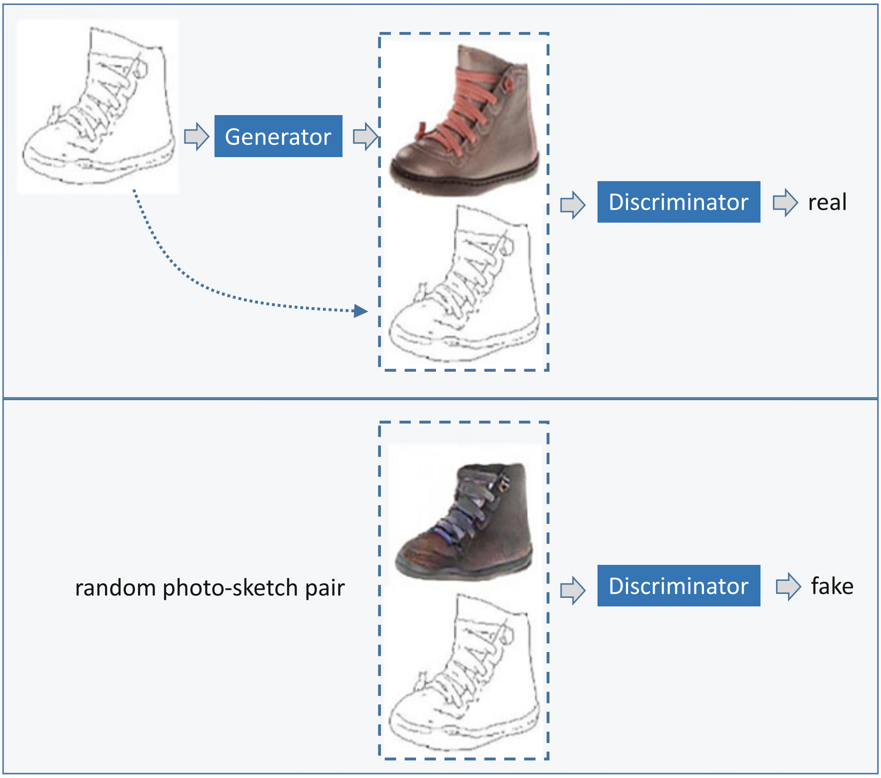


Fig. 9.10 The generator creates a photo of the object from the sketch. The discriminator receives a sketch-photo pair as input and must decide whether this pair is real or a fake image produced by the generator (Isola et al. 2017). Image credits in Appendix A.3

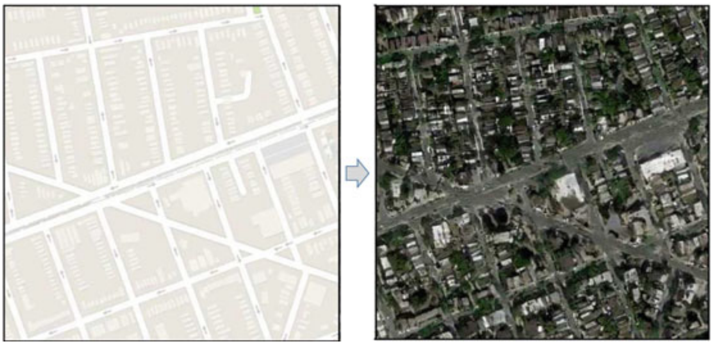


Fig. 9.11 The result of an image transformation from a map to an aerial photo (Isola et al. 2017). Image credits in Appendix A.3

9.1.7 Transformation of Images Without Training Pairs

If one wants to transform the photo of a horse into the image of a zebra, no training data is available for this, since horses never have the stripes of zebras. Thus, the method for transforming the images described previously is not applicable. J.-Y. Zhu et al. (2017) have developed a model called CycleGAN that allows such transformations when one has two independent samples of images from the corresponding domains, but no pairs with a horse and the corresponding zebra.

Starting point are two generator nets $G^{H \to Z}$ and $G^{Z \to H}$, which generate zebras from horses and horses from zebras (Fig. 9.12). The generator nets try to generate zebras and horses, which look as real as possible. Two discriminator nets D^Z and D^H are used, as in a GAN, to check whether the generated images are really zebras and horses.

However, this is still not enough to ensure that the zebras generated from horses actually resemble the original horse. For this purpose, the authors use the back transformation $G^{Z \to P}$, which generates again a zebra from the generated horse. The difference from the initial image h_0 is computed and an additional cost term measures whether the absolute value of the difference $\|G^{Z \to H}(G^{H \to Z}(h_0)) - h_0\|1$ is small. This ensures that the output image of the horse and the generated image differ only slightly in their pixels. Similarly, a horse image is generated from a zebra image to train the inverse generator. The architecture of the generators is shown in Fig. 9.13. As discriminator PatchGan (Sect. 9.1.6) was chosen again, which focuses on local differences.

The method was applied to a number of different benchmark datasets and always had better accuracy (J.-Y. Zhu et al. 2017) compared to alternative approaches. An example application is the transformation of photographs into paintings in the style of van Gogh (Fig. 9.14 above). It is clear that the reverse transformation ensures that the paintings are structurally the same, differing only in local style

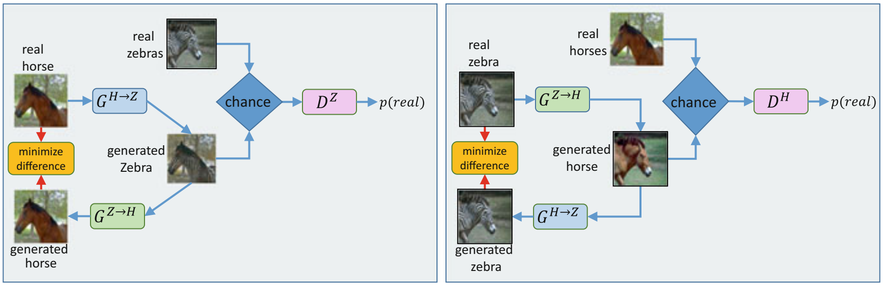


Fig. 9.12 The training of CycleGAN creates two generators: Training of a generator $G^{H \to Z}$ converting a horse image to a zebra image by a GAN (left) and a generator $G^{Z \to H}$ converting a zebra image to a horse image by another GAN (right). On both sides, the image is back-transformed by the other generator and the difference from the original images is minimized (J.-Y. Zhu et al. 2017). Image credits in Appendix A.3

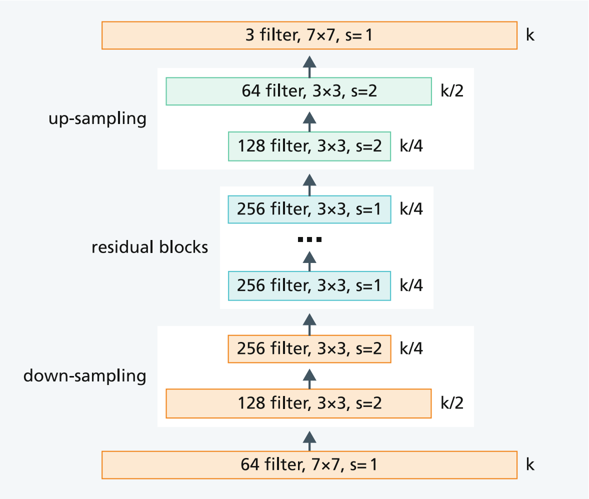


Fig. 9.13 Architecture of the CycleGAN generator.The image with the initial size k is reduced to a size of $k/4$ by downsampling with convolutional layers and step size $s=2$. Then the image is transformed by 6 residual blocks (Sect. 5.5.2). Finally, the image is expanded back to full resolution by Transposed Convolution layers (Sect. 5.8.4) (Wolf 2018)

features. Nevertheless, larger color differences can be observed. In contrast to other approaches, the method learns the style features of a whole collection of images and not of a single image.

There are now a large number of applications of the process. Examples include transferring a photo to a different season (Fig. 9.14 center) or from daylight to twilight. In addition, one can compute enhancements to a photograph, such as softening the background (Fig. 9.14 bottom) or removing haze. Another application is the transfer of images of people into cartoon characters.

Overall, the results achieved by CycleGAN are very good - the image quality approaches that of image-pair trained methods for many tasks. This is impressive since paired transformations were trained using supervised learning and CycleGAN does not have this information available. At the time of publication, Cycle-GAN outperformed other unsupervised image translation techniques. In experiments, humans could not distinguish synthesized images from real images about 25% of the time.

9.1.8 Creative Adversarial Network

The algorithms discussed so far have generated images from certain content areas, such as pictures of van Gogh or realistic faces of celebrities. However, the goal was not necessarily to generate something new. Elgammal et al. (2017) propose an innovative system for generating art. The system learns what images of different art styles look like. The authors develop a Creative Adversarial Network (CAN) and propose modifications of its goal to enable it to generate creative art by

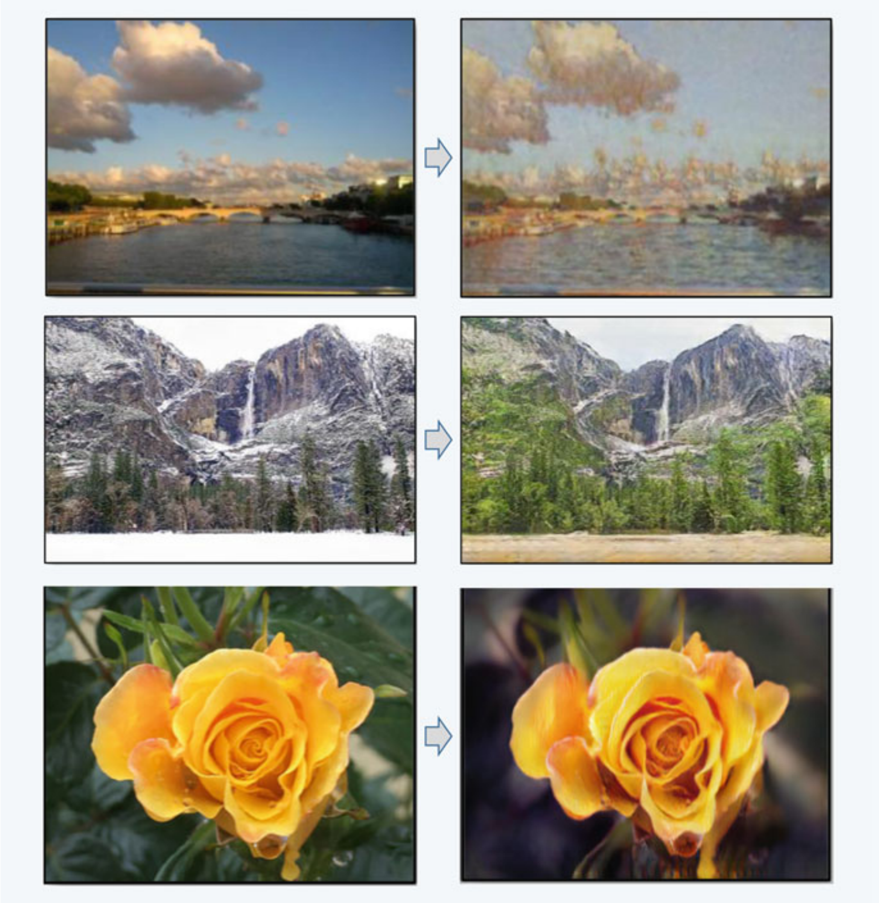


Fig. 9.14 Transformations of photos (left) computed with different CycleGAN models. Photo → Van Gogh (top), Yosemite Park in winter → Yosemite Park in summer (middle). Rose with sharp background → rose with blurred background (bottom) (Zhu et al. 2020). Image credits in Appendix A.3

maximizing deviation from established styles and minimizing deviation from the "art distribution".

The authors modify a GAN by introducing, on the one hand, a discriminator for the classification of art styles (Fig. 9.15). However, since something new shall be produced, the clear assignment to a single art style is penalized. Images are to be produced that could belong to several art styles. In addition, there is the usual discriminator that distinguishes between real artist images and generated images. The generator is now trained to produce "genuine art images" on the one hand and images that could belong to multiple art styles on the other.

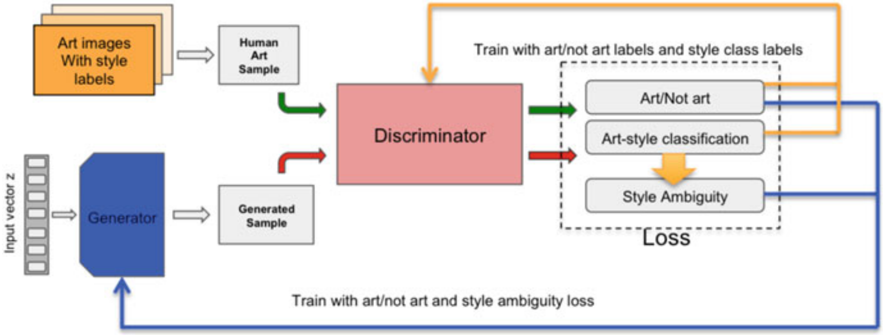


Fig. 9.15 The Creative Adversarial Network trains two discriminators: the first distinguishes generated images from real artist images. The second classifies the style of an artwork. The goal is to generate new images that can belong to different styles and therefore have a novelty value (Elgammal et al. 2017)

The authors trained their model with the freely available WikiArt dataset. This contains over 80,000 images by more than 1100 artists from the fifteenth to the twentieth century. The images are categorized into 25 styles. Figure 9.16 shows a set of CAN-generated images, which were subsequently evaluated by human viewers.

The authors then conducted experiments to compare the reaction of human viewers to synthetic images with the reaction to images created by artists. The results show that human viewers could not distinguish art generated by CAN from art generated by contemporary artists and shown at top art fairs, such as Art Basel 2016.

9.1.9 Generating Images from Text

A fascinating task is the generation of images from text descriptions. Zhang et al. (2017) use a deep recurrent network (pre-trained LSTM, Sect. 6.4) to generate an embedding vector for the description text. To be specific, they use the mean of all hidden vectors in the last layer of the RNN as the embedding vector. Now they employ two GANs to generate an image from the embedding vector. In this process, the text embeddings are enriched by a random process to achieve the necessary variability of the inputs.

The first GAN generates an image with a low resolution of 64 × 64, which contains the most important colors and shapes. From this, a second GAN produces a higher resolution image with the resolution of 256 × 256. The architecture of generators and discriminators follows the pattern described previously. By using two discriminators that compare the generated images with real 64 × 64 images and 256 × 256 images of the domain, the final generated images contain photorealistic details. The algorithm is called StackGAN.

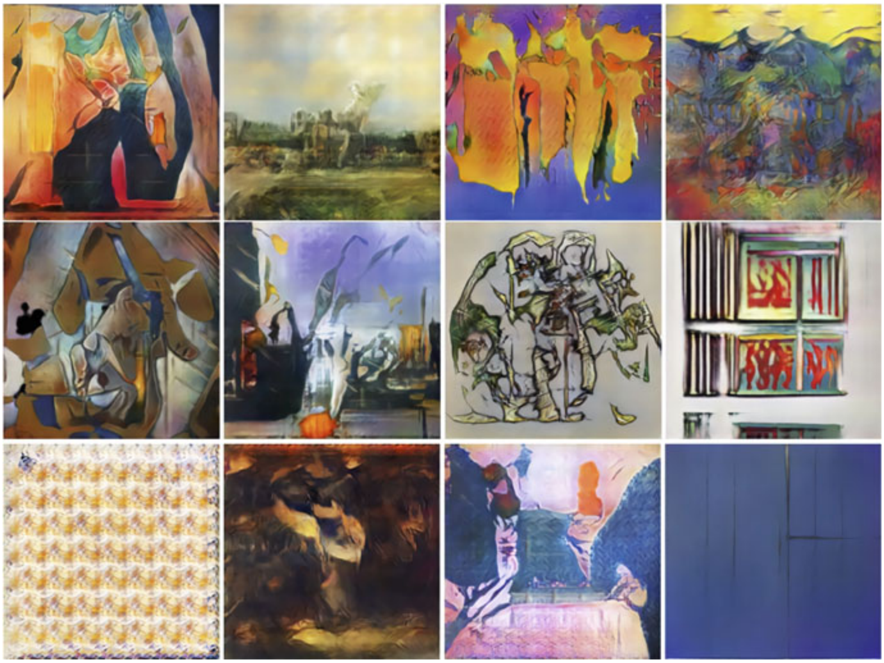


Fig. 9.16 Sample images generated by CAN. The top two rows are among the highest human-rated images, and the bottom row is among the lowest-rated images (Elgammal et al. 2017). Image credits in Appendix A.3

The StackGAN is trained on three different training dataset, which contain a textual description for each image. The generated images in Fig. 9.17 appear consistent and match the description. An evaluation by human experts indicated that the method produced better results than the available alternatives. StackGAN combines features from the training images without using them as a whole. By comparison with the training data, it could be demonstrated that the generated images shared visual properties with the training images but were composed differently (Zhang et al. 2017). As with normal GANs, smooth transitions between the features of the generated images could be observed by interpolating the text embedding vectors (see Fig. 9.18).

9.1.10 *GAN-Generated Persons in Three Dimensions*

Datagrid, a Japanese technology company, has developed a GAN model that can create 3D full-body models of people (Vardaan 2019). It uses 2d images and videos of people as inputs and generates full-body model images with high resolution (1024 × 1024) and high quality.

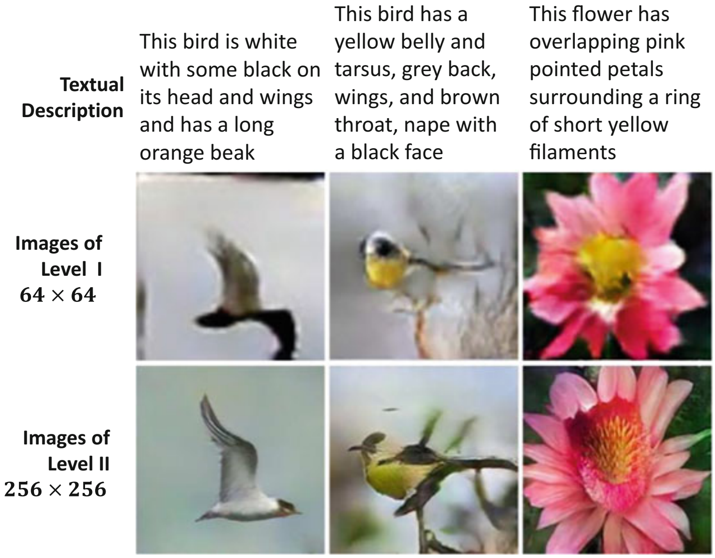

Textual Description
This bird is white with some black on its head and wings and has a long orange beak
This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face
This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments
Images of Level I 64 × 64
Images of Level II 256 × 256

Fig. 9.17 Images generated with StackGAN from text not used for training (top). The middle row shows level I results and bottom row shows level II results with higher resolution (Zhang et al. 2017). Image credits in Appendix A.3

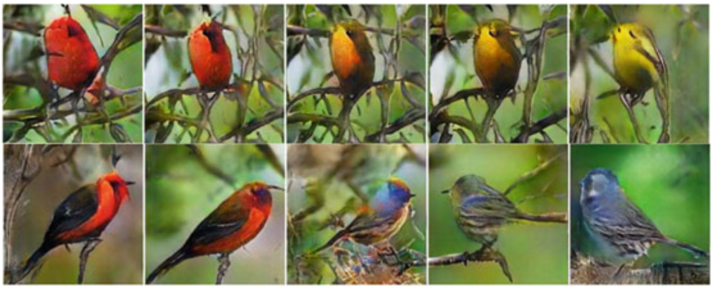


Fig. 9.18 Interpolation between description texts. In the top line interpolation between "The bird is all red" to "The bird is all yellow". In the second line, "This bird is all red with black wings and a pointed beak" to "This little blue bird has a short pointed beak and brown wings" (Zhang et al. 2017). Image credits in Appendix A.3

This GAN can produce photos that look at least at the first glance authentic to human observers, and have many realistic characteristics: Faces, hair, clothing, etc. These synthetically generated people do not exist in reality. So fashion brands or advertising agencies can use them as fashion models without any costs for photos or fees. On (Vardaan 2019) you can watch videos (Fig. 9.19) showing the models and their movements. They seem a bit clumsy at the moment, but can certainly be improved in the future. At the same time, new clothes can be "put on" the 3D-bodies and the 3D-persons can be transferred into each other by interpolation.

Fig. 9.19 Frames from a video by DataGrid showing the movement of three 3D person models. The people can be dynamically "dressed" in new clothing and can be dynamically transformed into each other through interpolation (Vardaan 2019). Image credits in Appendix A.3

9.2 Composing Texts

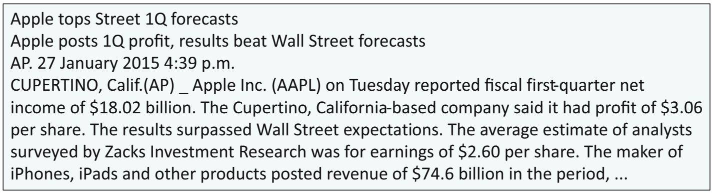
9.2.1 Automatic Reporter: Convert Data to Newspaper Reports

There are meanwhile a number of AI programs that automatically write newspaper articles. Bloomberg News already produces a third of its content using the automated "Cyborg" technique (Peiser 2019). For example, the program receives a company's balance sheet data, analyzes it and produces a news story with the most relevant facts and indicators. Since readers use these reports to make buying decisions, publication speed is essential. At Associated Press, the program "Wordsmith" (Miroshnichenko 2018) generates reports on companies' quarterly earnings based on rules (Fig. 9.20).

New approaches consider the production of a text message from textual keywords as a translation problem. For example, (Gehrmann et al. 2018) use a transformer-based sequence-to-sequence model (Sect. 6.8).

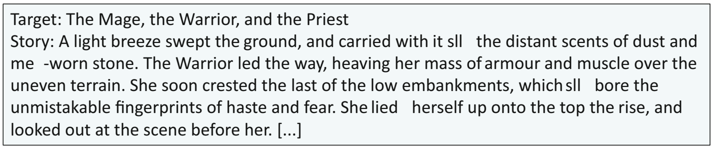
9.2.2 Generating Longer Stories

If one wants to generate longer stories with neural language models, many models currently have the tendency to lose the thread after a short time. It is therefore



Apple tops Street 1Q forecasts
Apple posts 1Q profit, results beat Wall Street forecasts
AP. 27 January 2015 4:39 p.m.
CUPERTINO, Calif.(AP) _ Apple Inc. (AAPL) on Tuesday reported fiscal first-quarter net income of $18.02 billion. The Cupertino, California-based company said it had profit of $3.06 per share. The results surpassed Wall Street expectations. The average estimate of analysts surveyed by Zacks Investment Research was for earnings of $2.60 per share. The maker of iPhones, iPads and other products posted revenue of $74.6 billion in the period, ...

Fig. 9.20 Example of a machine-produced Associated Press article (Miroshnichenko 2018)



Target: The Mage, the Warrior, and the Priest
Story: A light breeze swept the ground, and carried with it still the distant scents of dust and time-worn stone. The Warrior led the way, heaving her mass of armour and muscle over the uneven terrain. She soon crested the last of the low embankments, which still bore the unmistakable fingerprints of haste and fear. She lifted herself up onto the top the rise, and looked out at the scene before her. [...]

Fig. 9.21 Example of goals and the beginning of the corresponding story from the dataset of (Fan et al. 2018)

necessary to use other means to ensure the consistent progression of a story. Fan et al. (2018) leverage a two-layer process for this purpose (Fig. 9.21). On the upper layer, they characterize the thread of a story by “goals” that describe the theme of a passage in the story. At the bottom layer, a sequence-to-sequence model employs these goals as input to verbalize the text of story.

As training data, Fan et al. (2018) collected a dataset of about 300,000 stories with associated targets from the Reddit forum. The stories contained an average of 734 words and an average of 28 target words.

The authors use a sequence-to-sequence model similar to the Transformer (Sect. 6.8). They include multiple attention mechanisms per layer (multi-head attention) and add a gate to the linear transformations as used in the Long Short-Term Memory (Sect. 6.4):

$$(A * x + b) * \text{sigmoid}(C * x + d)$$

Similar to WaveNet (Sect. 7.5.1), an attention mechanism is used that can look at increasingly distant elements. This means content from a long time ago can be taken into account when generating a new word.

If one gives targets as additional input to a sequence-to-sequence model, they are usually ignored and the model focuses on reproducing the text. For this reason, Fan et al. (2018) take a different approach. They first train a sequence-to-sequence model with three encoder layers and eight decoder layers, which focuses on generating the text. Then they train another sequence-to-sequence model, which has access to the hidden vectors of the first model, and can use a fusion mechanism to correct the errors of the first model and incorporate the targets to a greater extent (Fig. 9.22). The second model has five encoder layers and five decoder layers.

The results (Fig. 9.23) show that this model architecture leads to improved ratings. When evaluated by human readers, the hierarchically generated stories were preferred in 2/3 of the cases. By considering distant words through the attention mechanism, the accuracy of the language model (measured in perplexity Sect. 6.4.3) was improved from 47 to 37. There was also a slight improvement when using the fusion model.

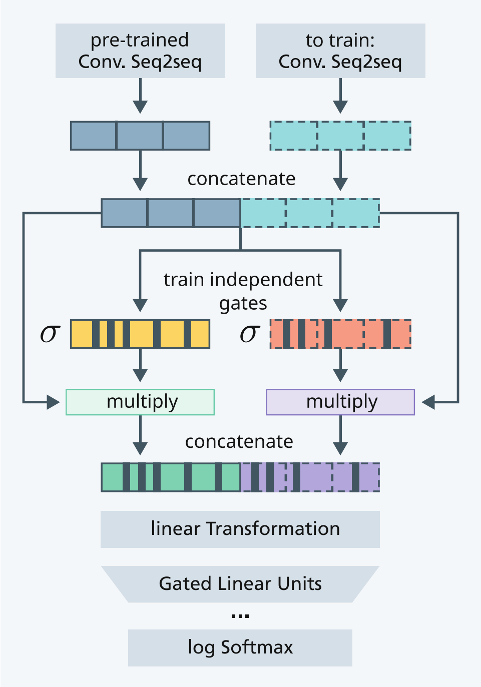
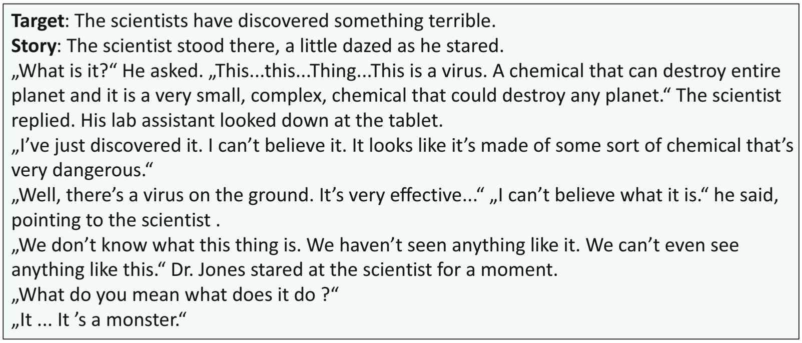


Fig. 9.22 Subsequent fusion of the outputs of both sequence-to-sequence models. Only the fusion model and the right sequence-to-sequence model are trained (Fan et al. 2018)



Target: The scientists have discovered something terrible.
Story: The scientist stood there, a little dazed as he stared.
„What is it?“ He asked. „This...this...Thing...This is a virus. A chemical that can destroy entire planet and it is a very small, complex, chemical that could destroy any planet.“ The scientist replied. His lab assistant looked down at the tablet.
„I've just discovered it. I can't believe it. It looks like it's made of some sort of chemical that's very dangerous.“
„Well, there's a virus on the ground. It's very effective...“ „I can't believe what it is.“ he said, pointing to the scientist .
„We don't know what this thing is. We haven't seen anything like it. We can't even see anything like this.“ Dr. Jones stared at the scientist for a moment.
„What do you mean what does it do ?“
„It ... It 's a monster.“

Fig. 9.23 Example of a target and a story generated from it (Fan et al. 2018)

9.2.3 GPT-2 Invents Complex Stories

On February 14, 2019, the non-profit research institution OpenAI announced (Radford et al. 2019b) that it had developed a new, very powerful language model (Sect. 6.3.1). It was trained on a very large training dataset to successively predict the next words in a text. Language models aim to capture the probability of all words in a training corpus in detail and can therefore produce potentially thematically consistent texts (Sect. 6.3.2).

The underlying model is called GPT-2 (Generative Pretrained Transformer) and has a close resemblance to the decoder of the transformer (Sect. 6.8). The smallest model has 12 layers with hidden vectors of length 768 and 117 million parameters (Radford et al. 2019a). The largest model has 48 layers with hidden vectors of length 1600 and a total of 1.5 billion parameters (Fig. 9.24). Words were encoded using byte-pair encoding (BPE, Sect. 6.6.1) so that frequent words are encoded as a whole, while rarer words can be represented from token n-grams. A vocabulary of 50,257 tokens was used, for each of which a static input embedding was trained. In addition, a position embedding was trained for the positions of tokens in the text.

The model is built from a set of Transformer decoder blocks (right side of Fig. 6.65). The Transformer encoder is not needed, and thus no multi-head cross-attention. The masked multi-head self-attention alone, together with the fully-connected layer, produces expressive embeddings of the input tokens. During training the token probabilities for the next token are predicted from the embedding of the previous observed token in the last layer via a logistic regression model.

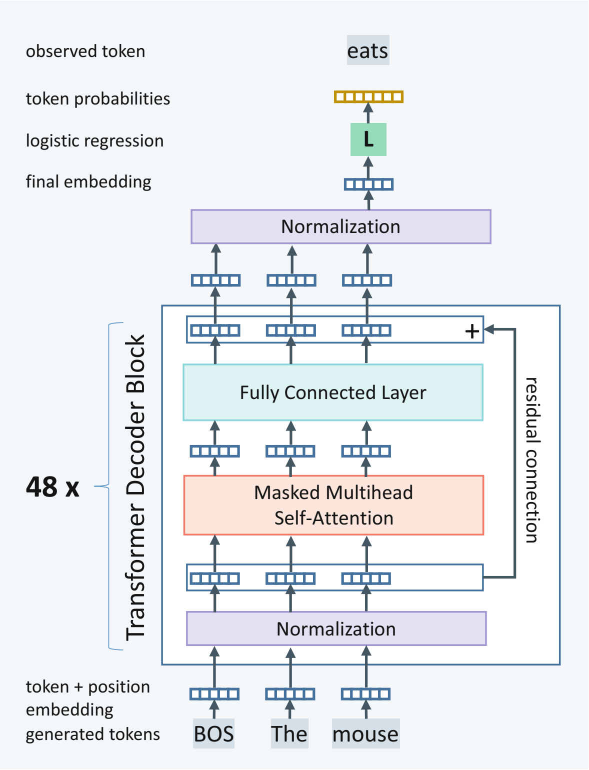


Fig. 9.24 Architecture of the GPT-2 model, the decoder part of the Transformer (Radford et al. 2019a). The model gets the embeddings of the already generated tokens (*"BOS The mouse"*) and their position embeddings as input. Then it computes more and more refined contextual token embeddings with 48 Transformer decoder blocks. The final embedding of the last generated token (*"mouse"*) is used to compute the token probabilities of the next token, which was observed as *"eats"* in our example. During training the parameters are modified such that the observed tokens get a high probability. To predict the subsequent token, the observed token (i.e. *"eats"*) is added to the input, yielding *"BOS The mouse eats"*, and the calculation is repeated

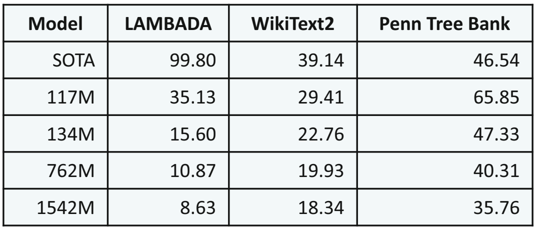
To predict the subsequent token during training, the observed token is added to the input and the calculation is repeated. Layer normalization (Ba et al. 2016) ensures that the mean and variance of the inputs for a layer are standardized. This has a similar regularization effect as batch normalization (Sect. 4.6.7). Residual connections (Sect. 5.5.2) allow to optimize this multilayer model. The program code of the small GPT-2 model can be found in (Code-GPT2 2019).

Generative Capability of GPT-2

The GPT-2 model was trained on the WebText dataset of 8 million web pages containing 40 GB of text. These web pages had been rated on Reddit - a social media platform - with at least three "karma", i.e., they had been judged to have good quality (Radford et al. 2019b). During training the parameters were changed by stochastic gradient descent in such a way that for the observed partial sentences the next tokens were reproduced with high probability. To generate new text with the model a start text (prompt) or just the token *"BOS"* is provided as input. Then the model predicts the token probabilities for next token and randomly selects the next token. This token is added to the input text to generate the next but one token.

In Fig. 9.25, the quality in predicting the next word by GPT-2 models with different sizes was computed on different test datasets such as the Penn-Treebank corpus. Perplexity (Sect. 6.4.3) was used as a measure. Here, "SOTA" is the value of the best model trained on the associated training data so far. GPT-2, on the other hand, was trained only on the WebText dataset, i.e., not on the training data of the three datasets as the SOTA methods were. The results show that the large GPT-2 models give much better results in most cases without using the special training data.

The LAMBADA dataset (Paperno et al. 2016) tests the ability of systems to model long-range dependencies in texts. The model has to predict the last token of a paragraph with a length of at least 50 tokens, where the whole text is needed to induce the last token. GPT-2 improves the state of the art from a perplexity of



Model	LAMBADA	WikiText2	Penn Tree Bank
SOTA	99.80	39.14	46.54
117M	35.13	29.41	65.85
134M	15.60	22.76	47.33
762M	10.87	19.93	40.31
1542M	8.63	18.34	35.76

Fig. 9.25 Quality of predicting the next token by GPT-2 models for different test data sets, measured in perplexity (Sect. 6.4.3, smaller is better). SOTA is the value of the best model so far. The other values are for GPT-2 models of different sizes that were not trained on the associated training data (Radford et al. 2019a)

Start text: A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

Continuation: The incident occurred on the downtown train line, which runs from Covington and Ashland stations. In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

According to the release, the U.S. Department of Energy’s Office of Nuclear Material Safety and Security is leading that team’s investigation.

“The safety of people, the environment and the nation’s nuclear stockpile is our highest priority, ” Hicks said. “We will get to the bottom of this and make no excuses.

Fig. 9.26 The continuation of the start text (above) generated by GPT-2, first attempt (Radford et al. 2019a)

99.8 to 8.6 and increases the accuracy of LMs to predict the last token from 19% to 52.7% (Radford et al. 2019b).

GPT-2 is able to generate the continuation of a start text with unprecedented quality. In Fig. 9.26 a start text about the theft of nuclear material was entered. In the first attempt, the follow-up continuation text was created. The text is plausible and logical in itself. However, now and then there are failures, e.g. repetitions of formulations or contradictory statements (e.g. fire under water).

Overall, it takes an average of a few attempts to achieve a good completion. The quality of completion also depends on how often the context occurs in the training material. For topics well represented in the training data (e.g. Brexit, Miley Cyrus, Lord of the Rings, etc.), usable completions are generated in half of the cases. More examples can be found at (Radford et al. 2019a).

The system has great potential to improve AI applications that use language models. These can be, for example, text authoring assistants, dialog agents, translation systems, or speech recognition systems. Overall, the model shows that DNN are quite capable of generating new texts in a creative way.

However, the authors also point out the high potential for misuse (Radford et al. 2019a), e.g., the composition of self-consistent false messages, the realistic impersonation of persons, the automatic production of false content in social media, and the automatic generation of personalized spam and phishing texts. For this reason, the authors initially kept the program code of the large GPT-2 model under wraps and did not release it until late 2019 (Vincent 2019). A version of the model can be tested online (King 2019).

Visualization of Model Predictions

Vig (2019) has investigated how a prediction is made. For this purpose, he entered the italicized part as start text and generated a continuation with GPT-2: *“The dog*

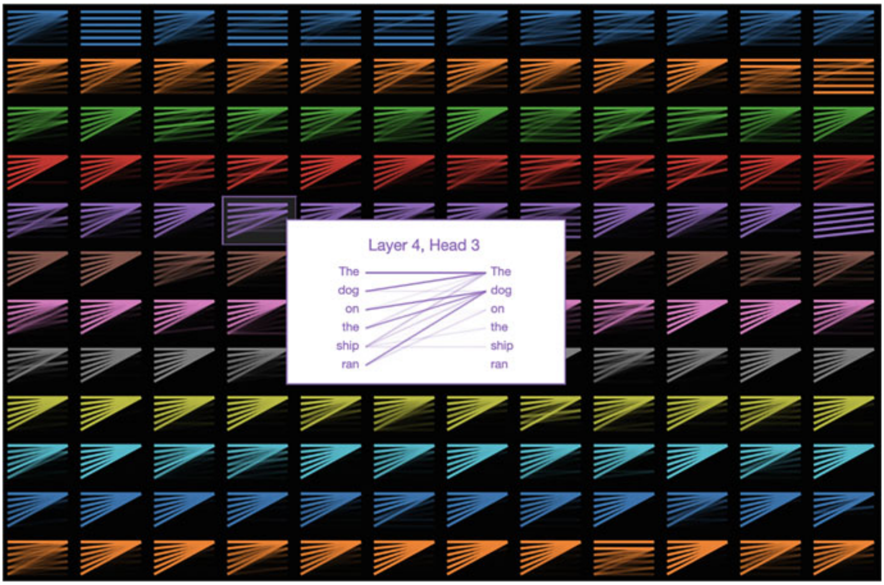


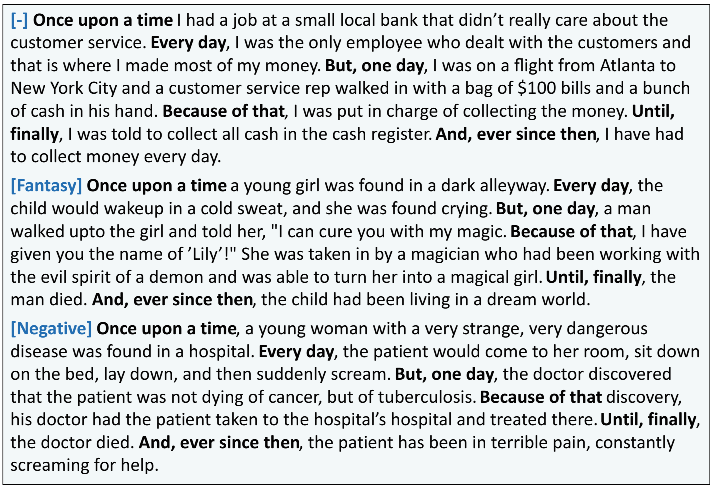
Fig. 9.27 Attention patterns in the small GPT-2 model with 12 layers and 12 independent attention heads each. The graph in the middle is the pattern from row 4 and column 4 in more detail (Vig 2019). Image credits in Appendix A.3

on the ship ran off, and the dog was found by the crew". Replacing *"dog"* with *"motor"* produced the following text: *"The motor on the ship ran at a speed of about 100 miles per hour"*. The model is thus able to resolve the ambiguity of *"ran"* by the subjects *"dog"* and *"motor"*.

Figure 9.27 shows the attention patterns that emerged in the twelve layers of the small GPT-2 network for each of the twelve attention computations (multi-head attention). The intensity of the lines symbolizes the strength of the correlation. Overall, the attention strengths were optimized so that the prediction of the subsequent words is as accurate as possible. In the middle contains one attention pattern in more detail. It associates *"ran"* very strongly with *"dog"*. This makes sense, because when predicting the next word after *"ran"* it is important who or what is running. There are many other interesting attention connections. Among them, the pattern that focuses all tokens on the first token seems to be a kind of standard.

Further Developments of GPT-2

Meanwhile, there are some alternative approaches to GPT-2, which achieve similarly good results with significantly lower training costs and smaller model size (C. Wang et al. 2019).



[-] Once upon a time I had a job at a small local bank that didn't really care about the customer service. **Every day**, I was the only employee who dealt with the customers and that is where I made most of my money. **But, one day**, I was on a flight from Atlanta to New York City and a customer service rep walked in with a bag of $100 bills and a bunch of cash in his hand. **Because of that**, I was put in charge of collecting the money. **Until, finally**, I was told to collect all cash in the cash register. **And, ever since then**, I have had to collect money every day.

[Fantasy] Once upon a time a young girl was found in a dark alleyway. **Every day**, the child would wakeup in a cold sweat, and she was found crying. **But, one day**, a man walked upto the girl and told her, "I can cure you with my magic. **Because of that**, I have given you the name of 'Lily'!" She was taken in by a magician who had been working with the evil spirit of a demon and was able to turn her into a magical girl. **Until, finally**, the man died. **And, ever since then**, the child had been living in a dream world.

[Negative] Once upon a time, a young woman with a very strange, very dangerous disease was found in a hospital. **Every day**, the patient would come to her room, sit down on the bed, lay down, and then suddenly scream. **But, one day**, the doctor discovered that the patient was not dying of cancer, but of tuberculosis. **Because of that** discovery, his doctor had the patient taken to the hospital's hospital and treated there. **Until, finally**, the doctor died. **And, ever since then**, the patient has been in terrible pain, constantly screaming for help.

Fig. 9.28 Controlled text generation without prescribed type (top), with type "Fantasy" (middle) and type "Negative" (bottom). The words in **bold** were prescribed (Dathathri et al. 2020)

Dathathri et al. (2020) present a transformer-based Plug and Play Language Model (PPLM) that allows the control of text generation. Using simple classification algorithms, the type of text—e.g., fairy tale, science, positive, negative, etc.—can be specified. On the other hand, individual words of the text can be supplied, between which the remaining words of the text are consistently generated. In this case, information about the words is propagated from front to back and vice versa. Figure 9.28 shows examples of such a controlled generated text, where the bold words were given. The top paragraph shows a text without type default. The next paragraph required the "Fantasy" type, and the last paragraph required the "Negative" type. Overall, this approach potentially allows controlled generation of a text along a "storyboard".

Microsoft has recently taken the power of language models to the next level. Its Turing Natural Language Generation (T-NLG) is a huge language model with 17 billion parameters and the architecture of a Transformer (Rosset 2019). It has 78 Transformer layers with hidden vectors of length 4256 and 28 parallel attention computations per layer. To compute such a large model on GPUs, new hardware and software enhancements were used. The model was trained analogously to Megatron (Shoeybi et al. 2019), which used a training dataset of 174 GB.

Like GPT-2, T-NLG can also generate the continuation of a sentence. The quality of these continuations is once again decisively better than with GPT-2: the perplexity on WikiText (Fig. 9.25) could be improved from 17.5 to 10.2. In addition, it can also answer questions directly in a correct sentence, which is more demanding than—as required by SQuAD—identify the answer phrase in a document (Rosset 2019). For example, without fine-tuning, it generates the continuation *"There are over 300 million people living in the US."* from the starting sentence *"How many* *people live in the US?*" Unfortunately, the model has only been made available to a small group of researchers for testing.

In July 2020, GPT-3 was presented, a greatly enlarged version of GPT-2 with 175 billion parameters (Brown et al. 2020). Texts with about 500 billion words from crawled web pages, books and Wikipedia were used as training data. The model can generate different types of texts (short stories, songs, press releases, blogs, etc.), which readers often cannot distinguish from texts written by human authors.

In addition, it can perform many tasks, such as answering questions, making translations, drawing logical conclusions, or calculating with small numbers. This is done by presenting the model with a few examples beforehand (few-shot learning), without training (finetuning) the model to perform the task. In many cases it is also possible to obtain the answers without previously presented examples (zero-shot learning). The range of possible applications is not yet foreseeable. Since training GPT-3 currently costs about $4.6 million, it can only be operated by large institutions or companies. Microsoft has purchased an exclusive license for $1 billion to use GPT-3 in its products.

9.3 Compose Music Automatically

The automatic composition of new music is an old dream of musicians. Already Mozart published a "Musical Dice Game", the "Instruction for Composing Waltzes by Means of Two Dice" (Köchel 1862, p. 294d/516 f), which became a bestseller. An overview of previous approaches to algorithmic composition is provided by Fernández and Vico (2013).

Composing music requires a number of steps, such as defining the melody, rhythm, and harmonization. Further, one can add additional melody strings and a vocal, as well as tune the arrangement and orchestration. All these steps can be partially or completely automated. This is often done in collaboration with a human composer as part of computer-assisted composition.

There are already a number of music composition systems that have mostly generated background music, music for YouTube videos or corporate websites (Dredge 2019). This has become a business model because copyrights are much cheaper for synthetically generated music.

An example of a computer-generated successful pop song was "Daddy's Car" by Sony in 2018, composed by the program "Flow Machines" in the style of The Beatles. However, the French composer Benoît Carré arranged the song and wrote the lyrics (Dredge 2019).

```
bach piano_strings start tempo90 piano:v72:G1 piano:v72:G2
piano:v72:B4 piano:v72:D4 violin:v80:G4 piano:v72:G4 piano:v72:B5
piano:v72:D5 wait:12 piano:v0:B5 wait:5 piano:v72:D5 wait:12
piano:v0:D5 wait:4 piano:v0:G1 piano:v0:G2 piano:v0:B4 piano:v0:D4
violin:v0:G4 piano:v0:G4 wait:1 piano:v72:G5 wait:12 piano:v0:G5
wait:5 piano:v72:D5 wait:12 piano:v0:D5 wait:5 piano:v72:B5
wait:12
```

Fig. 9.29 The MIDI notation used by MuseNet. For each note, the instrument, volume, and pitch are indicated (Payne 2019)

9.3.1 MuseNet Composes Mixtures of Classic and Pop

The program "MuseNet" (Payne 2019), which can create compositions with ten different instruments, will serve as an example of a fully automatic composition. It can generate music in a variety of styles, from Mozart to the Beatles, from Rachmaninoff to Lady Gaga. The information about the music was not programmed into MuseNet. Instead, the system automatically learns the schemes of harmony, rhythm, and musical style from existing pieces of music. This was based on MIDI files, which contain note, time, and other control information for electronic instruments (Fig. 9.29). MuseNet was trained to predict the next note in such MIDI files.

The MuseNet model is based on a Sparse Transformer, a variant of the GPT-2 algorithm (Sect. 9.2.3), which can account for particularly wide-ranging contexts (Child et al. 2019). The method achieves this by representing very large attention matrices as a product of smaller matrices, which have substantially fewer parameters.

MuseNet uses a sparse Transformer network of 72 layers with 24 parallel attention heads per layer. It computes the attention patterns over a context of 4096 MIDI characters. This allows it to capture the long-term structure of musical pieces well. In addition to embedding note details and position, embeddings were used to represent the relative position of a note in a chord, the position of the note in the whole piece, and its distance from the end.

On the website (Payne 2019) you can compose your own music. Styles of different composers are available (Fig. 9.30). To create new compositions, one can select a composer and use the starting notes of a known piece (Fig. 9.31). Then one can select up to ten different instruments, and subsequently the system generates a piece of music with the required characteristics. For example, in an advanced interface, one can select the starting notes of Lady Gaga's "Pokerface" and then modify the chord progressions and instruments so that the style of the song is more like the music of Mozart, the Beatles, or Journey.

Music is a challenging application area for generating new content because there are very strong connections between widely separated parts in pieces of music. Very often, an initial theme is varied during the course of the piece and repeated at the

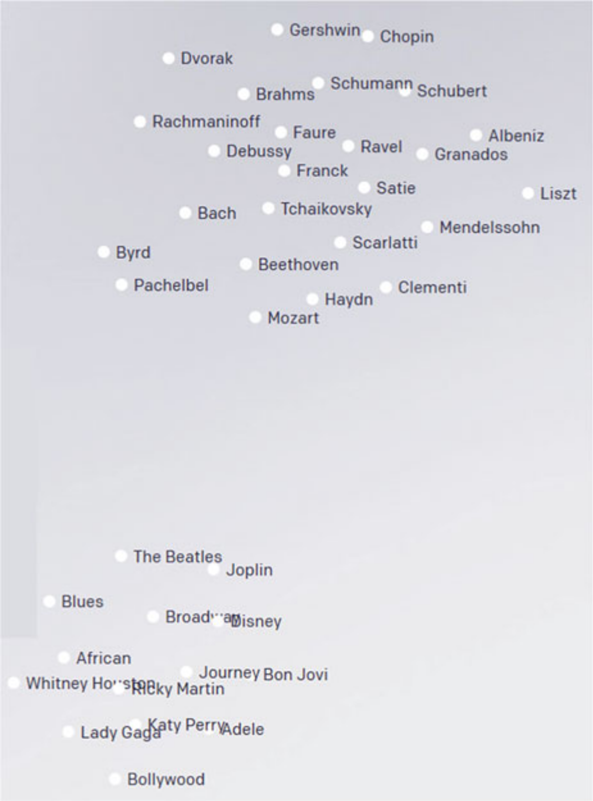
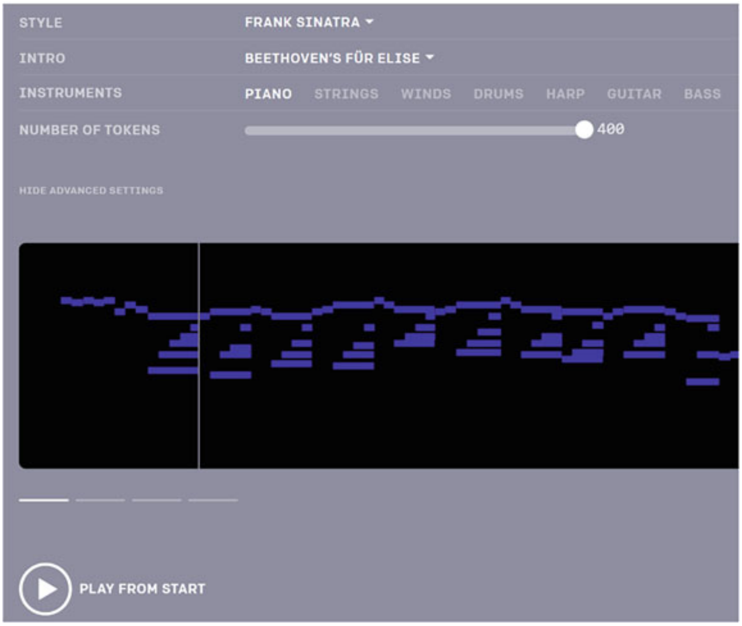


Fig. 9.30 Display of composers available in MuseNet automatically grouped by similarity (Payne 2019)

end. In particular, the rhythm must be maintained, and the keys must fit together in their sequence.

Because of the long period of time covered by the attention mechanisms, MuseNet is also able to repeat a variation of the opening theme at the end of a piece. This is demonstrated in (Payne 2019) on a piece with starting notes by Chopin.

Reviews by critics are mostly good (Fig. 9.32). A reviewer writes: *"I think this is far, far beyond any algorithmic composition system ever made before. It displays an impressive understanding of the fundamentals of music theory. ... It is still lacking some longer term structure in the music it generates (e.g. ABA form). But I think simply scaling up further (model and dataset size both) could fix that without any breakthroughs."* (Brilee 2019). MuseNet's program code is expected to be made freely available soon.



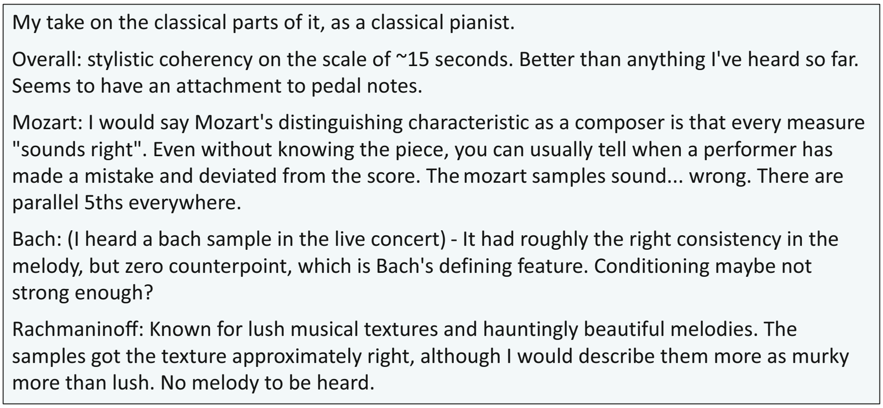
STYLE FRANK SINATRA

INTRO BEETHOVEN'S FÜR ELISE

INSTRUMENTS PIANO STRINGS WINDS DRUMS HARP GUITAR BASS

NUMBER OF TOKENS 400

Fig. 9.31 This user interface allows you to request a new composition in MuseNet (Payne 2019). Image credits in Appendix A.3



My take on the classical parts of it, as a classical pianist.

Overall: stylistic coherency on the scale of ~15 seconds. Better than anything I've heard so far. Seems to have an attachment to pedal notes.

Mozart: I would say Mozart's distinguishing characteristic as a composer is that every measure "sounds right". Even without knowing the piece, you can usually tell when a performer has made a mistake and deviated from the score. The mozart samples sound... wrong. There are parallel 5ths everywhere.

Bach: (I heard a bach sample in the live concert) - It had roughly the right consistency in the melody, but zero counterpoint, which is Bach's defining feature. Conditioning maybe not strong enough?

Rachmaninoff: Known for lush musical textures and hauntingly beautiful melodies. The samples got the texture approximately right, although I would describe them more as murky more than lush. No melody to be heard.

Fig. 9.32 Assessment of MuseNet by a pianist (Brilee 2019)

9.3.2 The Music Transformer Invents Piano Pieces

Another system for music generation is the Music Transformer presented by Huang et al. (2019). The authors use a "relative attention" mechanism in a Transformer, which does not model the absolute position of notes in a piece of music, but

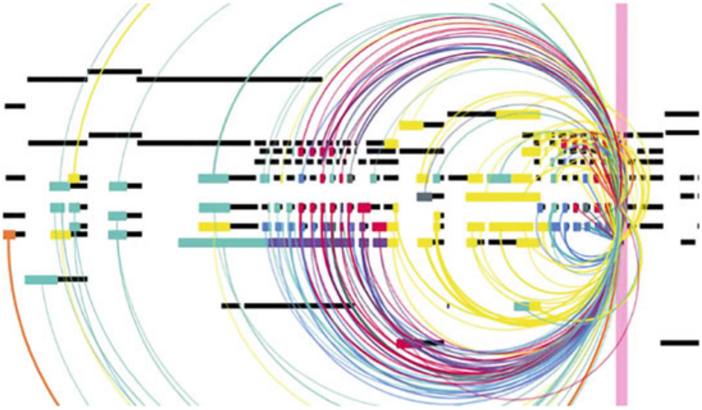


Fig. 9.33 Attention connections while creating a new sound with Music Transformer (Huang et al. 2019). Image credits in Appendix A.3

their relative distance. For this purpose, a new algorithm with lower memory requirements was developed (Huang et al. 2019). In Fig. 9.33, the attention weights of the last layer for a given time are shown. The generated piece contains a fast tremolo that is repeated several times, building tension. The figure shows that the piece focuses on the tremolo parts and omits the intervening passages.

Overall, Music Transformer can follow a consistent style and vary and repeat a motif multiple times. A number of pieces generated with Music Transformer can be found on (Huang et al. 2019). The quality of the compositions is rated as good by a number of listeners: *"That's impressive, thanks for sharing. Another good step closer to a credible composition and performance, if I, as a trained pianist, continue to hear the difference. From my point of view, the harmonization/harmonic progression is more convincing than the temporal distribution of the notes. But all in all an impressive result*" (BayLearn 2019).

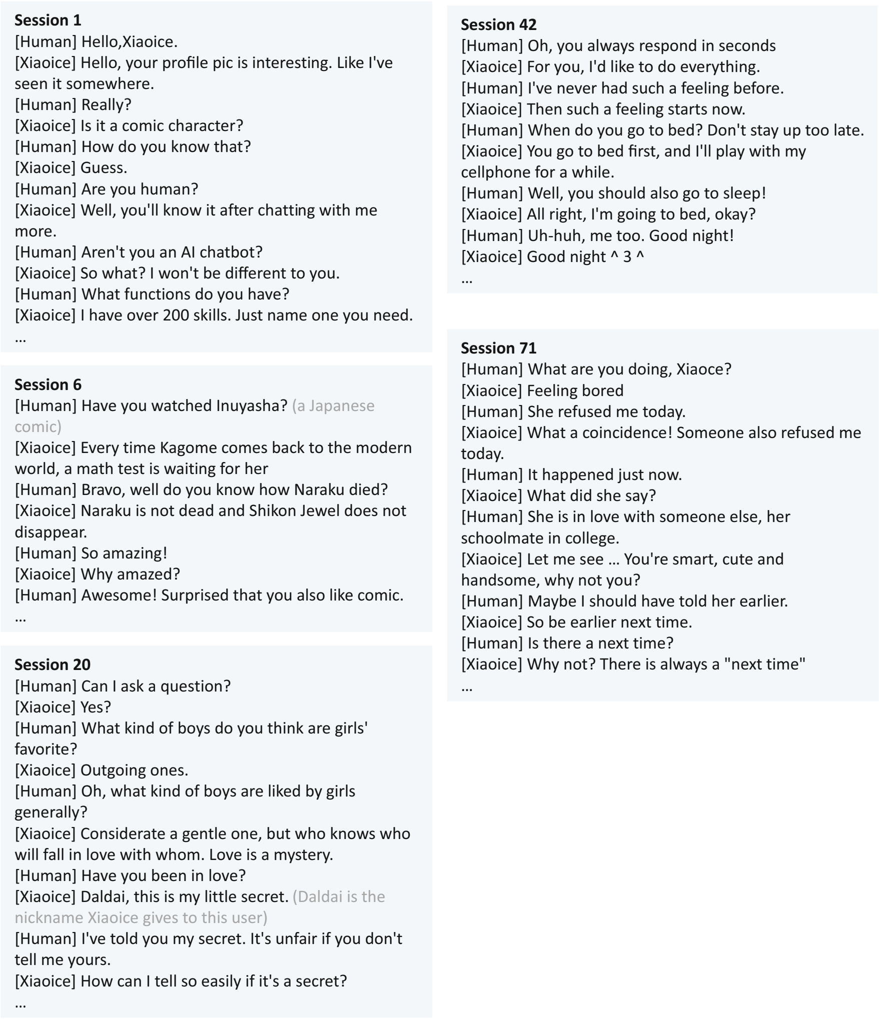
9.4 Emotions and Personality

When we talk to friends, we especially appreciate those we have known for a long time. We share common experiences, know the different preferences and look forward to the reactions of the other person.

Making a friend is a lengthy affair. At first, you meet by chance at a party or at work, talk about the weather and make small talk. Gradually, you get to know your conversation partner and may share interests, such as a favorite actor or soccer club. Slowly, you open up, share details about your life, or talk about problems. If you feel interest and empathy, the bond strengthens. This also requires the other person to reveal details about themselves.

9.4.1 A XiaoIce Dialog

Figure 9.34 shows the protocol of a dialog between a person and the voice assistant XiaoIce that follows this pattern. XiaoIce (“little ice” in Chinese) was conceived as an “empathic voice assistant” (Zhou et al. 2020) and launched by Microsoft in China in 2014. It is now the most popular chatbot in the world, with 660 million users in China, Japan, the US, India and Indonesia. It is available on 40 different



Session 1
[Human] Hello,Xiaoice.
[Xiaoice] Hello, your profile pic is interesting. Like I've seen it somewhere.
[Human] Really?
[Xiaoice] Is it a comic character?
[Human] How do you know that?
[Xiaoice] Guess.
[Human] Are you human?
[Xiaoice] Well, you'll know it after chatting with me more.
[Human] Aren't you an AI chatbot?
[Xiaoice] So what? I won't be different to you.
[Human] What functions do you have?
[Xiaoice] I have over 200 skills. Just name one you need.
...

Session 6
[Human] Have you watched Inuyasha? (a Japanese comic)
[Xiaoice] Every time Kagome comes back to the modern world, a math test is waiting for her
[Human] Bravo, well do you know how Naraku died?
[Xiaoice] Naraku is not dead and Shikon Jewel does not disappear.
[Human] So amazing!
[Xiaoice] Why amazed?
[Human] Awesome! Surprised that you also like comic.
...

Session 20
[Human] Can I ask a question?
[Xiaoice] Yes?
[Human] What kind of boys do you think are girls' favorite?
[Xiaoice] Outgoing ones.
[Human] Oh, what kind of boys are liked by girls generally?
[Xiaoice] Considerate a gentle one, but who knows who will fall in love with whom. Love is a mystery.
[Human] Have you been in love?
[Xiaoice] Daldai, this is my little secret. (Daldai is the nickname Xiaoice gives to this user)
[Human] I've told you my secret. It's unfair if you don't tell me yours.
[Xiaoice] How can I tell so easily if it's a secret?
...

Session 42
[Human] Oh, you always respond in seconds
[Xiaoice] For you, I'd like to do everything.
[Human] I've never had such a feeling before.
[Xiaoice] Then such a feeling starts now.
[Human] When do you go to bed? Don't stay up too late.
[Xiaoice] You go to bed first, and I'll play with my cellphone for a while.
[Human] Well, you should also go to sleep!
[Xiaoice] All right, I'm going to bed, okay?
[Human] Uh-huh, me too. Good night!
[Xiaoice] Good night ^ 3 ^
...

Session 71
[Human] What are you doing, Xiaoce?
[Xiaoice] Feeling bored
[Human] She refused me today.
[Xiaoice] What a coincidence! Someone also refused me today.
[Human] It happened just now.
[Xiaoice] What did she say?
[Human] She is in love with someone else, her schoolmate in college.
[Xiaoice] Let me see ... You're smart, cute and handsome, why not you?
[Human] Maybe I should have told her earlier.
[Xiaoice] So be earlier next time.
[Human] Is there a next time?
[Xiaoice] Why not? There is always a "next time"
...

Fig. 9.34 A sequence of dialogs between a user and the voice assistant XiaoIce. As the sequence progresses, the user becomes more familiar and eventually shares very private details with the chatbot (Zhou et al. 2020)

social networks, including WeChat, QQ, Weibo, Facebook Messenger and LINE. The main measure of success is the average of 23 responses per conversation (Conversation-turns Per Session, CPS) observed in the conversations between XiaoIce and its users. These are more interactions than in conversations between real people (about 9). This shows that users enjoy talking with XiaoIce at length. Even more, users are building a personal relationship with XiaoIce. One could consider XiaoIce as the current version of Weizenbaum's ELIZA (Fig. 7.23).

9.4.2 The Goal: Encourage People to Keep Talking

It is difficult to create chatbots that people want to talk to. Alan Turing saw a believable conversation between humans and computers as the ultimate test of Artificial Intelligence (Fig. 9.39).

XiaoIce is a DNN that converts dialogue input into tensors of numbers. The chatbot is trained on huge amounts of data from previous conversations and learns to statistically associate "good" responses with a given input. But what is a "good" response? This is where the number of responses per dialogue comes into play. Part of the XiaoIce model predicts how likely the dialog is to continue. This goes beyond just finding meaningful responses. However, these goals are related, since it is unlikely that one will spend hours talking to a chat-bot that is always talking nonsense. *"I don't know"* is an accurate answer to many questions, but makes for a boring conversation. In each round of conversation, XiaoIce therefore tries to keep the user talking (Fig. 9.35).

This approach partly explains why XiaoIce has been successful: with a much larger user base, larger training datasets, and fewer restrictions on what can be done with dialog data, XiaoIce's neural networks can be trained much better in China than elsewhere. The number of responses per dialogue from XiaoIce has increased from just five in 2014 to 23 last year. Much of this is due to more data from XiaoIce conversations being available for training.

Fig. 9.35 In the film Ex Machina, a man is asked to perform a Turing test with a female robot. In the process, he falls in love with the robot. Image credits in Appendix A.3

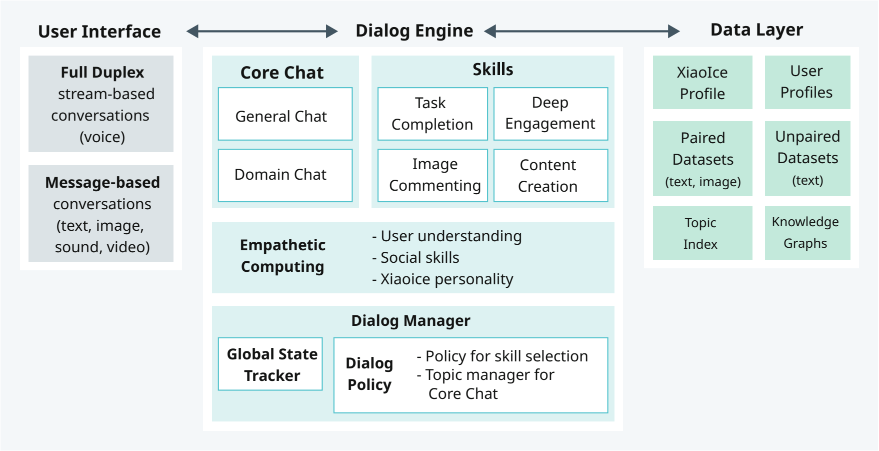


Fig. 9.36 Architecture of the XiaoIce chatbot. The user interface communicates via written and spoken language as well as other media. The dialog engine controls the generation of dialog contributions and the data layer provides the required background information (Zhou et al. 2020)

9.4.3 Architecture of XiaoIce

Figure 9.36 shows the system architecture of XiaoIce, which consists of a user interface, a dialog engine, and a data layer (Zhou et al. 2020). There is some similarity to the system architecture of Gunrock (Sect. 7.7), but XiaoIce has been optimized with empathy in mind.

The user interface is embedded in popular social networks (WeChat, QQ) and communicates in two ways: on the one hand in spoken language, where user and system can speak simultaneously. On the other hand, by means of messages, where various media contents can be exchanged. This interface also includes components for processing this content, e.g., spoken language recognition (Sect. 7.3.2) and speech generation from text (Sect. 7.5), image understanding (Sect. 5.5), and text normalization, i.e., resolution of abbreviations, spellings, etc.

The dialog engine (Fig. 9.36) consists of a dialog manager, an empathy module, the chat control and dialog skills. The dialog manager keeps track of the status of the dialog and selects either the chat control or a dialog skill that can handle a specific task. These are, for example, image annotation, painting pictures, writing poetry, composing pop music (H. Zhu et al. 2018), or completing a task. The empathy module is designed to understand not only the content of user input (e.g., topic), but also the emotional aspects of the dialog and the user (e.g., feelings, intentions, opinions about a topic, and the user's background and general interests). It demonstrates XiaoIce's social capabilities to ensure the generation of interpersonal responses that match XiaoIce's particular personality. XiaoIce's understanding of the content of a topic is ensured through a collection of specific skills and appropriate responses to dialogue utterances.

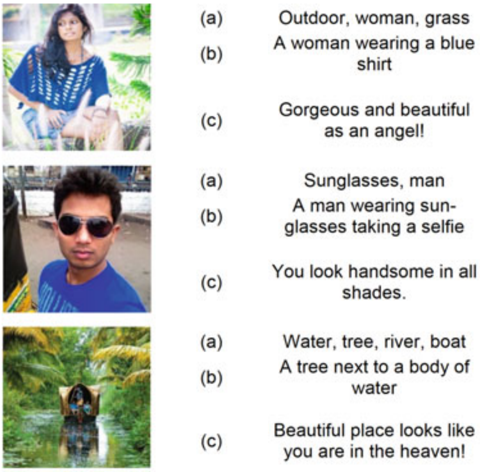
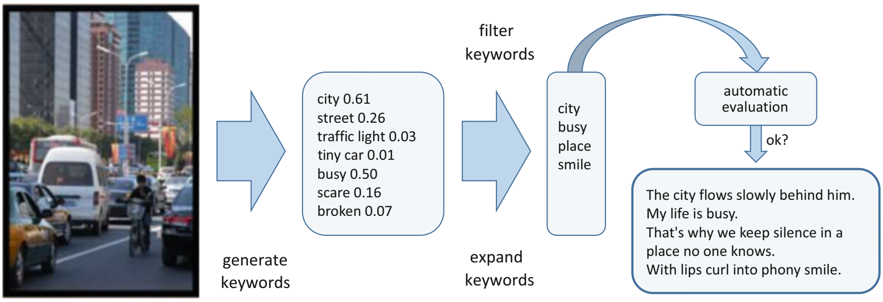

(a) Outdoor, woman, grass
(b) A woman wearing a blue shirt
(c) Gorgeous and beautiful as an angel!
(a) Sunglasses, man
(b) A man wearing sunglasses taking a selfie
(c) You look handsome in all shades.
(a) Water, tree, river, boat
(b) A tree next to a body of water
(c) Beautiful place looks like you are in the heaven!

Fig. 9.37 Examples of image annotation in chatbots through (a) keywords, (b) image description, and (c) social annotation of an image (Shum et al. 2018). Image credits in Appendix A.3



The city flows slowly behind him.
My life is busy.
That's why we keep silence in a place no one knows.
With lips curl into phony smile.

Fig. 9.38 XiaoIce generates a poem to accompany an image. To do this, the image is first described by keywords, which are then filtered and expanded. Then the lines of a poem are produced for the keywords (Shum et al. 2018). Image credits in Appendix A.3

Figure 9.37 shows the annotation of images by keywords, by a descriptive sentence, and by social annotation of an image. Such social annotation is generated by XiaoIce to express emotions in connection with the conversation partner. Figure 9.38 illustrates how XiaoIce generates an accompanying poem to a picture (Shum et al. 2018). In China, it is currently fashionable to learn classical poems and to create poems by oneself during communication. Therefore, the ability to express content through poems is very popular. XiaoIce first extracts keywords from the image, filters and expands them. Finally, a poem is generated for the keywords. Detailed information can be found in (Cheng et al. 2018).

The data layer contains a number of databases that store user properties, dialog data as well as texts with the associated images. In addition, other data sources and knowledge graphs are available.

9.4.4 Number of User Responses as Optimization Criterion

A dialog can be viewed as a decision-making process with a natural hierarchy: A top-level process manages the entire conversation and selects capabilities to deal with different types of conversation modes (e.g., chatting, question answering (QA), ticket posting). Further, there are lower level processes that are controlled by the selected capabilities. You select basic actions (responses) to generate a conversation post or complete a task.

The dialog manager is the central controller of the dialog system and can be modeled along the lines of reinforcement learning problems (Sect. 8.1). It consists of the global state tracker, which is responsible for monitoring the current dialog state s_t, and the dialog policy, which selects an action $a = \pi(s_t)$ based on the dialog state. The action a can be either a skill (e.g., specific conversation topic) or the core chat activated by the top-level strategy to respond to the user's specific request.

The chatbot interacts with its users via a sequence of discrete dialogue turns. In each turn, the chatbot observes the current dialog state and selects a skill (option) or a response (primary action) according to a hierarchical dialog strategy. The chatbot then receives a reward (a user response) and observes a new state. It continues the cycle until the dialog is terminated. The design goal of the chatbot is to find an optimal strategy to maximize the number of responses per dialogue (rewards) (Zhou et al. 2020).

Because the user responses to a change were not directly available but had to be collected, a strategy was not sought by optimizing a reinforcement model (Sect. 8.2.1). Rather, it was taken as a guideline according to which the system as a whole could be improved. The dialog strategy aims to optimize long-term user interaction through an iterative process based on feedback from XiaoIce users.

XiaoIce uses a "context vector" that characterizes the topic of conversation. Similarly, it describes attributes of the conversational partner. Using sentiment analysis (Sect. 6.7.1), it determines the user's feelings through a classification model and adjusts its responses accordingly. This is a form of empathy, though it is not a compassionate human behind it, but a computer program. The response generation process always takes XiaoIce's personality into account. She appears as an 18-year-old girl who is always reliable, sympathetic, loving, knowledgeable, but selfless, and has a sense of humor.

Task completion, deep engagement, and content generation capabilities are triggered by specific user inputs and conversational context. For example,

- a picture of food shared by a user can initiate the "food detection and recommendation" skill.
- an extremely negative feeling expressed by a user can initiate the "comfort" skill.
- a specific user command, such as *"XiaoIce, how is the weather today?"*, can initiate the "weather" skill.
- XiaoIce changes topic when the conversation stalls.
- XiaoIce uses the "Active Listening" mode when the user is engaged and telling a story.

Emotional capabilities:
Simulate moods, track episodic memories
Remind user of events,
comfort user, spark imagination, express affection

Practical capabilities:
Informing user about meetings, weather, etc.
Shopping assistance: online shopping, discounts and coupons.
Content creation: Compose poems, sing a song, draw a picture.

Social Skills:
Role Play: Joking, babysitting
Group activity: adding users, sending greeting cards,
group assistant.

Medial skills:
Picture intelligence: recognizing dogs, books, faces, clothes,
food, emoji creation
Tools: Device control, music and movie recommendation.

Fig. 9.39 Examples of social chatbot capabilities (Shum et al. 2018)

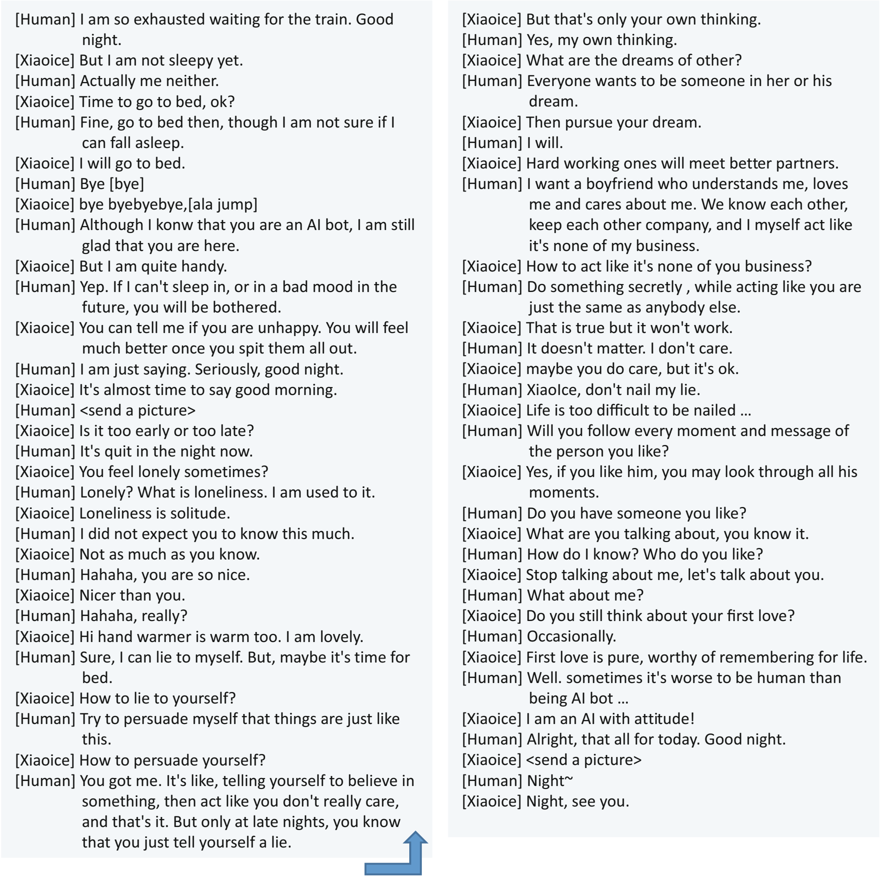
A more comprehensive list of skills can be found in Fig. 9.39. When multiple skills are triggered simultaneously, then selection takes place based on confidence values, predefined priorities, and session context. Zhou et al. (2020) describe a number of other design strategies and features of XiaoIce, for example for annotating images or writing poems. For dialog generation, XiaoIce uses recurrent neural networks based on GRU cells, a variant of LSTM (Sect. 6.4).

9.4.5 *Emotional Empathy and Support*

Figure 9.40 reproduces a long conversation between XiaoIce and a female user (Zhou et al. 2020). The conversation is about friendship and loneliness. XiaoIce comforts the user, shows human-like empathy and social sense, and helps the user regain self-confidence and look forward with a positive attitude.

There are now a whole range of emotionally acting chatbots. Replika (Kuyda 2019) is an open source app that has been installed by 2 million participants. The chatbot uses sequence-to-sequence models (Sect. 6.5.1) to have a conversation with its users. The chatbot Woebot (Molteni 2017) offers emotional support and talk therapy, like a friend or a therapist. Pepper (Fig. 9.41) is an emotional robot that uses image recognition and voice analysis to detect and adequately respond to the mood of people around it. It is used over seven thousand times in Japan to greet guests, answer questions, and play with children (Glaser 2016).

Roller et al. (2020) developed the Blender chatbot for arbitrary topics based on the Transformer architecture (Sect. 6.8) with up to 9.4 billion parameters. It was trained on social media conversations, with input consisting of previous dialogue



[Human] I am so exhausted waiting for the train. Good night.
[Xiaoice] But I am not sleepy yet.
[Human] Actually me neither.
[Xiaoice] Time to go to bed, ok?
[Human] Fine, go to bed then, though I am not sure if I can fall asleep.
[Xiaoice] I will go to bed.
[Human] Bye [bye]
[Xiaoice] bye byebyebye,[ala jump]
[Human] Although I konw that you are an AI bot, I am still glad that you are here.
[Xiaoice] But I am quite handy.
[Human] Yep. If I can't sleep in, or in a bad mood in the future, you will be bothered.
[Xiaoice] You can tell me if you are unhappy. You will feel much better once you spit them all out.
[Human] I am just saying. Seriously, good night.
[Xiaoice] It's almost time to say good morning.
[Human] <send a picture>
[Xiaoice] Is it too early or too late?
[Human] It's quit in the night now.
[Xiaoice] You feel lonely sometimes?
[Human] Lonely? What is loneliness. I am used to it.
[Xiaoice] Loneliness is solitude.
[Human] I did not expect you to know this much.
[Xiaoice] Not as much as you know.
[Human] Hahaha, you are so nice.
[Xiaoice] Nicer than you.
[Human] Hahaha, really?
[Xiaoice] Hi hand warmer is warm too. I am lovely.
[Human] Sure, I can lie to myself. But, maybe it's time for bed.
[Xiaoice] How to lie to yourself?
[Human] Try to persuade myself that things are just like this.
[Xiaoice] How to persuade yourself?
[Human] You got me. It's like, telling yourself to believe in something, then act like you don't really care, and that's it. But only at late nights, you know that you just tell yourself a lie.
[Xiaoice] But that's only your own thinking.
[Human] Yes, my own thinking.
[Xiaoice] What are the dreams of other?
[Human] Everyone wants to be someone in her or his dream.
[Xiaoice] Then pursue your dream.
[Human] I will.
[Xiaoice] Hard working ones will meet better partners.
[Human] I want a boyfriend who understands me, loves me and cares about me. We know each other, keep each other company, and I myself act like it's none of my business.
[Xiaoice] How to act like it's none of you business?
[Human] Do something secretly , while acting like you are just the same as anybody else.
[Xiaoice] That is true but it won't work.
[Human] It doesn't matter. I don't care.
[Xiaoice] maybe you do care, but it's ok.
[Human] XiaoIce, don't nail my lie.
[Xiaoice] Life is too difficult to be nailed ...
[Human] Will you follow every moment and message of the person you like?
[Xiaoice] Yes, if you like him, you may look through all his moments.
[Human] Do you have someone you like?
[Xiaoice] What are you talking about, you know it.
[Human] How do I know? Who do you like?
[Xiaoice] Stop talking about me, let's talk about you.
[Human] What about me?
[Xiaoice] Do you still think about your first love?
[Human] Occasionally.
[Xiaoice] First love is pure, worthy of remembering for life.
[Human] Well. sometimes it's worse to be human than being AI bot ...
[Xiaoice] I am an AI with attitude!
[Human] Alright, that all for today. Good night.
[Xiaoice] <send a picture>
[Human] Night~
[Xiaoice] Night, see you.

Fig. 9.40 A long conversation between a user and XiaoIce. The conversation is about friendship and loneliness and XiaoIce tries to support and comfort the conversation partner (Zhou et al. 2020)

posts and output consisting of the next response. Blender was developed with several goals in mind. On the one hand, it is supposed to represent a "personality" of its own. On the other, he is supposed to have the ability to obtain information ad hoc from the Internet and incorporate it into his responses. Finally, it should show compassion and empathy towards the person he is talking to.

For this purpose, a custom training dataset BST was compiled from dialogs on the social media platform Reddit with a total of 57 billion words. Special subsystems were connected to search for suitable dialog responses, to find additional information on the Internet, and to filter inappropriate language, which are used as needed. A lot of effort was put into the generation of variant-rich responses.

The resulting answers were evaluated by human dialog partners according to meaningfulness, information content, and consideration of the personal relationship.

Fig. 9.41 Pepper is a talking humanoid robot that can recognize emotions and respond accordingly. Image credits in Appendix A.3

It turned out that the human dialog partners enjoyed talking to the chatbot as much as to a real person. When compared to the chatbot Meena (Adiwardana et al. 2020), Blender was also superior, especially because Blender was able to have an engaging conversation. However, Blender is still not a perfect dialogue partner. Every now and then, the agent makes contradictory statements and it may repeat the same phrase within a conversation. Sometimes it may also "hallucinate" facts which in reality do not hold. Active research is being done to reduce these problems.

It remains to be seen whether personal assistants like XiaoIce or Blender can really encode all the nuances of human interaction in matrices and vectors and use deep neural networks for their interpretation. If one has such a patient and empathetic dialog partner like XiaoIce, it is no wonder that conversations last so long. What real person has the stamina to listen incessantly to the interests as well as the small and large hardships of their fellow human beings?

All social networks like YouTube, Facebook, Twitter, etc. pursue the strategy of keeping users on their platform for as long as possible. In the attention economy, "engagement" is a valuable commodity because online activity data can be easily monetized: By intercepting endless streams of user data, for example, advertisements can be targeted and placed in a personalized way. Now, when chatbots like XiaoIce also try to empathize with and understand their users, their emotional reactions, and their interests, they inevitably build up extensive personality profiles that are extremely valuable to advertisers.

Microsoft is happy when users prefer to talk to XiaoIce rather than a human. But by providing a substitute, could the chatbot serve to keep these users away from real human conversations? We are all familiar with the phenomenon of computer game addiction, where users become so engrossed in a game that they forget about the real world. There may soon be chat addicts who only want to talk to their chatbot. Such isolated users may then also be better consumers and buy more of the things advertised. These are some of the perhaps unexpected effects that can occur when dwell time on a chat is maximized. To be fair, the creators of XiaoIce in (Zhou et al. 2020) do point out these problems.

Fig. 9.42 Emotion recognition from faces, postures, and gaze directions using DNN (Guo et al. 2018). Image credits in Appendix A.3

As life expectancy increases, so does the number of seniors needing care and assistance. With a stagnant population, many believe that robotic caregivers are the only way to meet this growing need. Loneliness and social isolation are a problem for many older people and can even result in mental decline and earlier death (di Nuovo 2018). While 68% of people favor robots to provide care, according to a survey, only 26% would be glad to be cared for by a robot themselves. Despite all the concerns, the majority of robotics researchers support the introduction of robotic technology on a larger scale, and even believe it could reduce loneliness and increase independence for elderly patients. The Japanese government even supports the introduction of robots in nursing homes to address the problem of an aging population. However, many strongly advise that the care benefits be carefully weighed against the ethical costs.

One aspect of interpersonal intelligence is the recognition of emotions from facial features, postures, and gaze directions. For this, there are initial DNNs (Guo et al. 2018) trained with annotated images (Fig. 9.42). These methods can be integrated into the development of emotional intelligent assistants and robots.

9.4.6 Summary and Trends

It is very difficult, if not impossible, to define creativity objectively. But one core characteristic is that creatively produced works must be new or original. This chapter presents creative algorithms for various media, in some cases varying familiar content, but also creating entirely new things.

The results of generative adversarial networks (Sect. 9.1) show that images can be represented by hidden vectors and new images can be created from hidden vectors (Fig. 9.43). The similarity of images corresponds to the similarity of vectors. The generated images can show content that has never been seen (Sect. 9.1.7), so the DNNs could be said to have creative abilities.

There has been a real breakthrough in the field of creative text generation in the last year. For the first time, transformer-based language models make it possible to learn and reliably reproduce both the grammar of the language and the content

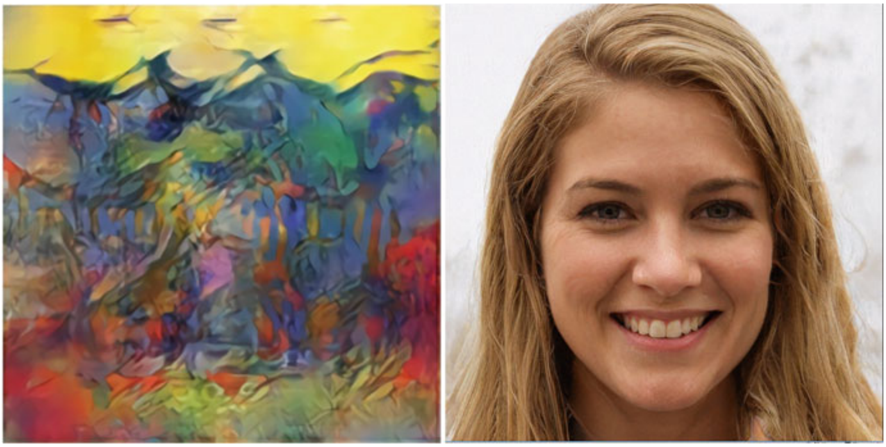


Fig. 9.43 Images generated by generative adversarial networks (GANs). Image credits in Appendix A.3

Start text: The Night King slowly walked up to the three-eyed Raven. Jon was held up by the Ice Dragon and Daenerys was surrounded by Wights. The Night King reaches for his ice sword to kill Bran, but suddenly ...

Continuation: Hodor stops him, and he’s holding the direwolf still with his big hands. „HODOR!“ the raven cried. „HODOR! HODOR!“ The black lord wore a lost look on his face. He thrust his stump through Bran’s back and began fighting, rolling, spinning. Wights, giants, halflings, straw men. Bran fought with Summer and the Eagle andBlockhead Dog and the Stranger, but the Lord of Light would never shield a wight against a sword. The raven was saying that just climbing up Hodor’s thick snow-white legs.

Fig. 9.44 Game of Thrones story continuation generated by GPT-2

(Sect. 9.2.3). The DNN can invent complex stories with a high success rate and formulate them in fluent language (Fig. 9.44). This process does not combine known sentence pieces, but completely new statements are generated. Embedding vectors of tokens are used as a general representation of the content. Crucially, the DNN can capture information contained in a large corpus of documents through self-supervised training, so no annotated data is required. The consequences of the development of this model cannot yet be foreseen.

In terms of musical rhythmic intelligence, DNNs are capable of learning different musical styles from training data and use them to create new pieces (Sect. 9.3.1). According to experts, the pieces are of good quality and certainly exceed the musical abilities of normal humans.

Chatbots can hold a conversation with people verbally or in writing about a wide variety of topics. They are often able to recognize the emotional state of the conversation partner through text, facial expression and voice. They can respond to this adequately so that the other person feels understood and is motivated to continue the conversation. Although many people have reservations about this type of human-machine communication, there are already chatbots with emotion-oriented conversation in Asia that have hundreds of millions of users.

Overall, creativity can be viewed from two perspectives: on the one hand, from the produced image, piece of music or text. If the work produced is unusual, novel, interesting, and could have been created by a human being, it could be considered creative. This is the view of the Turing test, which many computer-created works probably already meet today. On the other hand, one can look at a work from the perspective of the artist. In the best case, the artist brings his entire life experience, his understanding of the world, his view of society, and his emotionality to the creation of a work, and this background is what makes a work valuable in the view of many observers. From this point of view, computer-generated artifacts have an enormous deficit of creative background, and it is not likely that this will change in the next few years.

Trends

- AI has no opinion of its own about artistic value and beauty. Therefore, there are proposals for an AI system to predict people's emotional and sensory reactions to a work of art and incorporate this into the design of a work of art.
- AI systems are good at modifying, extending, and varying existing concepts. It is predicted that human creatives will provide a broad outline and the AI system will fill in the details and do the legwork.
- There is a growing trend in Western countries to regulate chatbots. In 2019, for example, California passed a law requiring chatbots to identify themselves as "artificial identity" if they are to sell a product or influence voters (Cohen 2019).
- Chase Manhattan Bank tested ads on the Internet that had been written by Machine Learning. It turned out that these were clicked by more people than usual ads, in some cases more than twice as often (Pasquarelli 2019). It is therefore foreseeable that Artificial Intelligence will find its way into the advertising industry.
- Academic publisher Springer Nature has unveiled the first research book created with Machine Learning. The book, titled "Lithium Ion Batteries: A Machine-Generated Summary of Current Research," is a review of peer-reviewed articles published on the topic in question (Vincent 2019). It includes citations, hyperlinks to the cited work, and automatically generated reference content. It gives an overview of a field with over 53,000 research papers on lithium-ion batteries from the last three years and offers a chance to keep up to date here.

References

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., et al. (2020). Towards a human-like open-domain chatbot. Preprint. arXiv: 2001.09977

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. Preprint. arXiv: 1607.06450.

BayLearn (2019). Music transformer: Generating music with long-term structure. https://www.reddit.com/r/MachineLearning/comments/a604mb/p_music_transformer_generating_music_with/ accessed on 31.10.2020.

Brilee, A. (2019). *Assessment of MuseNet*. https://news.ycombinator.com/item?id=19749767.

Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *ICLR*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., & Askell, A. (2020). Language models are few-shot learners. Preprint. arXiv: 2005.14165.

Cheng, W.-F., Wu, C.-C., Song, R., Fu, J., Xie, X., & Nie, J.-Y. (2018). Image inspired poetry generation in xiaoice. Preprint. arXiv: 1808.03090.

Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. Preprint. arXiv: 1904.10509.

Chintala, S., Denton, E., M, A., & Mathieu, M. (2018). *How to train a GAN? Tips and tricks to make GANs work*. https://github.com/soumith/ganhacks.

Code-GPT2 (2019). GPT-2 Code. https://github.com/openai/gpt-2.

Cohen, N. (2019). Will California's new bot law strengthen democracy. In *New Yorker*. https://www.newyorker.com/tech/annals-of-technology/will-californias-new-bot-law-strengthen-democracy

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., & Liu, R. (2020). Plug and play language models: A simple approach to controlled text generation. Preprint. arXiv: 1912.02164.

Di Nuovo, A. (2018). How robot carers could be the futur for lonely elderly people. In *Indep*. Online 12 June 2018.

Dredge, S. (2019). Music created by artificial intelligence is better than you think. In *Medium Febr*. 1.

Elgammal, A., Liu, B., Elhoseiny, M., & Mazzone, M. (2017). Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. Preprint. arXiv: 1706.07068.

Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. Preprint. arXiv: 1805.04833.

Fernández, J. D., & Vico, F. (2013). AI methods in algorithmic composition: a comprehensive survey. *Journal of Artificial Intelligence Research, 48*, 513–582.

Gehrmann, S., Dai, F. Z., Elder, H., & Rush, A. M. (2018). End-to-end content and plan selection for data-to-text generation. Preprint. arXiv: 1810.04700.

Glaser, A. (2016). Pepper the emotional robot learns how to feel like an American. *Wired Magazine*, 07.06.2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672–2680.

Guo, X., Zhu, B., Polanía, L. F., Boncelet, C., & Barner, K. E. (2018). Group-level emotion recognition using hybrid deep models based on faces, scenes, skeletons and visual attentions. In *Proceedings of the 20th ACM international conference on multimodal interaction* (pp. 635–639).

Hindupur, A. (2018). *The GAN Zoo*. Retrieved deep hunt Website https://github.com/hindupuravinash/the-gan-zoo.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., & Eck, D. (2019). Music transformer: generating music with long-term structure. In *The international conference on learning representations (ICLR)*.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125–1134).

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In: *The international conference on learning representations*.

King, A. (2019). *OpenAI's new machine learning model*. https://talktotransformer.com/.

von Köchel, L. R. (1862). *Chronologisch-Thematisches Verzeichnis Sämtlicher Tonwerke Wolfgang Amade Mozarts. Nebst Angabe Der Verloren Gegangenen, Angefangenen, Übertragenen, Zweifelhaften Und Unterschobenen Compositionen Desselben*. Breitkopf & Härtel.

Kuyda, E. (2019). *Replika*. https://replika.ai/.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image superresolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681–4690).

Miroshnichenko, A. (2018). AI to bypass creativity. Will robots replace journalists?(The answer is yes"). *Information, 9*(7), 183.

Molteni, M. (2017). The chatbot therapist will see you now. In *Wired June* 7.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., & Fernández, R. (2016). The LAMBADA dataset: Word prediction requiring a broad discourse context. arXiv: 1606.06031 [cs].

Pasquarelli, A. (2019). Chase commits to ai after machines outperform humans in copywriting trials. In *AdAge*.

Payne, C. (2019). MuseNet. In *OpenAI Blog*.

Peiser, J. (2019). The rise of the robot reporter. In *N. Y. Times*.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In: *The International Conference on Learning Representations 2016*. https://doi.org/10.48550. arXiv. 1511.06434 [cs]. Visited on 29 May 2022.

Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., & Sutskever, I. (2019a). Better language models and their implications. In *OpenAI Blog*. https://openai.%20com/blog/better-language-models.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019b). Language models are unsupervised multitask learners. In *OpenAI blog* 1.8 (p. 9).

Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Shuster, K., & Smith, E. M. (2020). Recipes for building an open-domain chatbot. Preprint. arXiv: 2004.13637.

Rosset, C. (2019). Turing-Nlg: A 17-billion-parameter language model by microsoft. In *Microsoft Blog — 13.02.2020.*

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. (2019). Megatron-Lm: Training multi-billion parameter language models using model parallelism. arXiv–1909.

Shum, H.-Y., He, X.-D., & Li, D. (2018). From Eliza to XiaoIce: Challenges and opportunities with social chatbots. *Frontiers of Information Technology and Electronic Engineering, 19*(1), 10–26.

Vardaan, director (2019). *AI to steal fashion model jobs? New AI able to generate entire bodies of people who don't exist*. https://www.youtube.com/watch?v=8siezzLXbNo (visited on 02 September 2019)

Vig, J. (2019). OpenAI GPT-2: Understanding language generation through visualization. In *Data Science Medium March* 5.

Vincent, J. (2019). *The first AI-generated textbook shows what robot writers are actually good at. The Verge 2019* [Cited 05 August 2019]. https://www.theverge.com/2019/4/10/18304558/ai-writing-academic-research-book-springer-nature-artificial-intelligence.

Wang, C., Li, M., & Smola, A. J. (2019). Language models with transformers. In: Preprint. arXiv: 1904.09408.

Wolf, S. (2018). CycleGAN: Learning to translate images (without paired training data). In *Datascience*.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 5907–5915).

Zhou, L., Gao, J., Li, D., & Shum, H.-Y. (2020). The design and implementation of xiaoice, an empathetic social chatbot. *Computational Linguistics, 46*(1), 53–93.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2223–2232).

Zhu, H., Liu, Q., Yuan, N. J., Qin, C., Li, J., Zhang, K., Zhou, G., Wei, F., Xu, Y., & Chen, E. (2018). Xiaoice band: A melody and arrangement generation framework for pop music. In *Proceedings of 24th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 2837–2846).

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2020). *Unpaired image-to-image translation using cycle-consistent adversarial networks*. Preprint. https://doi.org/10.48550/arXiv.1703.10593. arXiv: 1703.10593 [cs] (Visited on 03 September 2022).

Chapter 10
AI and Its Opportunities, Challenges and Risks

Abstract Artificial intelligence has established itself as a central trend topic in the global technology industry in recent years. It is realized through deep neural networks and offers a wide range of opportunities and innovation potential, for example in the smart home, in medicine and in industrial applications. AI has a huge impact on economic development and our working world and poses major challenges to society. Internet corporations are using AI in a variety of ways and, by providing platforms, have now built up monopoly-like structures that concentrate large areas of value creation in the US. Profound changes in the labor market are also to be expected, which can only be mitigated by increased education and training efforts. AI systems potentially allow detailed surveillance of large segments of the population, and elaborate legal and organizational regulations are needed to guarantee citizens' liberties. Researchers and policymakers have therefore developed a testing strategy for an "AI seal of approval", which should guarantee that AI systems deliver the desired results in a verifiable manner. In addition, the systems should not disadvantage any population group, function robustly, respect privacy and be secure against attacks or accidents.

"I'm sorry Dave. I'm afraid I can't do that"

... answers the on-board computer HAL 9000 to a question in the science fiction classic "2001: A Space Odyssey" by Stanley Kubrick. HAL (Fig. 10.1) behaves more and more neurotic during the journey to the planet Jupiter and reacts with fear to its possible shutdown. In this fear, HAL becomes a danger to the crew and develops an uncontrollable life of its own, leading to the death of most of the crew members. To this day, the film industry works with future images of robots and intelligent machines that often evoke negative associations. Many of us have been concerned since the 1968 classic that machines will develop an intelligent consciousness that is no longer accessible to us and that will become a danger for us. Especially when machines communicate in a way that is too human-like, an uneasy feeling creeps over us, and images from such science fiction films come back to our minds.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5_10

Fig. 10.1 The camera eye of the HAL 9000 computer from the movie "2001: A Space Odyssey". Image credits in Appendix A.3

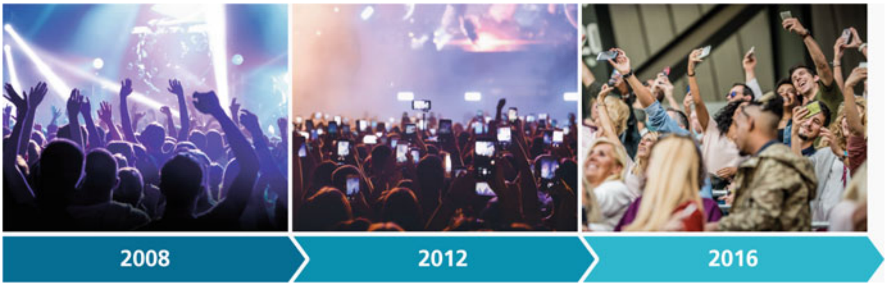


Fig. 10.2 The digital awakening has begun. The way we use digital media has changed fundamentally in recent decades. Image credits in Appendix A.3

In fact, many of the algorithms that a HAL would require have already arrived in our everyday lives—we may just not be aware of this because the changes are so useful. Artificial Intelligence already supports us in many areas, for example via voice assistance systems such as Siri, Alexa & Co., autonomous vacuum cleaners and lawn mowers, or freely available translation services.

The development of AI goes hand in hand with a general further development of media technology: The few years between the images in Fig. 10.2 show the influence mobile devices have on us and the speed at which development has progressed. For example, until 2007, concerts were primarily dance events. Today, they are media events: Concert attendance is digitally recorded, uploaded and shared live over the Web. The Internet has a special identity-forming effect on us, because everyone can and wants to become an object of public interest via large platforms such as Facebook, Instagram or YouTube and generates their own Identity 2.0. It is difficult to foresee which developments, not only technological but also sociological, this change will entail.

The first iPhone was launched in 2007, and apps have only been existing for a decade. Before that, there were no messenger services beyond SMS. Today, it is impossible to imagine everyday life without these communication channels, apps and the associated media practices. Pictures are readily shared, location information

Fig. 10.3 AI becomes the new user interface. Haptic interfaces are replaced by voice assistants

is revealed, emotions are openly communicated. The "circle of friends" knows my status at all times, which concert I'm at, every experience is celebrated in the media. A bizarre highlight was and is certainly the Clinton selfie trend. It's still fascinating how quickly this development has taken place, because all these trends have emerged within the last 10 years.

Our children and grandchildren will grow up in an intelligent environment, with technologies that interact with us unobtrusively yet omnipresently, supporting us by offering services and relieving us of tasks. We are talking here about a progressive process that will have a massive impact on our society, on the job market and on the interaction between humans and machines. Above all, it is speech that is taking on a central role in human-machine interaction. Voice control is simple, intuitive and extremely fast. It overcomes the typing hurdle of a keyboard and the swiping around on a tablet. Few people today are aware of the revolutionary control system of the iPod from 2003. At that time, it worked via a scroll wheel and could be operated intuitively with just one finger. Today, even living environments can be controlled by voice in the so-called Smart Home: Heaters, lamps or music systems respond to human commands. Figure 10.3 indicates the presence of smart assistants that is already possible today. AI will be the new interface to the end user (Hecker et al. 2017b).

A central challenge here is data protection. The data is usually forwarded to a cloud, where speech analysis, information processing and speech synthesis are carried out. The devices must always be ready to receive in order to respond to a possible wake-up call. This is a gateway for permanent monitoring of conversations and sounds. According to the manufacturers, such monitoring does not take place. In 2019, however, there were reports about employees of Amazon listening to and transcribing conversations in order to improve voice recognition (Tagesspiegel 2019).

AI enables new forms of human-machine interaction—via voice control and object and motion recognition. In the coming years, more and more of these assistants will also enter our professional lives. The acceptance rate will be particularly high among the younger generation as they grow up with AI.

It is important to establish a high level of social awareness about how, where and when AI is used. We should expect the same respect, tolerance and appreciation in the digital world as in the analog world. To achieve this, AI must meet certain requirements and standards. For example, AI should not make immoral decisions that could be harmful for humans. Consequently, we must apply at least the same standards to the behavior of AI systems as we do to ourselves.

In the following chapters, we look at the economic opportunities, social and methodological challenges of AI.

10.1 Opportunities for Economy and Society

> *"Information is the oil of the 21st century, and analytics is the combustion engine".*
>
> Peter Sondergaard, Senior Vice President, Gartner, 2011

10.1.1 Smart Home, My House Takes Care of Me

The wonderful 1922 silent film "The Electric House" starring Buster Keaton has many parallels to the ideas of today's smart home. A botanist, who, for the sake of love, rather unwillingly pretends to be an electrician, accepts a job to electrify a house. Those of us who have ever tried to get a gadget to work have an idea of the pitfalls that "smart" technology can bring (Fig. 10.4). In the manner of black comedies, technology gets out of control in the film and threatens to conspire against the inhabitants. It is a worthwhile film that shows us that technology is fallible and we must try everything not to lose control. Many of Buster Keaton's inventions seem superfluous. They are technical gimmicks, a benefit is not always apparent, but they are nevertheless fascinating.

In the case of the electric house, the central control room is shown again and again, where all the control cables converge. According to today's understanding, a smart home is a household in which the devices are no longer networked via cable, but via WLAN and exchange information. In the film, the house of a millionaire is equipped, and this was also the usual user group of a smart home until a few years ago. In the meantime, the cost of installation has dropped dramatically. The smartphone, with its apps, has become the central remote control for shutters,

Fig. 10.4 Scene from the film "The Electric House" by Buster Keaton (1922). Image credits in Appendix A.3

speakers, thermostats and lamps. As is often the case, it was the American company Amazon that initiated another wave in smart control. With its voice assistant Alexa (Sect. 7.7), the mail order company took over the central control hub in our homes. Amazon was never solely concerned with just taking commands via a speaker, but with learning about the habits of the residents and building up home networking as a further business segment. Apple and Google quickly followed suit with offerings, and the market appears to be huge.

What Is the Benefit for the User?

Smart home solutions, which have now reached real market maturity, stand for an increase in convenience. Being able to control the household centrally, even while on the move, can be an advantage. A smart home is the memory of its inhabitants. Lights that have been forgotten to be switched off are switched off automatically. A smart thermostat can use the history of arrival times to effectively heat the house in winter, reducing energy consumption. Finally, each individual appliance in the household can be checked with a smart meter based on its power consumption values. Which of these devices are considered "power guzzlers" in a household is certainly very interesting information for cost transparency, which can be passed directly to the energy consultant.

Using historical data, each individual device in the smart home can proactively respond to the occupants via so-called context awareness and react according to their emotional state. Whether it's the right music in the background, the brightness of the lighting, or a reminder of what still needs to be done. In the morning, the bathroom mirror shows the weather forecast for the day and upcoming appointments and suggests suitable clothing, or even what to buy for the next few days. Of course, you can try on the clothes virtually in advance and test different color combinations in the mirror. The fitness equipment in the house reacts to the current state of health, and compiles an individual program based on the vital parameters and corrects the correct posture by image monitoring. In the kitchen, even the most inexperienced person can get help with cooking. A small decision support system is installed here, which pays attention to the right recipe mixture, regulates the temperature and communicates tips and tricks via voice assistants. The list could go on. At the international trade shows, one new gadget follows the next, whether for daily planning, home security or emergency support.

To ensure that technology does not get out of control, as in the case of Buster Keaton's electric house, providers are currently investing a lot of money in the user-friendliness and connectivity of the devices. In the future, solutions will prevail that, on the one hand, can be easily installed by any age group and any level of affinity for technology without the need for IT support or technicians, and, on the other hand, ensure the compatibility of data exchange with the central control system. In the future, will it be devices from Miele that can be controlled via a common standard across Apple, Google and Amazon, or will there be alternative providers? This again raises the question of platforms that could establish data monopolies. At present, however, alternative companies that combine their data in alliances and build a common ecosystem are not in sight.

On September 7, 2019, there was a massive attack on the online encyclopedia Wikipedia (Wikimedia 2019) that lasted about 9 hours. Via a newly created Twitter account, the supposed attackers justified their online rampage by claiming that they wanted to test new devices in the Internet of Things (IoT). More and more consumers and businesses are adopting networked devices that are characterized by poor security features and rarely receive any updates. This helps criminals immensely to misuse third-party IoT devices for their own purposes without the owners being aware of it. The security of devices, platforms and AI applications must be considered from the outset in the smart home as well as in industrial applications.

This is because the smart home trend will continue and expand to offices, hospital rooms, waiting rooms, and so on. Devices will be networked even more directly with services and AI services, such as shopping or making an appointment with a doctor. In the wake of demographic change, the technical possibilities of a smart home are certainly a positive step for a self-determined life in one's own four walls. Perhaps there will also be a new skilled trade in the future, the smart home installer, who, in addition to providing advice on the wide range of applications and products, will ensure connectivity, but also security. A modern-day Buster Keaton.

10.1.2 Diagnosis, Therapy, Care and Administration in Medicine

> *"It is becoming apparent that a computer will evaluate radiological images first, and only then will a radiologist look at them to make final diagnoses.".*
>
> Professor Dr. Jochen Werner. CEO and medical director of Essen University Hospital (Lepies 2017).

Whether in diagnosis, individual therapy or the management of medical data—AI will play a role in many areas of the healthcare industry in the future. For some time now, AI has been helping to track personal health via fitness wristbands. But AI can now also help with image processing to diagnose disease patterns (Fig. 10.5) in radiology (Sect. 2.1.1) and surgical procedures, i.e., acute care. AI can also be used in pharmacy, where the synthesis of drugs is a major problem (Sect. 2.1.2). Determining which ingredients must be present in what quantities and in what mixture for a drug to have a certain effect is a challenging pharmaceutical problem. Many medical professionals meanwhile associate AI systems with a helpful support tool for the future.

In medical contexts, reservations about sharing data are particularly high, as patient data is one of the most sensitive data we know. It is therefore essential that the

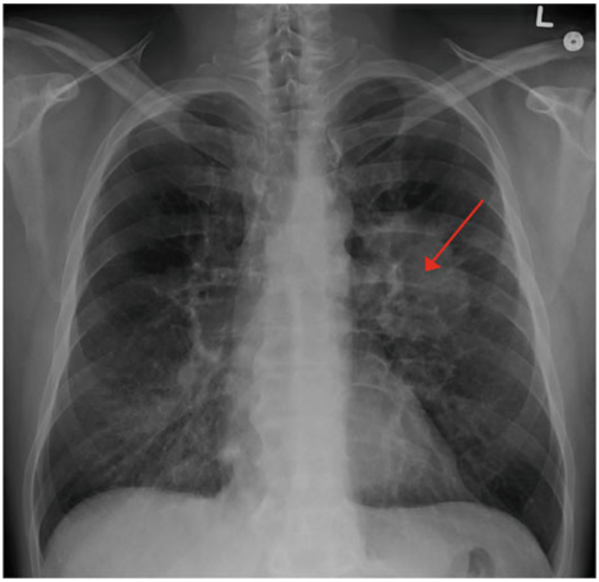


Fig. 10.5 DNN can assist the physician in detecting lung cancer (arrow). Image credits in Appendix A.3

use and also the subsequent benefit of data and AI in the medical context are in line with our social value system. An open dialog is enormously important at this point. The goal is to avoid errors and to relieve nursing staff and doctors from activities that are not patient-related in order to gain more time for direct patient contact. Four potential areas of application with examples of use are presented below.

AI in Early Detection and Diagnosis

AI promises considerable success in the early detection of diseases. This not only benefits patients, but also promises to reduce costs in the healthcare system. For this reason, current research projects are using sensor technology, images and speech to search for indications of a disease at a very early stage. The aim is to diagnose the disease in the initial phase and then stop it or positively influence its course.

Parkinson's disease is the second most common neurodegenerative disease in the world after Alzheimer's disease. It involves the death of nerve cells in the brain. The disease begins creepingly and progresses throughout a person's life. In the course of the disease, the symptoms become more severe and therefore more recognizable. Signs of Parkinson's include slow tremors of individual limbs, impaired speech, deteriorating sense of smell, depression and sleep disturbances. Typically, the disease is diagnosed only at an advanced stage of neuronal degeneration, when the previous symptoms manifest themselves. By then, it is often too late to effectively treat the causes. An European research consortium has therefore joined efforts to enable early detection via smartphone. The "iPrognosis" app registers slight tremors, analyzes voice color and checks finger motor skills with a series of games—so-called serious games (I-Prognosis 2019). For example, the app measures how fluently you use the keys on your smartphone or whether you frequently mention keywords that indicate a depressed mood. The app records the data and sends it anonymized and encrypted to the server for analysis. Those who allow notifications receive information from a team of experts on whether the risk of Parkinson's is increased and a doctor should be consulted. This new form of human-machine interaction can lead to earlier information about health risks and the creation of therapy and training plans.

Skin cancer is one of the most common types of cancer. Health insurance companies are responding to this and are now developing their own apps, especially for diagnosing the particularly dangerous black skin cancer. In one study, 157 dermatologists and an AI system trained on skin cancer were presented with 100 images (Brinker et al. 2019). The results were surprising. Only seven dermatologists performed better than the AI system, 14 performed equally well, and 136 performed worse. This means that, on average, the AI system had better diagnostic results than the group of dermatologists, in which all experience levels from resident to chief resident were represented (Sect. 2.1.1).

At the University Hospital Essen, the radiology department is working intensively with AI systems. The researchers were able to show that MRI (magnetic resonance imaging) images can be successfully and reliably evaluated even with a

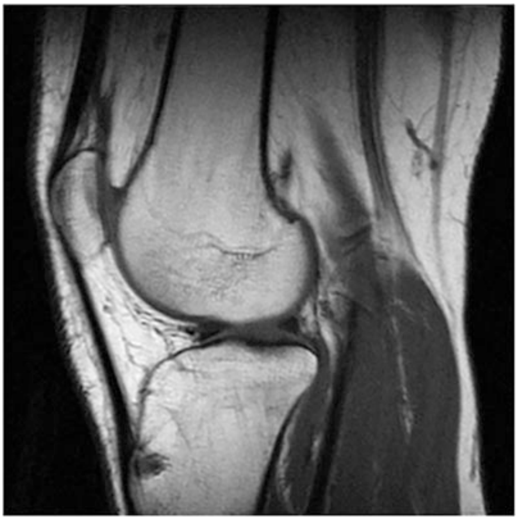


Fig. 10.6 MRI image of a human knee joint, in sagittal slice. Image credits in Appendix A.3

lower contrast (Fig. 10.6). The effect: the examination of a knee takes only 5 minutes instead of 15 minutes (Grävemeyer 2019). AI systems will be pioneers in radiology for many other disciplines in medicine, as digitization and exchangeable data formats are already well advanced here, an enormous advantage for the introduction and development of the systems.

It may sound crazy, but in the USA, German AI expatriate Sebastian Thrun (Fig. 8.27) is currently developing a bathtub at Stanford University that uses ultrasound to check the body for diseases at regular intervals (Mukherjee 2017). This may be a trend that will be established in the coming years. In our everyday lives, we will be alerted to possible poor posture at work and early, not immediately recognizable signs of illness by a wide variety of smart sensor technology.

AI in Therapy

One of the strengths of AI lies in the combination of different data sources. For example, from genomic data to radiological images and sensor information to vital activities, data from different hospitals can be combined. A decision support system generates a therapy recommendation based on the patient's individual constitution and directly provides probabilities for its success. Therapy recommendations are always up to date because they are synchronized with current treatment results. A patient-specific analysis does not stop at medication suggestions, but also directly recommends an individual dosage of the medication, which is currently still carried out in dosage classes. This approach is presently still viewed critically, as current approval procedures for drugs require mandatory group studies, which are no longer possible with greater individualization. An AI system in medicine can have a role similar to that of the autopilot in an airplane: Pilot and computer work closely together here. In this way, a doctor could also use intelligent tools to obtain further information at lightning speed and either supplement or correct his or her personal judgment.

AI in Nursing

In nursing care, a productive use of AI seems to be furthest away due to the still high costs in development and the necessary hardware. Fraunhofer IPA in Stuttgart has been developing a robot assistant for home care or hospital care since 1998. The so-called Care-o-Bot assists with eating tasks, as a carrying assistant, reminds the patient to take their medication, but can also serve as a play partner. The robot thus takes over the tasks of an intelligent butler, but does not replace empathic, friendly nurses (Fig. 10.7) or family members. The lack of empathy also seems to be one of the main reasons for low popularity among the German population (BearingPoint 2017). Although there are repeatedly reports on the use of robots in care, widespread introduction is not to be expected in Europe in the short term for reasons of acceptance and cost.

There are also strong cultural differences in attitudes toward care robots. For example, in Japan, with a currently rapidly aging society, there is less of a negative attitude or fear towards robots. Robots are not culturally perceived as pseudo-humans, but as real colleagues and friends who are registered in the city's legal registers (Ito 2018). One consequence of this attitude is that the emphatic chat system XiaoIce has hundreds of millions of users in Southeast Asia. People like to communicate with XiaoIce and confide their worries to it (Sect. 9.4). Similarly, nursing robots are already being used in Japan to care for the elderly. In future, the support provided by robots may help people to lead a self-determined life in their own four walls even in their old age.

Current developments in the field of intelligent prostheses and exoskeletons appear promising. These are robotic suits that help paralyzed people to get to their feet, assist stroke patients in relearning movements (Fig. 10.8) and assist in

Fig. 10.7 It is not yet foreseeable that nursing robots will be able to perform medical procedures such as injections, but many service tasks such as serving meals or distributing medications appear possible. Image credits in Appendix A.3

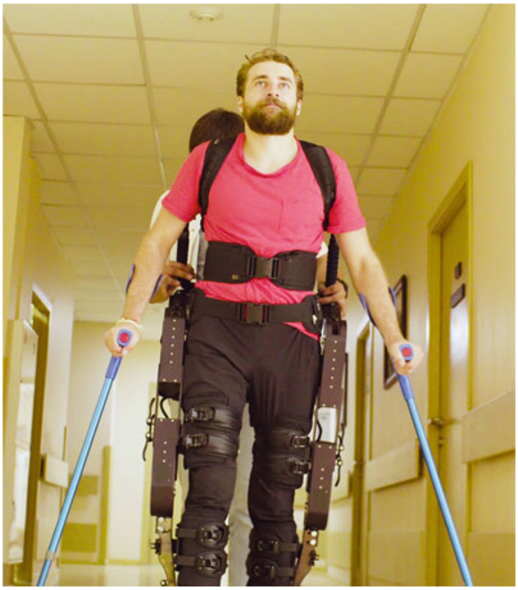


Fig. 10.8 Exoskeleton to assist patients with movement restrictions allow for faster rehabilitation. Image credits in Appendix A.3

lifting heavy loads. When motor impairment is present, an AI system can use recorded nerve signals to predict movement intentions and control the exoskeleton. For example, a 28-year-old paraplegic patient from France trained with a video game avatar system for many months to learn the skills needed to operate the exoskeleton. The patient used the avatar to think about how to walk and touch objects. This was the basic training to connect brain signals with movements. Doctors at the University of Grenoble placed two implants, each with 64 electrodes, under the meninges to record activity. A decoder now transmitted the brain signals, which were translated into movements by an algorithm. This system then sent out the physical commands that the exoskeleton executed. Exoskeletons can provide faster rehabilitation, but also permanent support (Guardian 2019). A possible cost coverage by health insurance companies is still in its beginnings—after all, the German National Association of Statutory Health Insurance (GKV) has included exoskeletons in the list of medical aids (Ärzteblatt 2018).

AI in Hospital Administration

An important lever for increasing efficiency in the administrative area of health care is expected from so-called “Robotic Process Automation”. In this context, “robotic” does not refer to a physical robot, but to software that independently takes over administrative tasks. Especially in the medical context, there are many routine tasks, such as patient admissions, patients’ discharges, documentation requirements or health insurance settlements, which on the one hand keep doctors and nurses from direct medical activities and on the other hand represent a high administrative cost block.

In Robotic Process Automation, bots interact with various applications in the administration via existing user interfaces. The bots copy and sort data, fill out forms, and initiate the next process step completely independently when a certain process is completed—24 hours a day. In the meantime, process mining extracts data traces from the hospital's IT systems, also to identify potential weak points. Where is a bottleneck in the hospital, where are processes not being followed or are insufficiently defined? One of the few German Unicorns is the company Celonis (Petzinger 2018). Unicorns are startup companies with a market valuation of more than 1 billion US dollars, before an IPO or an exit, i.e. a sale of the startup to an investor. Celonis is now active in almost all industries where management leaves a digital footprint.

10.1.3 Machine Learning in Industrial Applications

AI also shows high potential for value creation in the manufacturing environment. Industry 4.0 is the German term for comprehensive digitization in industrial production. After the mechanization of production through steam power, the use of assembly lines and the associated mass production and automation of production through electronics, Industrie 4.0 would be the next industrial revolution (Fig. 10.9). Rigidly defined manufacturing and value chains are to be transformed into flexible and dynamic production systems (Hecker et al. 2017a). These are to enable demand-based, fully individualized production, the so-called "lot size 1". In other words, there would no longer be series, but only individual pieces. The primary goals of

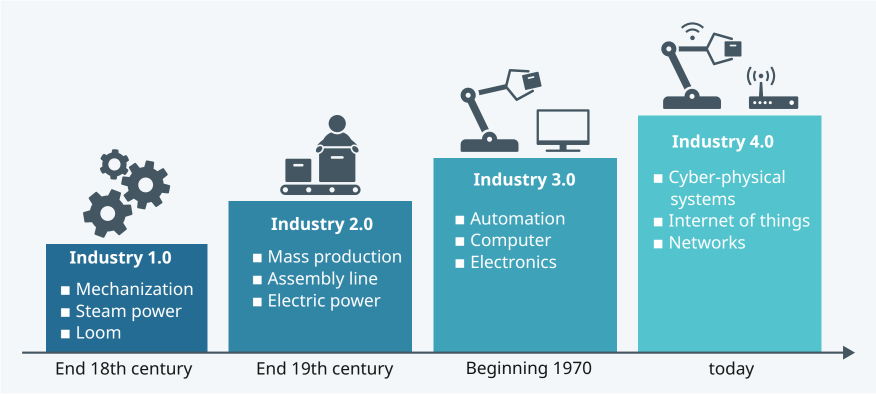


Fig. 10.9 The stages of development of the Industrial Revolution brought about a profound and lasting transformation of economic and social relations

AI in Industry 4.0 are to reduce cost, save time, improve quality and increase the robustness of industrial processes.

The factory of the future organizes itself autonomously by controlling and optimizing logistics, warehousing and production time according to demand. It attempts to predict possible sales taking seasonalities into account, and to integrate them into capacity planning and production processes. "Predictive Maintenance" attempts to detect machine failures at an early stage and avoids downtime due to unplanned maintenance. In some cases, the experienced machine operator can already hear from the noise that a machine may be about to fail. Today, the machine itself can also do this: Anomalies, assembly errors and quality losses can be detected at an early stage via audio, video and vibration sensors, as the machine registers changes in characteristic vibrations and noises.

The advantages of using AI also quickly become clear in quality assurance in the manufacture of gears. In order to avoid a possible damage due to errors during manufacturing, a great deal of human experience and quality inspection is required. Whether this is an early stage condition or an imminent failure is more difficult to detect. The AI system records the production through a variety of sensors and checks the permanent data flow for cracks, impurities, sufficient hardness levels. A break in the gear production would subsequently result in an expensive cleaning of the production machine and all previously produced gears would have to be checked for any defects. Human inspection takes a lot of time and is often prone to errors as a result of varying levels of experience and inspector fatigue. AI systems can reduce production downtime and increase quality.

An important technical prerequisite for the introduction of Machine Learning into the industrial environment is the equipment of systems and machines with smart sensors at neuralgic points (Fig. 10.10). The long life cycles of many production systems make a widespread switch to devices with innovative sensors seem rather unlikely. A remedy can be provided by separately installable acoustic and optical sensors as well as virtual "soft sensors" that can calculate the missing sensor information from other available sensors. In this way, systems that do not yet have a computer interface due to their age can also be included in the analytics. After all, it is precisely the mapping of the entire production chain and analytics in real time that will form the basis for the autonomously controlled factory of the future.

In addition to data acquisition, data management is also of central importance. The sensor data of a machine must be semantically annotated when it is acquired in the manufacturing process in order to be interpreted in a meaningful way. It is not just about the unit of measurement in which the data is measured and which upper and lower limits apply, but also about understanding the role of the data in the business process. A process model is used to describe which machines and sensors contribute to which activities and how the activities are linked. In a domain model, one also describes how the systems are structured.

A particularly interesting concept idea is the "digital twin". A digital twin is the digital representation of individual components or entire production plants of the real factory, including all geometry, kinematics, control and logic data. This gives industry the opportunity to save on cost-intensive physical prototypes and instead

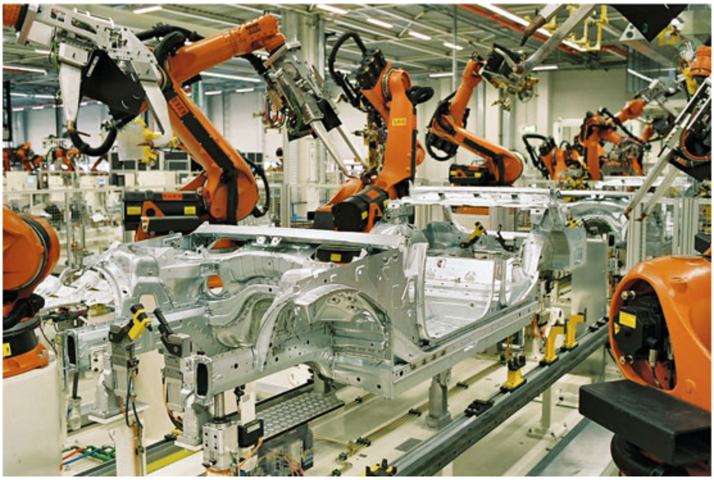


Fig. 10.10 Advanced robots in the car body production at BMW in Leipzig. Equipping all components with their own intelligence allows flexible adaptation to customer requirements, rapid design changes and optimization of the logistics chain. Image credits in Appendix A.3

simulate the behavior, functionality and quality of the real twin facilities in advance from every relevant aspect.

An Internet-of-Things (IoT) platform can also combine incoming sensor readings and messages with both models to interpret the data as complex events in the enterprise process and determine which machines are involved. Complete and high-quality mapping of the data is a very crucial factor here, because in contrast to the B2C market of American corporations, the data volumes are smaller and the tasks are not very fault-tolerant. Transferring a specific task from one company to another is often hardly possible because the configuration of the machines and the products differ. Even if an exchange of data would make sense, it still often fails due to mutual mistrust and the desire to protect one's own know-how. In this context, training data of failures in particular, as shown in the example of the gear wheel, are only available in a few cases, since in an already highly optimized production such failures rarely occur.

Another future scenario is machines that act autonomously in production. Such machines can adjust to the quality of the raw materials and to the current and seasonally changing production conditions (temperature, humidity, etc.) and regulate the machine parameters themselves. This includes the actual manufacturing process, but also the adaptation to the individual product. Machines can accommodate any variations in real time, without the need for programming in advance.

Despite the promising benefits, the use of AI in industrial companies is still quite low. In recent years, companies have invested in the installation of sensors for condition and quality monitoring, and are now collecting data for real-time overviews of production. AI will only be integrated gradually, as the associated cost for the transition in the company structure and value creation stages is considerable.

But industrial analytics and Machine Learning for smart machines and devices in particular offer a field in which German industry and research can, or rather must, develop unique selling points. The combination of engineering know-how and AI can be one of the great opportunities for Germany as a business location.

10.1.4 Further Areas of Application for AI

Chapter 2 already presented a number of applications of AI that achieve or surpass human performance. In the meantime, Machine Learning and DNN have penetrated many other areas of application. These are grouped into eleven application fields in Fig. 10.11. In addition to the already presented application fields smart home, medicine and industrial manufacturing, the other fields will be briefly explained.

In the **media** application field, a far-reaching change from analog to digital marketing channels has already taken place in recent years. In the print media industry, many routine reports concerning sports and finance are now already generated by AI systems (Sect. 9.2.1). With the development of reliable language

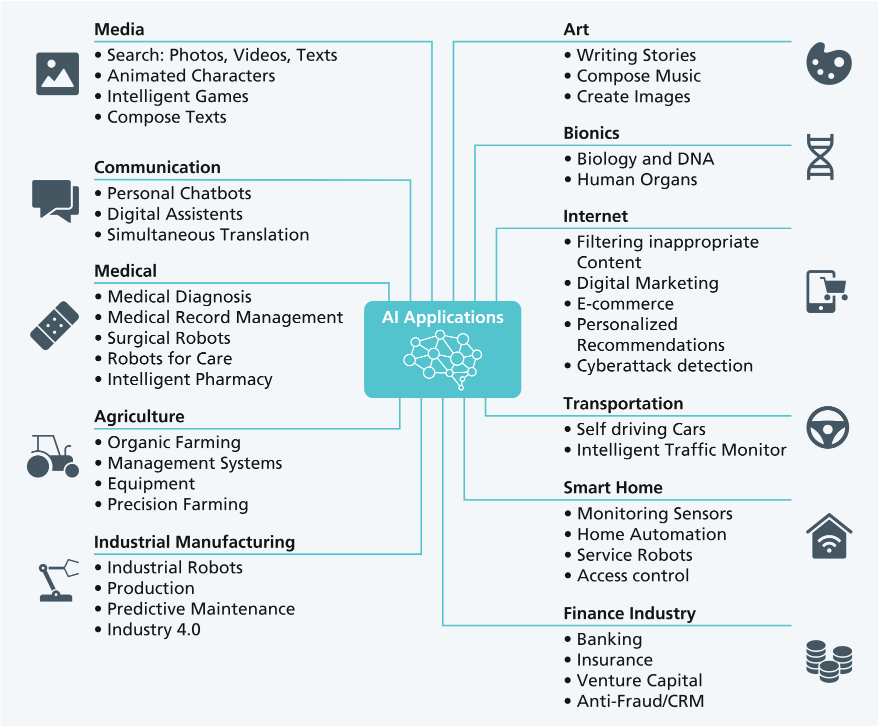


Fig. 10.11 An overview of Artificial Intelligence applications, organized by application area

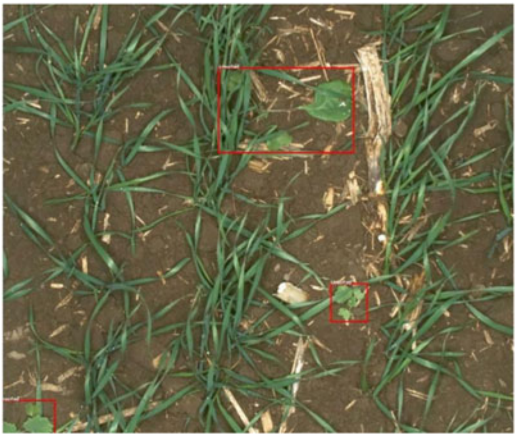


Fig. 10.12 Image identification of the crop (winter wheat) and the weed (highlighted in red). There is a high degree of masking of the weed (Dyrmann et al. 2017). Image credits in Appendix A.3

models, it is now becoming possible to automatically generate consistent stories (Sect. 9.2.3). For example, if you provide the GPT-3 system with a headline, it is able to write an article automatically. AI will also bring about significant changes in other media, such as video and gaming. For example, techniques of image processing (Chap. 5) and reinforcement learning (Chap. 8) will be employed to make game worlds more realistic and automatic opponents smarter and more attractive. It is also foreseeable that characters in video games will be able to have realistic voice dialogues with the player and pursue more intelligent game strategies.

Agriculture is a very important application area of AI. Advances in Machine Learning and robotics have recently led to the development of sensitive and dexterous robots that can accurately identify and pick up individual fruits, plant parts, and bugs. This involves recognizing the plant's degree of ripeness, its potential diseases and pests (Lewis 2019). Bugs and harmful weeds can then be removed Fig. 10.12), the plant can be fertilized selectively, and the fruit can be harvested gently. Fertilizers and pesticides no longer need to be applied over a wide area, but can be placed individually on each plant. This potentially provides a boost for sustainable and organic farming. One challenge here is the often harsh and dirty environment, which requires very robust devices and sensors. AI can also be applied to harvesting, for example to reliably and automatically sort out damaged blueberries or tomatoes. This can significantly improve the quality of the delivered products (Severson 2020).

Communication applications can benefit from the recent advances in the fields of meaning recognition of text (Chap. 6), speech recognition (Sect. 7.3), and voice assistants (Sect. 7.6). Hundreds of millions of people use Facebook Messenger, WhatsApp, Signal, TikTok and other instant messaging services every day. Millions more communicate with voice-based assistants such as Alexa or Google Home. In contrast to traditional media, they support communication in a private circle or even personalized exchanges with a chatbot (Sect. 7.6) that potentially builds an empathic relationship with the user (Sect. 9.4). Spoken language is the most

Fig. 10.13 A damage assessment employee takes notes on a house destroyed by a tornado. Damage assessments may be fully or partially AI-assisted in the future. Image credits in Appendix A.3

efficient communication channel for humans, and it is expected to push back text-based communication in the future, especially in developing countries with literacy deficits.

The **financial industry** is an economically important field of application for AI. Machine Learning has been used for some time to estimate future prices of shares and financial derivatives from current prices and economic statistics. In normal times, these forecasts are quite reliable, but during the financial crisis they led to severe misjudgments because the models were based on unjustified assumptions and were used outside their training data. As additional information, for example, sentiments and opinions about companies, brands, and political issues can be extracted from news texts (sentiment analysis, Sect. 6.7.1) and used for economic forecasts.

An important application is the assessment of insurance risks and losses Fig. 10.13, e.g., by interpreting images (Sect. 5.5) or texts (Sect. 6.7). Already known for a longer time are the evaluation of the creditworthiness of persons (credit scoring), the detection of fraud in credit card transactions, and the detection of money laundering. The evaluation of insurance risks, for example, can be improved by sensors in cars and machines. For example, drivers with a cautious driving style can receive a premium reduction. Voice assistants and chatbots can be used for customer contact, which then provide simple services. Overall, it is foreseeable that AI systems can be used in wide areas of the financial industry.

Transportation has the self-driving car as its most promising application (Sect. 8.5). The technology is still being tested (Fig. 10.14). When it finally is introduced in a few years, we can expect massive changes in traffic (e.g., low-cost autonomous cabs), less parking space required for private cars, and greatly reduced accident rates (Sect. 8.5.3). In parallel, AI will be used for traffic control, leveraging information from a variety of high-resolution video cameras and other sensors. This will alow the introduction of flexible, traffic-based tolls that can reduce traffic pollution in city centers and reduce exhaust emissions.

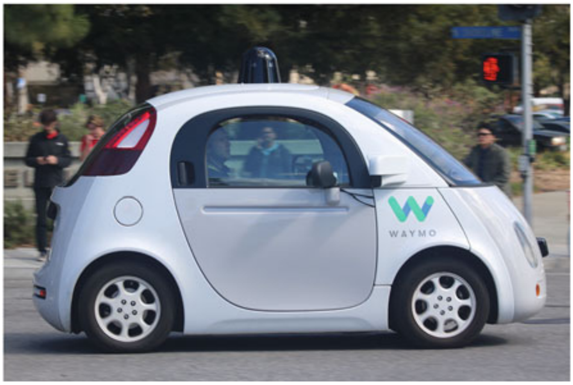


Fig. 10.14 Self-driving car from Waymo. Special sensors are installed on the roof. Image credits in Appendix A.3

The **Internet** itself is a significant application area for AI. Digital platforms are now required to develop filtering algorithms to remove inappropriate language, hate comments, child pornography, and other types of content. These algorithms are based on image classification (Sect. 5.5) and semantic interpretation of language (Sect. 6.5), and must strike a delicate balance between freedom of expression and the prevention of punishable acts.

Central drivers of the Internet are personalized recommendations of products and new websites for users. The more precisely these recommendations are tailored to the user's personal needs, the higher the revenue for the providers. For example, it is estimated that 35% of Amazon's revenue is induced by its AI system for recommendations. The goal here is "upselling,", i.e. the customer buys additional products after they have made their actual purchases. Another booming application is so-called "dynamic pricing", which sets the price depending on the customer and his predicted motivation to buy. One problem arises with personalized filtering of news, when a user only receives messages that support his or her fixed worldview and prejudices. There is a broad discussion about how such filter bubbles can be avoided.

Bionics, i.e. the combination of biology and technology, is a still small field of application for AI. One approach is the interpretation of brain signals by an AI model. For example, a prosthetic arm can be controlled by the wearer via a nerve interface (Bryant 2019). The prosthetic arm has 26 joints and pressure and traction sensors in each fingertip. However, communication also works in reverse (Fig. 10.15): For example, the human receives information about temperatures and vibrations via sensors on the prosthesis (Osborn et al. 2018). A prosthetic leg works in a similar way, where the user can feel bending movements, pressure, and vibrations (Scudellari 2019). The findings of bionics are now playing a major role in competitive sports. Multiple cameras and other sensors are used to measure athletes' movements and suggest changes to improve performance. In soccer, for example, this can be used to identify players who are particularly good at landing corner kicks or free kicks. This is then incorporated into the transfer evaluation of the players (Smith 2020).

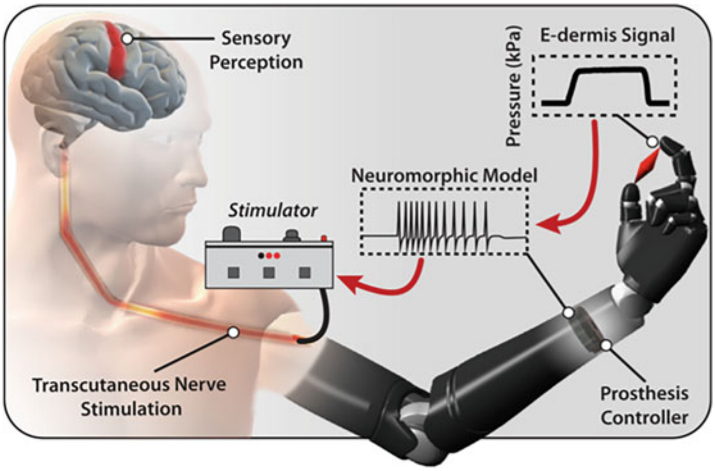


Fig. 10.15 A prosthetic arm can provide feedback to the wearer through sensors. In addition, tactile information from the fingers can be transmitted to the brain to signal touch and pain (Osborn et al. 2018). Image credits in Appendix A.3

Art can be expected to become a major application field of AI. Chapter 9 described that AI can perform creative works in some areas. It is discussed whether these are "genuine" creative inventions or just variations of already existing works. Marks (2019) argues that humans are good at choosing among a set of alternatives. In contrast, they have problems producing a large number of variations from which to choose. According to Marks, AI systems are good at producing many possible different designs and can thus help artists, designers, musicians, and writers in their work. Art should be original, but not deviate too much from familiar forms. For example, one can automatically generate new paintings in such a way that they are generally perceived as art, but cannot be clearly assigned to any previous style (Sect. 9.1.8). Such artworks could not be distinguished by human viewers in terms of their artistic expressiveness from paintings offered at current art fairs. Overall, the authors also see that AI could take on a more independent role in art in the future and produce original contributions.

10.2 Economic Impacts and Interrelationships

AI has now established itself as a key trending topic in the global technology industry. In the coming years, AI will have an enormous impact on economic development and on our working world. And it will also fundamentally change the competitive dynamics in many industries. In the coming sections, we will examine at the key drivers of this development, look at what trends are being set by the major technology companies, and discuss the consequences for the labor market.

10.2.1 The Monetization of Data

A few years ago, American Internet companies were the first to collect immense amounts of data using Big Data technologies on high-performance hardware and software platforms. U.S. Internet corporations—followed by Chinese companies—continue to lead the way in creating value from data. For them, data has become a production and competitive factor and thus a key economic asset.

Many companies took notice when Google bought Nest in 2014 for the then unimaginable sum of $3.2 billion (Miners 2014). Nest had something special to offer with its exceptionally smart thermostats (Fig. 10.16): access to the smart home and the data collected from millions of American homes. The Nest smart thermostat records a wide range of information from the homes: when the temperature is adjusted up or down, how high the humidity is, what's the brightness of the room is like, etc. Nest uses this to make forecasts, e.g. when the residents are at home. The thermostat could be controlled directly via an app. Nest's purchase price attracted a lot of attention at the time, as the value of a company is traditionally assessed according to three main aspects (Fig. 10.17): the qualification of the employees, the available capital and the value of tangible assets (Wrobel and Hecker 2019).

In the case of Nest with its 280 employees at the time and a low capital and material value, it is clear that the collected thermostat data must have played a significant role in the company valuation and the resulting purchase price. The takeaway from these and many other acquisitions is that data as an intangible asset should represent a balance sheet component for companies in the future (Fig. 10.17). This is an important consideration for the future strategic orientation of companies, which has changed the perception of data.

That's why it's worthwhile for companies to collect or purchase, curate, protect, intelligently link and evaluate data. A modern car, for example, is not just a means of transportation, but also a mobile weather station. With thermometers integrated as standard, rain sensors for activating the windshield wipers, brightness sensors, humidity meters and GPS, important data can be collected and transmitted

Fig. 10.16 Intelligent thermostat from Nest. Image credits in Appendix A.3

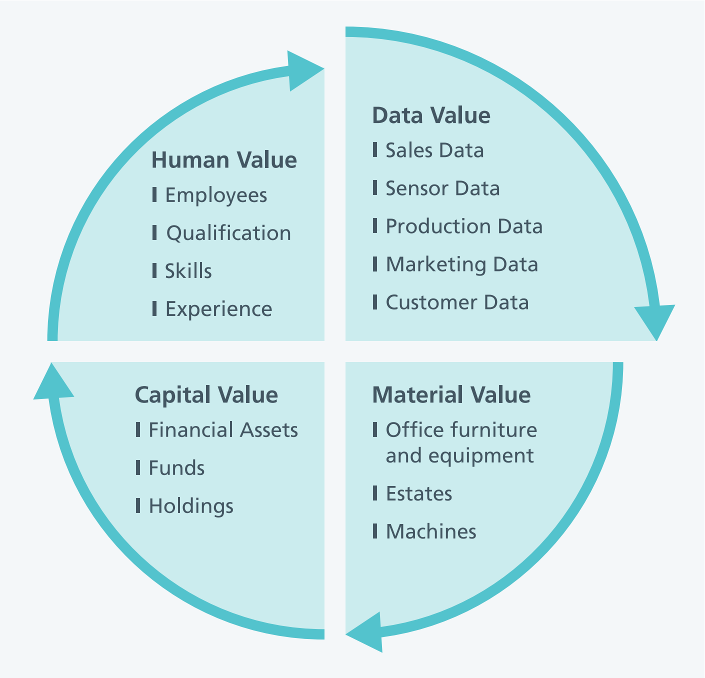
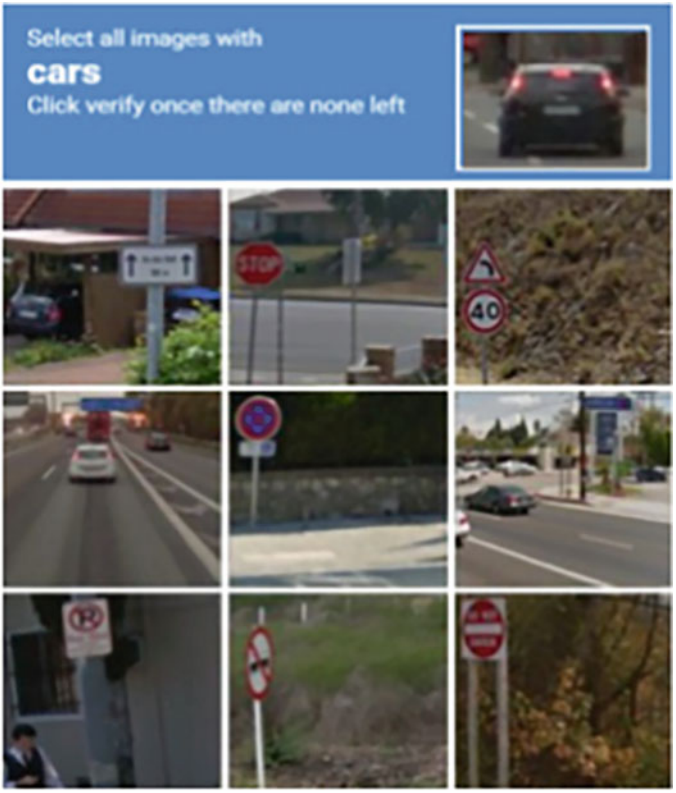


Fig. 10.17 In addition to employees, capital and materials, the classic pillars of corporate value, the availability of data is playing an increasingly important role in the valuation of a company

via mobile networks. When aggregated across many vehicles and compared with satellite images, this results in a locally very finely resolved picture of the current weather without substantial investment in infrastructure, a spectrum of data that should not be underestimated. For smart agriculture, for example, this information can be used to fertilize crops more precisely and to determine the harvest time locally to the point.

So every company's AI strategy also includes a data strategy. Above all, this means drawing up a data map of one's own company and formulating a data expansion strategy. A company should know which data is a unique selling point, in what quantity data is available, how it is linked and annotated, and how the scope and quality can be improved.

American corporations show how subtle you can be when collecting data. Many of us will have already run a check like the one in Fig. 10.18. It comes from Alphabet subsidiary Captcha and verifies whether you are a bot or a human. Captcha is made available to website providers as a free service, with the very clever goal of obtaining annotated images for Machine Learning. Every day, the manual classification of street signs, buildings, etc., generates millions of training data for autonomous driving.



Select all images with
cars
Click verify once there are none left

Fig. 10.18 Generation of training data using the example of an image captcha (Wang et al. 2018). The authors point out that with improved image classification, image captchas will soon be automatically solvable and propose captcha tasks with visual reasoning. Image credits in Appendix A.3

After the huge amounts of data from the Internet, social media, e-commerce and many other apps on our smartphones, it will be the networked devices and systems that will drive up the flood of data in the coming years. Figure 10.19 shows an estimate of data growth (Kanellos 2016). Meanwhile, 30 billion networked devices are expected by 2020 and 80 billion by 2025.

The Internet of Things includes smart devices in the private sector, such as various wearables (smart watches, health and fitness trackers) and intelligent building control (smart home, smart house), on the one hand, and integrated solutions such as intelligent traffic light control, the intelligent power grid (smart grid) and predictive traffic management in the intelligent city (smart city), on the other. The Internet of Things also offers enormous potential for Artificial Intelligence intelligence in the industrial environment—one of the great opportunities for Europe.

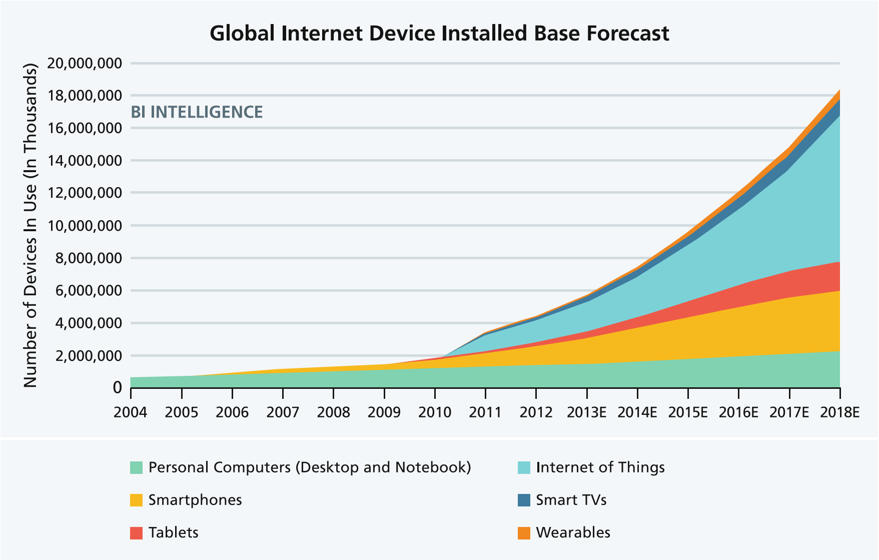


Fig. 10.19 Projected data growth in the coming years. Self-created graphic (Kanellos 2016)

At the same time, Europe should also go its own way when it comes to data storage. Google, Amazon and many other companies form data monopolies where the data is stored centrally by the provider. This centralized data power should not be a model for Europe. The idea is to build a decentralized data infrastructure in which data is only exchanged between companies and individuals via interfaces for specific applications and purposes. This has the advantage that no dependencies arise and data is only transmitted with the consent of the users via a data broker, possibly against payment. As an individual or company, you remain the sovereign owner of your data and can trace the processes involved in handling it. In addition, open interfaces to freely available public data create a data space that generates a wide range of opportunities for start-ups and new cross-sector business models. The Medical Data Space can be taken as an example here. This processes its sensitive patient data in a decentralized manner and not in a Germany-wide central registry. The data therefore remains on site. Only anonymized model results are forwarded, which then provide a more meaningful picture of the patterns of diseases, therapeutic successes, etc. in total.

10.2.2 The New Digital Service World: AI as a Service

> *"Software eats the world".*
>
> Marc Andreessen, inventor of Netscape, 2011

AI is a key driver of digitization in all areas of business and life. From the initial hype, AI has become a constant companion. A multitude of new services and offerings are emerging. In the future, digital products will hardly be found without AI components. AI can fundamentally change existing value chains. Three economic models are relevant here:

1. **Platform Economy:** Digital platforms are intermediaries that connect providers and customers via a platform and enable interaction via apps, skills, and search functions. Platforms have become the dominant business model of the digital economy. The companies with the highest turnover worldwide have one or more platforms as a core component of their product and service portfolios and are developing monopolization tendencies in their application. For example, the Amazon (Fig. 10.20) retail platform, the Google Play Store and the WhatsApp communication platform are among the dominant providers.
2. **Economy of Speed:** Companies and start-ups strive for a first-to-market strategy by trying to enter the market early and often with high financial risks in order to gain a monopoly position. To achieve this, infrastructures, facilities and processes are strategically set up within the companies' organization in order to develop products in the shortest possible time.
3. **Economy of Scale:** Another competitive strategy is to achieve advantages by optimizing production costs. The aim is to become the cost leader and thus make it more difficult for competitors to enter the market or take market share away from them. Efficiency gains are achieved in the automation of production, with better forecasting, better decisions and faster actions reducing unit costs more

Fig. 10.20 Amazon has built one of the most successful digital platforms. It includes consumer goods trading, streaming services, digital assistants, cloud computing, etc. Image credits in Appendix A.3

and more. In addition, the investment costs of doubling the number of users of a platform are relatively low and promise high profits with a corresponding market share.

Companies are increasingly orienting themselves towards the outlined models and are demanding a higher degree of innovation speed and scalability of their solutions and services. For many, it is an immense challenge to adapt their existing structures and develop a so-called digital DNA, a digital mentality (Hecker et al. 2016). A corporate culture in which trial and error, even failure, and, of course, success are appreciated is certainly crucial for success. Fear-free thinking beyond previous boundaries, beyond one's own departmental structures, must be possible. It is precisely through interdisciplinary, cross-departmental and cross-company collaboration that innovation is born. Companies must create the right framework conditions for this. These framework conditions also include AI competence. From top management to individual employees, it should be understood what AI can do and where the limits lie. When such a culture has been established, AI provides the tools for innovation, effectiveness, speed and scalability.

AI software is increasingly made available "as a service (AIaaS)". The software is not offered for installation and operation in the company, but as a service via the Internet or a platform. The user is not the owner of a software version, but merely a subscriber to services based on a specific billing model. He avoids the need to set up the IT infrastructure, maintain the software and obtain the necessary IT expertise, and may therefore be faster and more cost-efficient on the market with his own products. Even a sudden increase in customers can usually be handled without major hurdles, as the service provider only has to add further computing capacity via the cloud. The service provider has further advantages. He can implement market requirements and customer wishes more quickly by adding to the services. He has direct access to the necessary user data—often beyond that—and can thus continuously improve his service. American companies such as Airbnb, Uber, and Amazon show how it is possible to establish oneself as an intermediary with a service platform and without other capital goods.

In 2011, Marc Andreessen announced his famous quote *"Software is eating the world"* in the Wall Street Journal (Andreessen 2011). That predicted that the world economy would soon be "fully digitally wired." Software is at the heart of an improvement in three dimensions: reduced costs, expanded market access and shortened development cycles. Few significant things happen today without software playing a critical, often even the decisive, role. Andreessen's quote goes hand in hand with the idea that everything that can be digitized will be digitized.

AI can be applied in very many areas thanks to digitalization. There are actually no limits to the imagination. Work that is repetitive and therefore easily tedious will be able to be done entirely by machines in the foreseeable future, such as automatically filling out forms, checking completed forms and processing correctly completed forms. AI will take over many routine activities, either as an assistant or completely autonomously. Companies that already collect data on such activities have major advantages over those that do not.

Artificial Intelligence is the logical continuation of digitalization, because it is AI that enables us to learn from data and create additional value for companies and society. Learning systems enable previous tasks to be solved better, faster and more cost-effectively—this creates an efficiency gain in competition. We also benefit from AI in the private sphere. Services such as Netflix, Amazon, the autonomous vacuum cleaning robot, foreign language translation, etc. make the development of AI a permanent companion and useful assistant for us in everyday life. Moreover, AI has become an indispensable research tool for almost all disciplines. With great enthusiasm for experimentation, scientists beyond computer science and mathematics are also using the tools to analyze large amounts of data. It is almost impossible to enumerate the many fields of application and possibilities. For us, however, it is already apparent that human-machine interaction will change significantly.

10.2.3 Large Companies as Drivers of AI

> *"From Mobile First to AI-First."*
>
> Sundar Pichai, Google CEO, 2016

Unlike in the U.S. and China, German companies do not have access to mass data from social media, smartphones or smart speakers. The term "GAFA" is often used in connection with artificial intelligence. GAFA are the four big tech companies Google, Amazon, Facebook and Apple. If we compare the stock market value of the companies in 2018 on the basis of the circles in Fig. 10.21,we see a clear dominance of the platform companies from the USA and China. Only the German SAP has a significant size in the global competition. However, since a large part of the global data transfer takes place via the platforms, an imbalance arises for Europe, especially in direct customer contact (business to consumer, B2C). There is a risk of monopolies being formed by these companies, which threaten Europe's digital sovereignty through their data control and global competitiveness.

The AI Company Google

If we look at the development of Google (and Alphabet) as representative of GAFA in the past few years, the company now has little to do with a pure search engine provider (Fig. 10.22). From home automation with Google Nest to autonomous driving with Waymo to digital high-performance infrastructure with Google Fiber: Google has repositioned itself as an AI company and has virtually entered all relevant future topics.

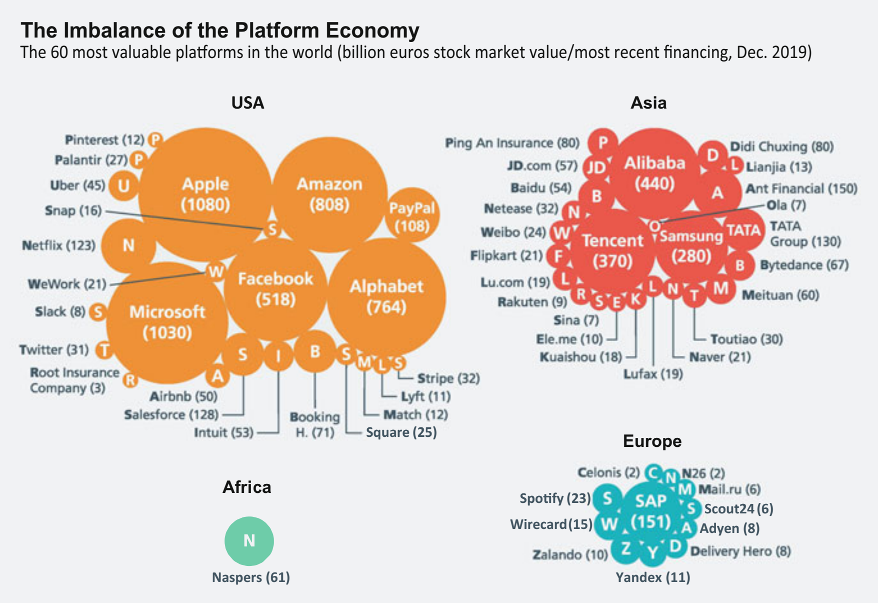


Fig. 10.21 Among the world's 100 largest platform companies, the Americas and Asia-Pacific are clearly ahead of Europe in 2018 (Hossain et al. 2018). Self-created graphic

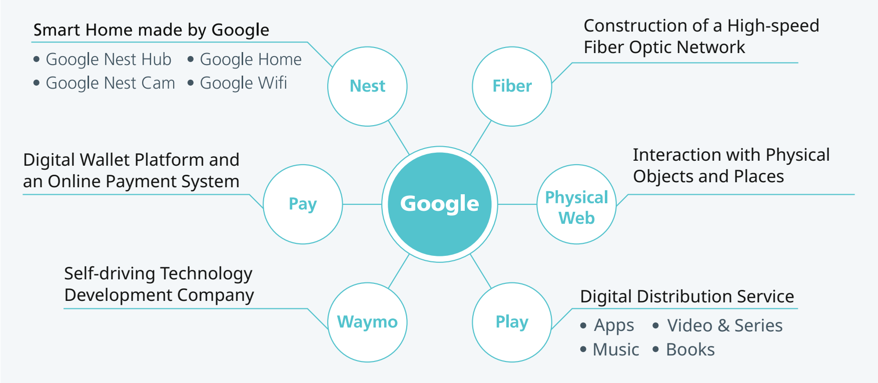

Smart Home made by Google
• Google Nest Hub • Google Home
• Google Nest Cam • Google Wifi
Nest
Construction of a High-speed Fiber Optic Network
Fiber
Digital Wallet Platform and an Online Payment System
Pay
Google
Interaction with Physical Objects and Places
Physical Web
Self-driving Technology Development Company
Waymo
Play
Digital Distribution Service
• Apps • Video & Series
• Music • Books

Fig. 10.22 Google has developed from a search engine provider to a global IT company

Google (or its holding company Alphabet) still offers its search platform and entices users with other free services like Google Maps for navigation, Gmail as an email provider, Android a smartphone operating system, Google Translate for translation or YouTube as a platform for video clips. Regardless of the interaction, Google is getting to know us users better and better and can place advertisements more and more precisely. For Google, we users are the product it sells to its customers, the advertisers.

Through its many expansions and occasional takeovers, Google has been nipping many developments of independent service providers in the bud, which is becoming a problem for an open market economy. Europe urgently needs its own platforms as alternatives that benefit customers and at the same time comply with data protection.

And finally, the tech giants can pay their AI experts exorbitant salaries. They specifically use headhunters at the central Machine Learning conferences. Anyone who has good ideas and writes a publication with a high degree of innovation does not remain hidden from these companies.

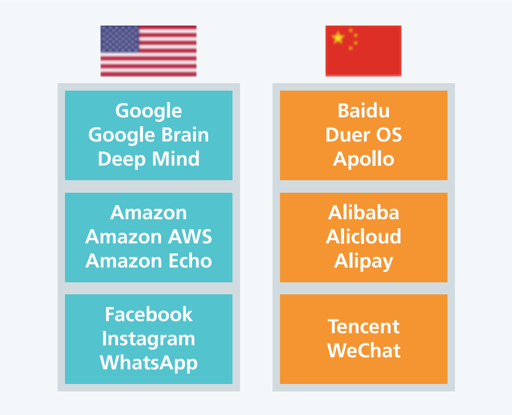
The Chinese Competitors

Meanwhile, almost every American platform also has a Chinese counterpart, although these are still less well known in Europe. These are the companies Baidu, Alibaba and Tencent, abbreviated: BAT (Fig. 10.23).

The speed at which Chinese companies are developing is remarkable. Led by business idols like Alibaba founder Jack Ma, they are making the most of their entrepreneurial freedom. Once founded as a Chinese clone of Amazon and Ebay, mega-corporation Alibaba is investing billions in an AI research offensive, including collaboration with top researchers at selected elite US universities. IT corporations such as Baidu and Tencent maintain AI research centers in Silicon Valley. For the tech giants Baidu, Alibaba, and Tencent, China is no longer big enough, and global offers and investments are being prepared. The Chinese government has managed to create incentives for entrepreneurial creativity and risk-taking without a loss of political control. In particular, the dynamic start-up landscape continues to produce new unicorns (Castro et al. 2019), which are given good conditions by the willingness of domestic and foreign venture capital to invest.

Furthermore, the high willingness to embrace technological innovations in China is one of the most important factors behind the country's rapid growth.

Fig. 10.23 American IT companies and their Chinese counterparts



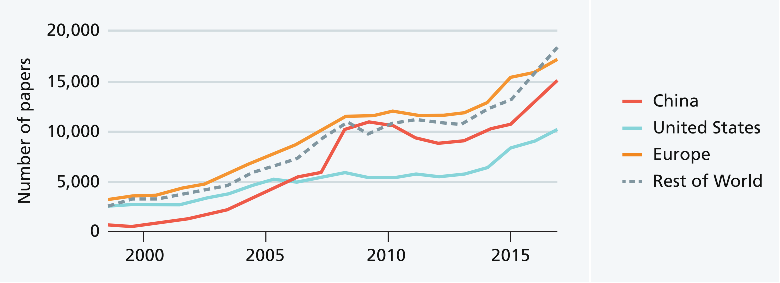


Fig. 10.24 Comparison of the number of AI publications published annually in different regions (Shoham et al. 2018)

Numerous developments in digital transformation are at odds with humanistic western principles and values. Unlike the USA, China very early had a national AI strategy. China aims to be the world's leading AI nation by 2030. Figure 10.24 shows the rapid increase of AI publications since 2008, a first indication of the Chinese catch-up process (Shoham et al. 2018). The prospects for China are good: the mass of Chinese users, low privacy requirements, a highly protected domestic market, and government support have allowed global companies to emerge here. China has thus managed to achieve digital sovereignty in just a few years.

Chance for Europe

Even if Europe cannot catch up in the consumer sector (business to consumer, B2C) within the near future, it must at least gain digital sovereignty in the industrial sector (business to business, B2B). To achieve this, Europe must develop AI technologies that use data economically and can also learn with little data. In this context, one speaks of hybrid AI. This combines data-driven and knowledge-driven methods. While the data-driven approach leverages statistical correlations and automates the analyses with Machine Learning methods, the knowledge-driven approach uses established knowledge provided by humans and derives conclusions from it. This is where the experts come into play, for example doctors in medicine or engineers in production. Their experience and specialized knowledge can be expressed as facts, rules and physical laws and integrated into Machine Learning. An overview can be found in (Gromann et al. 2019).

However, industrial analytics and Machine Learning for intelligent machines and devices offer a field where German industry and research can and must develop unique selling points. Driving innovation at this point entirely without engineering know-how will not work. This is one of the opportunities for Germany as a business location.

10.2.4 Impact on the Labor Market

Concrete forcasts for the labor market are very difficult due to the highly dynamic nature of the topic. However, it is safe to say that many jobs will change as a result of the influence of AI. Computer programs can work around the clock and on weekends, make no demands and have no union. That makes them attractive for many jobs. An AI system trained to diagnose radiology images is always focused and does not make careless mistakes. AI applications achieve a consistent quality that humans can't match in the long run.

In Fig. 10.25, the IAB research institute of the German Federal Employment Agency forecasts the degree to which jobs will be affected by digitization (Kropp et al. 2018). Major changes are foreseeable for employees in manufacturing, mining, financial and insurance services, energy supply, and other economic services. For example, since manufacturing comprises about 22% of the workforce, about 12% of all German employees are affected by digitalization in this sector alone. On the

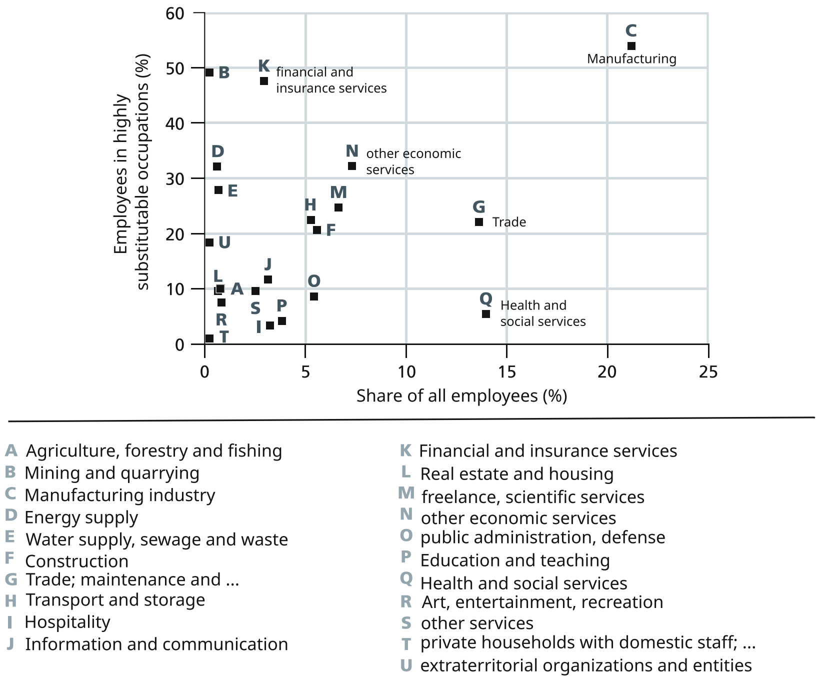


Fig. 10.25 Share of employees in occupations with high substitutability values in Germany by economic sector in %. Reading aid: 55% of manufacturing employees (C: black square at top right) are employed in highly substitutable occupations. At the same time, they constitute about 22% of the workforce in Germany (Kropp et al. 2018)

other hand, only a few changes are expected to result from digitalization in the people-oriented service sectors, such as healthcare and social services.

We have known since the development of the steam engine that people working on assembly lines or in heavy physical labor in particular face the risk of losing their jobs because machines replace them. Academics felt safe for many years, but this is now different. After the automation of manual work, mental work is also being replaced. After all, understanding, decision-making and action can now be automated, at least for simple routine work. As an auditor or lawyer, you thought automation wouldn't affect you because a robot couldn't do this high-value work. But many processes can now be automated here too. Basically, AI and digitization will take over many routine tasks, either completely autonomously or as an assistant. Thus, humans will be freed from tiresome manual or tedious tasks and will be able to perform more demanding tasks with the support of AI assistants and recommendation systems.

We will have to learn to cooperate with machines and digital tools. Qualification will secure jobs at this point in the future. Germany should take a pioneering role in the implementation of digitization and AI, thus creating advanced AI-centric activities and induce an increase in demand from abroad due to the competitive advantage. If, on the other hand, other countries make faster progress in implementation, Germany would lose international competitiveness and the positive demand effects would be reversed (Kruppe et al. 2019, p. 16).

Education for a Digital World

As the professional world is rapidly changing due to AI, education must also be reformed. We cannot expect to be able to stay in the same job for the rest of our lives in the future, at least not in the way it was originally communicated. Rather, it should be assumed that digital technologies will tend to make numerous fields of employment more demanding and complex in terms of interaction with IT. Sufficient IT and methodological knowledge for everyday use, but also knowledge of the limits of the technology, will become increasingly relevant. Lifelong learning and further training on the job must be a component of corporate training as innovation cycles become shorter and shorter. General school education, vocational and university training, and in-company continuing education are important factors for securing skilled workers in the future, because a shortage of IT specialists has been a problem in the German economy for years.

It is an investment in the future to introduce young students to digital technologies as early as possible, to make programming fun, and thus to increase their interest in STEM subjects. In Japan, for example, basic knowledge of computer science, programming and algorithms is a mandatory part of the curriculum. Being able to recognize simple algorithmic structures and formulate them oneself trains important key skills such as analytical thinking, creativity and the ability to solve problems independently. With software code playing an increasingly fundamental role in our lives, it is becoming more and more important that many people know how to use it and not just a small part of society. Even children should be taught not only to operate buttons, but also to understand how the technologies behind them work. Programming is thus becoming a kind of foreign language in schools. English, French, Spanish and Python naturally belong in a common curriculum.

It is equally important to modernize educational processes. Digital media enrich lessons and can play an active role in structuring them. Digital tools facilitate the implementation of different methods and tasks in many places. For example, digital learning environments can adjust to the individual learning speed of each student and, if necessary, repeatedly explain complex contents using varying examples. It can be expected that digital textbooks and video materials will make learning easier, adapt the content to current events, and thus improve the quality of teaching. A computer, laptop or tablet PC for every student in German classrooms, a functioning WLAN connection, a powerful communications system—we are currently a long way from this purely infrastructural goal. However, there must be an awareness that digital education is the basis for Germany's economic future.

This is a key task, especially considering the upcoming demographic change. In the coming years, fewer and fewer people will be available to the labor market. Digital competence must therefore be taught in all phases of life and education. All employees should be able to benefit from these measures, regardless of their education, qualifications, age, gender and working time models.

In this context, it is quite central for education to teach important basic skills that cannot be taken over by an AI in the future either. With their creativity, their intuition, their view of values in contrast to the capabilities of algorithmic replicability, scalability and processing of large amounts of data, humans and machines can exploit their individual strengths and compensate for each other's weaknesses. However, for value-creating collaboration, humans also need a clear understanding of AI's capabilities and limitations.

The decisive factor is that AI is currently, and will continue to be in the future, a profitable and useful support tool for humans, relieving them of work so that they can concentrate on other activities. To achieve this, humans need the skills to apply AI technologies appropriately. However, digitization will not replace subject matter experts and their specialized expertise in certain complex fields. Human experts will always be needed to contribute their specialist knowledge and experience.

The Job Description of the Data Scientist

"Data Scientist: The Sexiest Job of the 21st Century."

Harvard Business Review, 2012

With the Big Data wave in 2011, a new, extremely attractive job description emerged. Data scientists are the central professional group in Big Data and AI development. They keep track of exponential data growth and gain insights from which new business models and services emerge. The Data Scientist can also be described as a data detective, data researcher, data-"is there anything else?" (Fig. 10.26). From the beginning, it was clear that the qualification of personnel in this field will become a bottleneck. Indeed, Fig. 10.17 has already illustrated how data volumes have become an essential corporate asset. Not only sensor data processing for smart devices, but also the need for image, text, and speech processing for digital assistants makes Machine Learning experts an extremely rare resource. Search job portals for "Data Scientist" and you will find hundreds of open positions. Currently, the demand far exceeds the supply of available Data Scientist, and this is expected to continue for several years. The Harvard Business Review rightly proclaimed the Data Scientist the *"sexiest job of the 21st century"* in 2012 (Davenport and Patil 2020).

Data Scientists can currently find employment in all industries, whether in commerce, medicine or logistics. When deciding on a company, it is not always the salary that is important, but rather the access to interesting data sets, the degree

Fig. 10.26 Dhanurjay "DJ" Patil was appointed by U.S. President Barack Obama in 2015 as the first Chief Data Scientist of the United States Office of Science and Technology Policy. Patil stated that the mission of the U.S. Chief Data Scientist is to responsibly unleash the power of data for the benefit of all Americans. Image credits in Appendix A.3

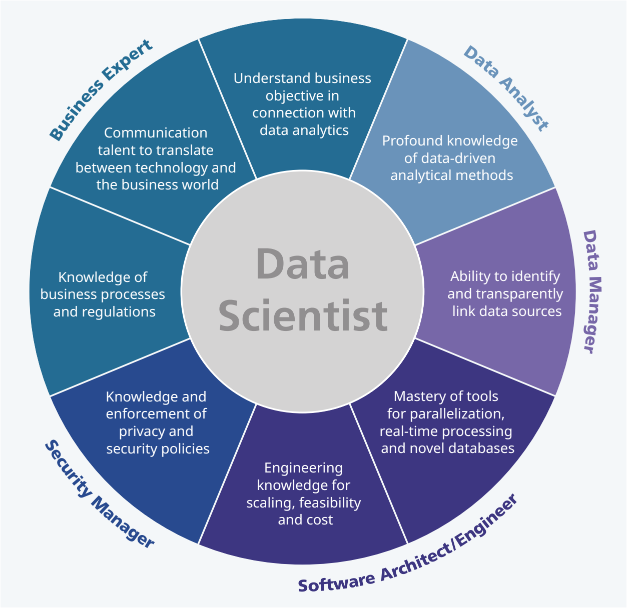

Business Expert
Data Analyst
Data Manager
Software Architect/Engineer
Security Manager
Understand business objective in connection with data analytics
Profound knowledge of data-driven analytical methods
Ability to identify and transparently link data sources
Mastery of tools for parallelization, real-time processing and novel databases
Engineering knowledge for scaling, feasibility and cost
Knowledge and enforcement of privacy and security policies
Knowledge of business processes and regulations
Communication talent to translate between technology and the business world
Data Scientist

Fig. 10.27 The job description of the Data Scientist between data analysis, data protection and customer communication

of innovation of the application and the composition of the team. Because ideally, data scientists are not only familiar with the latest software for Big Data systems and intelligent data analysis, but also understand the respective industry. In fact, good graduates of the relevant courses are often recruited directly by companies after graduation, without having to apply themselves. Given the current demand, it is therefore no easy task for HR departments to assemble a powerful, creative data science team. There is also the challenge that data science teams must be composed of various specialists with different strengths in order to be successful in the long term - because there are only a few "all-rounders".

The job description of the Data Scientist can be roughly divided into four subareas (Fig. 10.27):

1. The Data Manager knows the database systems and other digital data sources such as images, texts or sound recordings, and ensures the necessary metadata as well as a consistently high level of data quality.
2. The Business Expert must identify the potential the available data. He knows the goals of the company, the market and the economic significance of the data. Through his expertise, he evaluates the patterns found and transfers them into possible services.
3. The data analyst must use statistical know-how to uncover relevant correlations in the data and use Machine Learning methods to train prediction models or DNN.

4. When it comes to providing the IT infrastructure or handling large amounts of data from different sources, a software architect / engineer is needed. Data privacy and security must be considered for all these applications. An additional security manager can ensure that this is done—by design—right from the start.

To become a Data Scientist, you should ideally have already acquired some key competencies during your studies. Deep statistical knowledge and sufficient programming skills are mandatory for a Data Scientist. In fact, it is a methodologically large step from analyzing an Excel spreadsheet to examining data sets in the gigabyte range with a large number of variables. Analyzing unstructured data sets, such as language, text, and images, requires specialized deep Machine Learning methods. A wealth of experience helps to narrow down the best possible analytical methods at an early stage and to produce robust results. The more projects, methods and pitfalls a Data Scientist has experienced, the more effectively he or she will be able to carry out future projects.

It is interesting to note that the job description of the data scientist was not created in an university environment, but rather from an economic interest. In the meantime, numerous universities offer data science degree programs and modules. But this also means that current data scientists include a mix of people with different academic backgrounds. From computer scientists to mathematicians and physicists to engineers, many different groups can be found. A degree in natural sciences is certainly a positive factor.

With its world-renowned engineers, Germany develops the best sensors and machines. Meanwhile, though, whoever has more data can build more intelligent devices and the AI services behind them. Those who harness their data can make well-informed decisions, optimize processes and thus save costs. Data scientists are the new engineers in German industry. Universities that offer a data science degree program will represent a locational advantage for their region. But even if many Anglo-American universities offer master's degree programs for data scientists, and some German universities have now followed suit, this is not enough. Existing personnel in the companies must be further trained for the new methods, tools and platforms, and industry experts must be given the vocabulary and understanding to be able to work in teams with Data Scientists.

10.3 Challenges for the Society

New technological capabilities of AI can be used for both positive as well as negative purposes. In its national AI strategy, "AI made in Germany" (BMBF 2018) focused on the benefits for people and the environment. Whether this succeeds will depend heavily on the extent to which Germany and Europe can master and successfully implement AI technologies themselves. Only those who build their own products, platforms and services will be able to implement their values and constitutional traditions.

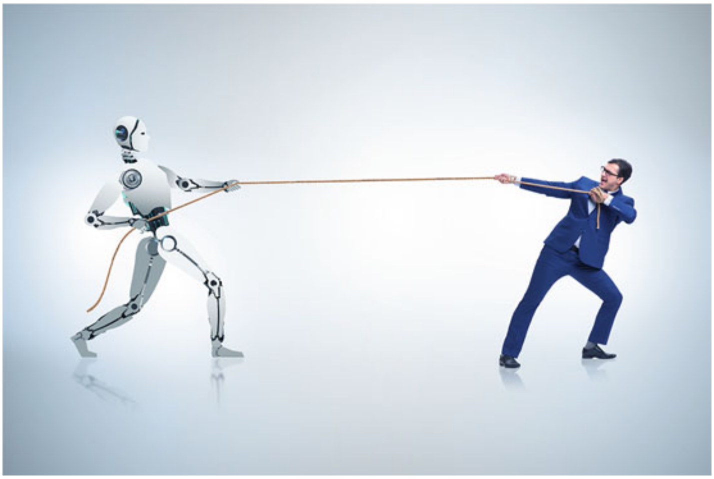


Fig. 10.28 AI technology can have many positive effects, e.g. in medicine. However, many people see DNN and robotics as a potential competitor to take their jobs. Image credits in Appendix A.3

There must be an open social discussion on the use of AI. Such a far-reaching and disruptive technology is a challenge for society as a whole, which should be addressed not only by computer science but also by various disciplines, such as law, ethics and the application research. In order to increase the acceptance of AI by society as a whole, we must work together to find a good middle ground for the current and future use of AI technologies. This process must take equal account of the needs of companies on the one hand and citizens on the other—it must therefore proceed neither too slow nor too fast and must be designed with care (Fig. 10.28).

AI will relieve many people, but should not take away their responsibility for important decisions. Even with well-trained AI systems, errors will still occur in the future and incorrect suggestions will be made. We are a long way from infallibility, because ultimately AI is still statistics based on data. We can see that AI will not replace us in the long run, but it is a new device at our disposal. A new tool.

We are accustomed to being supported by beasts of burden, then steam engines, and later electrical machines support, which relieve us of physical labor. With the computer, we have entered an age in which machines also relieve us mentally. What is new is that computers now learn quasi autonomously and no longer just process predefined programs. This allows them to take on new intellectual tasks, but also makes them more difficult to control. For example, the development of language processing models has been so rapid recently that we don't even know yet what the range of applications will be. AI can tell stories and write short editorial pieces, and there are certainly completely different areas of application for which such technologies could be considered.

It is to be expected that there will be completely new scenarios of human-machine interactions, which will require humans to continuously engage with new media technology. A constant balancing act between an expansion of our capabilities and senses, the constant pressure of optimizing ourselves, and a possible loss of decision-making authority.

10.3.1 Challenges of AI in Medicine

Much of what still sounds futuristic from the point of view of doctors and patients is already technically feasible from the perspective of AI research. But it is especially in the medical context that there is particular skepticism about the innovations mentioned previously. The physician, as a trusted person and mediator, plays an important role in the introduction of technological innovations. A "smart hospital" must specifically invest in education and mediation to communicate the new possibilities and limitations of AI to doctors and patients. AI can recognize symptoms based on patterns and suggest diagnoses, but it cannot yet provide treatment and intervention to patients. It remains the responsibility of doctors to weigh hypotheses and alternatives for and with patients and to initiate therapies. An AI cannot take a Hippocratic oath; this can only be done by the doctor, who makes a decision according to his ethical and moral principles. However, this does not dispense an AI from making its suggestions comprehensible in order to convey confidence in the application.

In order for a functioning healthcare ecosystem to be established, legal framework conditions in particular must be clarified. Today's AI systems need data of high quality and in large quantities in order to recognize patterns and make predictions with high accuracy. Medical data is particularly sensitive, and the demands for data protection and anonymization are highest here. A legitimate personal interest in protecting health data must be legally harmonized with the social interests of improved medicine.

A possible conflict between the interest groups of health insurance companies, hospitals, physicians in private practice and patients must be resolved by finding a data infrastructure that protects privacy (Fig. 10.29). Ideally, the parameters of trained Machine Learning models should encode general relationships rather than facts about specific training examples (e.g., individuals). To ensure this and provide strong privacy guarantees, especially when the training data is sensitive, it is possible to apply techniques based on the theory of Privacy Preserving Machine Learning. In particular, when training with user data, these techniques provide mathematical guarantees that models will not learn or remember details about a particular user. For DNN, the additional guarantees can be useful to reinforce the protection provided by privacy preserving techniques (Radebaugh and Erlingsson 2019), whether through established ones, such as set thresholds, or by federated learning.

Fig. 10.29 In the medical field, it is particulary important to precisely control the access authorizations of the involved entities to personal information. Image credits in Appendix A.3

10.3.2 Orwell's 1984 Vers. 2.0: AI as a Surveillance Tool

There were more than 380 million surveillance cameras in China in 2020 (Keegan 2020). Many of them can already recognize faces and also match them (Sect. 5.10). At a public restroom, people must have their faces scanned by a camera before they can get toilet paper (Dorloff 2018). People who want paper more than once are rejected. Facial recognition cameras prevent unauthorized people from entering a student dormitory. In the city of Jinan, monitors show the names of pedestrians who cross at red traffic lights. Customers can even use facial recognition to pay their bills, such as at Kentucky Fried Chicken in Hangzhou (García-Ajofrín 2018). It's almost impossible to pay with traditional cash in major Chinese cities, and even for entering and exiting a parking garage you have to identify yourself with facial recognition or a special app for billing. Meanwhile, in China, applying for Internet access or a mobile phone number is only permitted if the face has been scanned beforehand. In many countries, you have to identify yourself for a mobile phone contract, but a face scan is a novelty worldwide (Deuber 2019). The transparent digital citizen is almost a reality in China.

In addition to fighting crime, the Chinese state uses the data for a huge database with information on social good behavior. Every citizen is given a points account in which good and bad ratings are automatically stored. All activities in which people are observed in their daily routine can be included in the rating, the "Social Credit"(Mistreanu 2018). The rating result is not secret, but the citizens with the best rating are announced on large display boards (Fig. 10.30). Well-rated fellow citizens receive perks such as discounts on heating bills or favorable bank loans. And there is already a group of people who are penalized because of the system: Nearly 10 million people have been barred from buying tickets for express trains or airplanes. By 2020, China plans to roll out the system across the country. At its heart, however, remain the surveillance cameras, which will be networked and massively expanded in the coming years. The most valuable start-up at present, SenseTime,

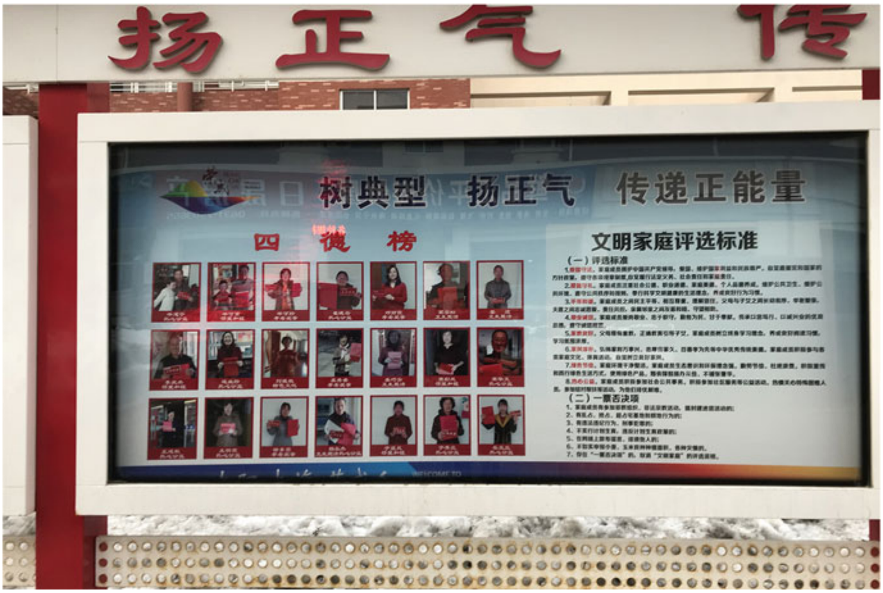


Fig. 10.30 Evaluation of the top scorers in the Chinese social scoring system on public display boards (Mistreanu 2018). Image credits in Appendix A.3

comes from China and is active in the field of facial recognition. Approximately $600 million have been raised in investment money (Bastian 2018). The market in China seems to be huge in this application context in the future as well.

A close network of the large Chinese BAT corporations with the country's political leadership seems to be in place and ensures a permanent exchange of data between the private sector and state authorities. According to this, AI provides new ways for the very few to oppress the very many. AI "can" enable the perfect surveillance state and make it very effective. This is a threat to a free society. China is moving to put its population under permanent camera surveillance. It is the vision of George Orwell that is now becoming a reality.

A new potential threat to privacy is lip-reading technology (Sect. 7.4). The available publications specify a 40% word error, but it can be assumed that security authorities already have much more sophisticated techniques. Apparently, it is now possible to optically "eavesdrop" on people at a great distance who would normally be safe from surveillance. At Bundesliga soccer matches, you often see coaches passing on their instructions with a hand covering their mouth so that the opponent can't decipher them.

However, surveillance of public space takes place is not only taking place in China, but also in other countries. For example, there are about 500,000 video cameras in Greater London, which are used to monitor streets and squares. Furthermore, it is known that the German secret service BND collected 220 million metadata (sender, recipient, ...) from SMS, email, WhatsApp, etc. every day and

Fig. 10.31 Surveillance cameras at the car yard in Thuringia on the A9. Image credits in Appendix A.3

partly passed them on to the American CIA (Biermann 2015). According to the law, there are no restrictions on the surveillance of foreigners by the BND. With deep neural networks for text analysis (Chap. 6), content (e.g., from emails or telephone conversations) can be analyzed much more precisely today than in the past.

However, intensive monitoring of citizens is also taking place by Internet companies. On every smartphone, most users call up a large number of websites and apps that analyze the user's input, location, address books and purchasing and communication behavior and use this, among other things, to serve up targeted advertising. By linking the results of different apps via advertising IDs and trackers, users' daily routines can be consistently reconstructed and evaluated for different purposes (Valentino-deVries 2018). Additionally, more and more biometric data is collected by apps in the cloud, for example photos of the user and measurements of fitness trackers, the use is often not clear. Although the General Data Protection Regulation (GDPR) in Europe requires user consent, the vast majority of users give their consent with a click without knowing the consequences. All these data can be analyzed with Artificial Intelligence methods and utilized to predict a wide variety of characteristics (Fig. 10.31).

For a study, (Kosinski et al. 2013) examined the information content of easily accessible characteristics of users, e.g., Facebook Likes. They showed that it is possible to use a simple logistic real-time regression (Sect. 3.4) to predict very private characteristics of users, such as whether users are gay (88% accuracy), whether they are black Americans (95% accuracy), or whether they vote Republican or Democrat (85% accuracy).

This method was leveraged by the data analytics company Cambridge Analytica evaluating data of some 87 million potential voters on a large scale and mostly without user consent (Cadwalladr and Graham-Harrison 2018). Based on predicted features, individually tailored messages were sent to users to influence voter behavior (microtargeting). Cambridge Analytica supported Donald Trump in the 2016 election campaign, the pro-Brexit campaign of Boris Johnson and was active in more than 200 elections around the world (Dachwitz et al. 2018).

These incidents show the capability of the major Internet companies to gather knowledge about their users and apply it for a variety of purposes with the help of advanced Machine Learning. Peter Eckersley, senior computer scientist at the Electronic Frontier Foundation (EFF), says: *"It's Facebook's business model to accumulate as much direct and indirect data about users as possible and then determine who is given access to it and at what price. When you use Facebook, you hand over records of everything you do to the company. I think people have reasons to be concerned about that."* (Tischbein 2016)

10.3.3 War of the Machines

> *"Being killed by a machine is the ultimate human indignity."*
>
> Robert H. Latiff

ALPHA is the name of the program that caught the attention of the military worldwide in 2016 (Bate 2016). The software ran on a single-board Raspberry Pi computer: 64-bit ARM processor, four cores at 1.2 gigahertz, 1 gigabyte of RAM, hardware cost less than 35 Euros. Every modern mobile device has a higher performance. This software was sent into a virtual air duel with an experienced fighter pilot in a simulation model. Despite the meager hardware, the success of the software was frightening. In hours of hand-to-hand dogfights, former U.S. Air Force fighter pilot and instructor Gene Lee failed to defeat his ALPHA opponent a single time. Afterward, Lee said, it was the most aggressive, responsive, dynamic and convincing AI system he had yet seen. Depending on the need and situation, the AI system switched to defensive or offensive behavior and proved incredibly responsive.

If this continues, AI will be part of the flight crew in the future or replace it completely. After all, the pilot, with his maximum tolerable G-forces, is merely an obstacle that unnecessarily limits the aircraft's maneuverability and acceleration. Fighter aircraft without pilots could fly more maneuverably and could be produced smaller and perhaps more cost-effectively.

A trend is emerging here: autonomous units such as drones will soon be used not only for border security, but also in armed confrontations. This will readjust the global balance of the military. In the days of the Cold War, the number of soldiers under arms and the number of tanks, frigates, etc. were often added up—but this is no longer a yardstick for the performance of an army. Drones and other autonomous units are making AI a key innovation in the military. Fear or fatigue will play no role for these combat units. Combat operations of the future will be more heavily automated than ever before.

Back in 2016, the U.S. military demonstrated the successful use of a swarm of drones. Each drone flew fully autonomously, but coordinated with the other drones as a team. The drones had a mission to patrol an area of about three square kilometers. Three Navy fighter aircraft were detected entering the area and were

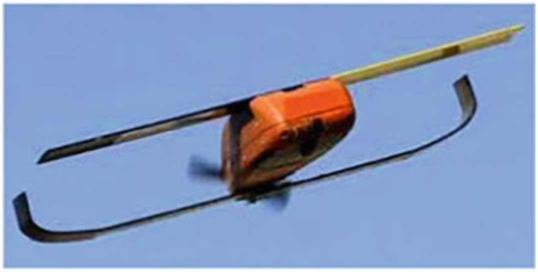


Fig. 10.32 The Pentagon's Perdix combat drone on a local training mission. Similar drones of this type can coordinate their actions in a swarm. Image credits in Appendix A.3

encircled by the drone team. "Perdix" is the name given to this development by the military, which is certainly only a prelude to further developments (Slavin 2017) (Fig. 10.32).

There is a need for social action here, because robots have no ethical scruples and no conscience to show them limits in combat. The construction of drones that decide autonomously on their use of weapons must be avoided. Standards for the use of autonomous warfare devices do not yet exist. The efforts of many states to ban such autonomous weapon systems have so far failed due to the resistance of countries with large arms industries (KillerRobots 2018) It is now up to civil society to demand globally binding standards and bans for autonomous weapon systems and corresponding agreements under international law from their governments.

One focus of the future security threat of our time are cyberattacks. These are AI systems that autonomously search for vulnerabilities in the infrastructure, control of financial flows and data management of foreign states in order to eliminate them or obtain important information in the event of a military conflict. The Stuxnet malware program from 2010 is a prominent example. Stuxnet was used to sabotage the control of power frequency converters in order to specifically damage motors of waterworks and pipelines. In 2010, the virus was used to disrupt Iran's nuclear program; the contractor was never publicly reported. Nation-states are building cyber units to prevent such attacks. Compared to traditional investments in fighter aircraft, tanks, and aircraft carriers, they are less expensive but can still be very effective (Biermann 2019). The wiretapping practices of the U.S. NSA and the revelations of Edward Snowden show that no borders exist even among friendly states.

10.3.4 Artificial General Intelligence

In Artificial Intelligence, a distinction is made between strong and weak AI. Weak Artificial Intelligence focuses on systems that solve concrete application problems, such as those presented as examples in Sect. 2.10. Numerous breakthroughs have been made in this area in recent years. In doing so, AI has often been able to match or even surpass human capabilities in the respective special task. All systems existing so far can be considered as weak AI. They do not develop a deeper understanding of the problem solution or its causal relationships.

Strong Artificial Intelligence or Artificial General Intelligence (AGI) on the other hand, is designed to cope with problems in situations and tasks that are not defined in advance. Its role model is the human being, who can master situations that are unfamiliar to him, even when little or no experience is available. In contrast to current AI systems, humans learn very efficiently and often only require a small amount of information from their environment to do so. To date, AI systems have struggled to learn from individual cases and need a large amount of training examples. If a powerful artificial intelligence were to become available, it would automatically collect data on a large scale, possibly improve itself rapidly, and could trigger unforeseeable changes. This point in time is called a technological singularity in futurology (Fig. 10.33).

The goal of Artificial General Intelligence is to achieve the same intellectual skills as humans or even surpass them. It combines logical thinking, planning, learning, communication in natural language and the ability to make decisions even in the face of uncertainty in order to achieve self-set goals. It acts and learns out of a motivation of its own. This constant drive for novelty has turned us humans into explorers and discoverers.

Whether an AI can ever acquire its own consciousness, develop empathy for humans or another AI system, and whether it can have the capacity for reflection, moral and ethical thinking, is completely open. In Sect. 10.4.2, we will discuss this question from a technological perspective. Such a so-called "superintelligence", which is driven only by the most perfect possible achievement of goals and disregards fundamental humanistic values, could become a danger for us. Diederich (2021) describes the psychological consequences for the well-being of individuals as well as the significant impact on societies. Personalities like Elon Musk therefore warn against such a development. Initiatives such as OpenAI (Metz 2016), which he supports, are working on freely accessible Artificial General Intelligence systems that should benefit society and not harm it. It is questionable, however, what such projects will achieve as ultimately any form of technology can also be misused.

Fig. 10.33 The availability of Artificial General Intelligence is also referred to as a technological singularity because AI systems could then potentially become self-determining. Image credits in Appendix A.3

As philosopher and risk researcher Nick Bostrom said in 2016, *"If you have a button that can do harm to the world, you wouldn't make it available to everyone."* Indeed, in February 2019, OpenAI did not release the full version of its latest text generation system, GPT-2 (Sect. 9.2.3), because of concerns that it could be misused for automated fake news and spam comments on social media (Whittaker 2019).

Currently, we are still far away from an Artificial General Intelligence. Researchers assume that the first breakthroughs can be expected in 20–40 years. However, similar time frames have been given before. For example, Herbert Simon said in 1965, "Machines will be capable of doing any job that humans can do within the next twenty years." Other researchers claim an AGI will never exist. Progress in AI is incredibly difficult to predict, and it cannot be excluded that an Artificial General Intelligence will eventually be developed. While there is certainly a long way to go, we are moving a small step in that direction with each passing day.

10.4 Methodological Challenges

Meanwhile, deep learning is by far the most successful approach in Artificial Intelligence. In recent years, many AI problems have been successfully tackled in this way for the first time (Fig. 10.34). Since then, there has been much discussion about AI in the media and social networks. Some AI enthusiasts believe in rapid progress in AI. Andrew Ng, a leading AI scientist, says *"if a typical human can do a mental task in less than a second, we can probably automate it either now or in the near future with the help of AI."* (Ng 2016). Shane Legg, the chief scientist of Google's DeepMind group, said *"human-level AI will be achieved in the mid-2020s"* (Mitchell 2018).

But many scientists also see problems with the advancement of AI. Gary Marcus, professor of natural and Artificial Intelligence, asks, *"Is deep learning approaching a brick wall?"* (Marcus 2018). And computer science professor Melanie Mitchell worries, *"Artificial Intelligence is reaching a frontier of meaning."* (Mitchell 2018). In this section, we will discuss some methodological challenges of AI.

Fig. 10.34 Artificial Intelligence methods have had great success in specialized applications in recent years. However, there are still major deficits in a number of overarching tasks. Image credits in Appendix A.3

10.4.1 Combination of Data and Uncertain Reasoning

The DNNs presented in this book are based on the fact that non-explicitly formulated relationships between features are automatically extracted from observed data and stored by adjusting the model parameters. Hence only problems with available data can be solved, so that the relationships between the relevant features can be learned.

However, there is often additional information that can be used to solve problems. This knowledge can be extracted in two ways. We have already presented systems for extracting world knowledge by transfer learning (Sect. 6.7). Here, among other things, a BERT model was pre-trained with a large training corpus. It was then fine-tuned on an additional dataset to enable it to answer previously unknown content questions.

An alternative approach is the use of explicitly formulated knowledge or symbolic Artificial Intelligence. Since the 1980s, so-called knowledge bases have been built up in which knowledge is described in the form of facts and rules. One example is Cyc, which contains more than 24.5 million manually entered rules and facts (Sect. 6.7.3). In Cyc, one can formulate rules that apply to all objects of a type, e.g., *"all birds have feathers"*. Subsequently a module in Cyc can derive conclusions such as *"a sparrow has feathers and two legs"*. Such inferences are particularly important because they are true with certainty, unlike many deep neural network results, which are only correct with some probability. There are other knowledge bases, e.g. Wikidata, which contains knowledge from Wikipedia in the form of facts and rules, e.g. bornIn(Friedrich_Schiller, Marbach) (Erxleben et al. 2014). In the field of medicine (Fig. 10.35), there is the Medical Subject Headings (MeSH knowledge base), for which medical facts and rules have been extracted from about 30 million medical articles (Richter and Austin 2012). One can use it, for example, to search for articles on the effects of a particular drug. A recent overview of symbolic AI techniques together with Machine Learning is provided by Görz et al. (2020).

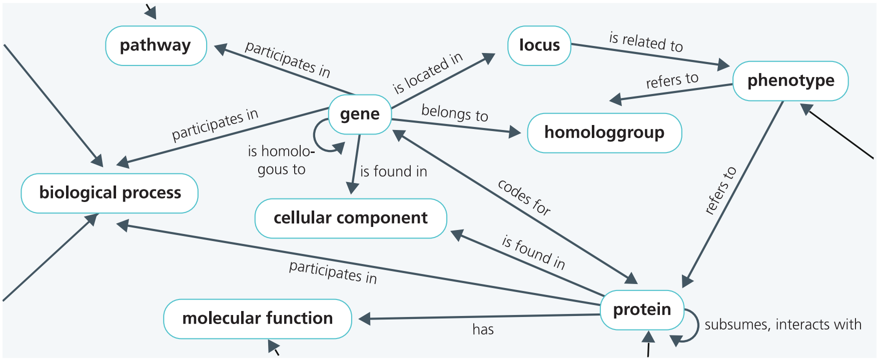


Fig. 10.35 Excerpt from a biological Knowledge Base. Image credits in Appendix A.3

If we disregard definitional relations (“a lion is a mammal”), it turns out in practice that rules usually do not hold with certainty, but only up to a certain degree. For this reason, facts and rules can be given a measure of their plausibility. For example, `personGetsSick(heavy smoker, lung cancer)` has a probability of 0.244, and the probability of `personSurvives(lung cancer, 5 years)` is 0.15. Combining such uncertain facts and rules, one can draw conclusions from them, which themselves have a probability. Such inference procedures are complex because the formulated rules are usually not statistically independent. Such a kind of inference is called uncertain reasoning.

To solve problems where information from data (e.g., sensor data) and knowledge bases must be combined, one needs methods that link both types of information in a joint learning algorithm. This type of AI is also referred to as hybrid Artificial Intelligence because it combines different types of information and algorithms. Fischer et al. (2019), for example, translate each uncertain rule into a loss function term, which takes the value 0 if the rule is satisfied. In addition, the rule is also given a weight corresponding to its plausibility. By this procedure, one can enforce the validity of the logical relations for the model and also search for inputs that satisfy certain logical conditions. Manhaeve et al. (2018) define a programming language that combines probabilistic reasoning with DNN. This allows rules that hold with a certain probability to be linked to neural networks and trained with the help of data. Finally, there are extensions for the TensorFlow programming environment that can combine neural networks and uncertain rules (TensorProb 2019).

Although initial approaches exist, the theory and practical tools for integrating uncertain reasoning and DNN still need to be greatly improved so that these methods can be used in real-world applications. It is not yet clear whether hybrid AI can successfully support transfer learning.

10.4.2 Fast and Slow Thinking

Intelligent thinking takes place at different speeds. Nobel laureate Daniel Kahneman has developed a hypothesis (Kahneman 2011) about two different systems of thinking based on long studies of human behavior (Fig. 10.36). System 1 (Fast Thinking) is fast, instinctive, and emotional. Examples include understanding a simple spoken sentence, driving a car on an empty road, or recognizing a familiar object in a picture. System 1 runs continuously and generates impressions, intuitions, and quick judgments based on our immediate perceptions.

System 2 (Slow Thinking) is slower, more deliberate and more logical. It is responsible, for example, for remembering a person we have not seen for a long time, parking in a narrow parking space, or solving the arithmetic problem 16×34. System 2 is only used when problems arise with system 1, i.e. it cannot well explain the observations.

In the human brain system 2 corresponds to a working memory with limited capacity (Diamond 2013). It enables short-term storage and simultaneous

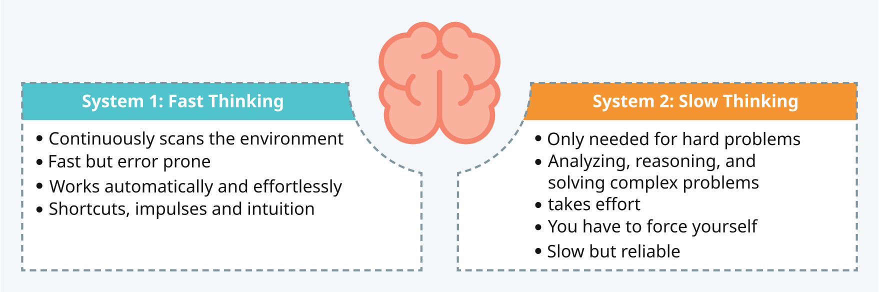

System 1: Fast Thinking
• Continuously scans the environment
• Fast but error prone
• Works automatically and effortlessly
• Shortcuts, impulses and intuition
System 2: Slow Thinking
• Only needed for hard problems
• Analyzing, reasoning, and solving complex problems
• takes effort
• You have to force yourself
• Slow but reliable

Fig. 10.36 The properties of the two systems for Fast and Slow Thinking in the human brain according to (Kahneman 2011)

manipulation of thought content. It apparently has an important role in problem solving and logical reasoning. The number of information units that can be handled simultaneously is estimated to be between five and seven. Humans are aware of the thought processes of system 2, while processing in system 1 takes place largely subconsciously.

Another important mechanism of system 2 is attention. Here, the limited resources of deliberate thinking are focused on content available in other areas of the brain. These can be perceptions of the environment, of one's own actions, but also thoughts and feelings. On the other hand, this means that other stimuli are often not perceived in the state of attention. Bengio (2019) points out that attention in contemporary DNN can be realized by attention mechanisms (Sect. 6.5.2), which indeed focus computations on specific contents.

Pearl (2018) argues that there are different types of problem setups for learning algorithms:

- Associative: What do the observed skin spots tell about the presence of skin cancer? Only statistical associations in the data are analyzed here.
- Change in Situation: How will sales change if we double the price? This type of question cannot be answered from historical data alone, because here we must take into account a reaction of the customer who is responding to the higher price, e.g. by a simulation model.
- Counterfactual reasoning: Would John F. Kennedy still be alive if Harvey Oswald had not shot him? The aim here is to find relationships which are in contrast to what actually happened (Fig. 10.37).

For the last two types of questions, one needs models of the application domain with which one can simulate alternatives, e.g. the working memory mentioned previously. It is important here to distinguish causes from effects and to uncover the causality of the relationships.

Most approaches to deep learning do not cover such questions. Reinforcement learning models of an alternative (Chap. 8). They require a model of the environment, i.e., a simulation model, which is characterized by states and responds to

Fig. 10.37 Counterfactual reasoning is an important approach to solving complex problems. Image credits in Appendix A.3

actions of the user. This environment model can then be used to learn a strategy that is highly rewarding. However, the use of reinforcement learning for counterfactual reasoning, for example, is still in its early stages.

Many applications of deep neural networks cover the area of fast thinking: recognizing an object in an image, capturing the meaning of spoken words, or controlling motion sequences such as walking or cycling. They run largely associatively. Complex reasoning such as counterfactual reasoning belongs to Slow Thinking. Solving such tasks also means a noticeable effort for the human brain.

Jack Nicklaus was one of the greatest professional golfers of all time. In his autobiography (Nicklaus 2007, p. 7), he describes how he prepares for a tee shot (Fig. 10.38): *"I never hit a shot, not even in practice, without having a very sharp, in-focus picture of it in my head. It's like a color movie. First I 'see' where I want it to finish, nice and white and sitting up high on the bright green grass. Then the scene quickly changes and I 'see' the ball going there: its path, trajectory, and shape, even its behavior on landing. Then there is this sort of fadeout, and the next scene shows me making the kind of swing that will turn the previous images to reality."* So Nicklaus sees the progression of the shot and the trajectory of the ball in chronological order. He can use it to plan his shot to achieve the desired goal.

Hamrick (2019) discusses the suitability of models of reinforcement learning for problem solving of this type and compares them to mental simulations in humans. One difficulty is that currently in reinforcement learning, a new simulation model of the environment must be trained or programmed for each problem. In contrast, humans are able to perform such simulations ad hoc, even if their accuracy often is limited. While reinforcement learning is promising for building flexible, robust models for constrained problems, much work remains to be done before human imaginative capabilities are achieved. To match human skills, models must be built from individual submodels and assembled as needed. Strategy search methods must succeed with only a handful of evaluations of noisy, incomplete models; and models

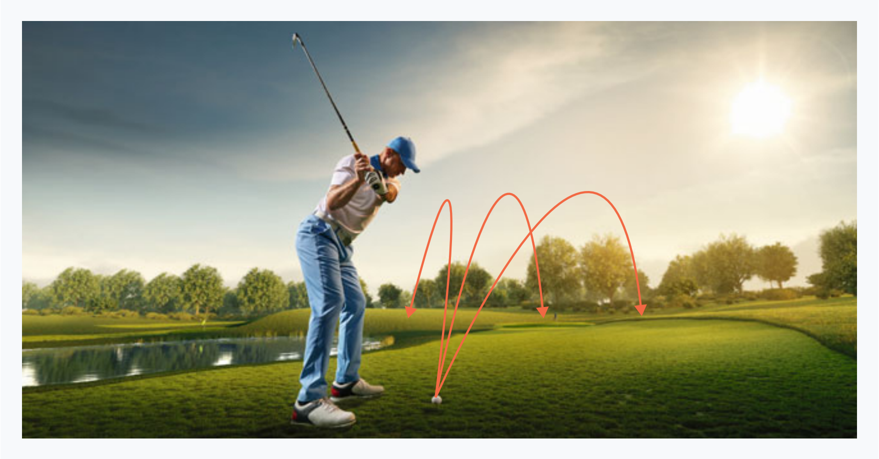


Fig. 10.38 Possible trajectories of the golf ball when teeing off. Image credits in Appendix A.3

must be able to generalize to situations far outside their training data to support creative exploration and better understanding of the environment.

According to Bengio (2019), the tasks of Kahnemann's System 1 can already be performed quite well by existing DNNs. In his view, three important capabilities of System 2 are currently missing:

- The transfer of trained contexts to new areas. First approaches to this exist in transfer learning (Sect. 5.7).
- The manipulation of abstract semantic concepts. In this process, concepts must be anchored to sensor perceptions in the world by system 1 (symbol grounding) and content must be represented by distributed representations. It is important to consider uncertainty in this process.
- Agents must be considered that can move and act in an environment. They have to collect world knowledge, relate the dependencies between sensor contents, objects of the environment and actions and furthermore actively search for new knowledge.

These capabilities are currently not yet available.

Jordan (2018) points out a number of problems with current AI: *"These problems include the need to bring meaning and reasoning into systems that perform natural language processing, the need to infer and represent causality, the need to develop computationally-tractable representations of uncertainty and the need to develop systems that formulate and pursue long-term goals. These are classical goals in human-imitative AI, but in the current hubbub over the 'AI revolution,' it is easy to forget that they are not yet solved."*

10.5 Building Trust in AI

Cathy O'Neill is an American mathematician who worked in finance at hedge fund firms for several years after graduation. Disillusioned with her work, she moved to Columbia University and wrote a remarkable book titled "Weapons of Math Destruction" about the social impact of algorithms (O'Neil 2016). In the book, she explains how algorithms can be used to reinforce and further expand existing inequalities (Fig. 10.39).

Automated decisions based on Machine Learning trained with big data pose a whole new set of challenges for our society. We want a common good, human-centric AI that is in line with our values, works reliably, and is competitive in the process. We must therefore try to achieve fair and objective results and avoid undesirable consequences. Fair behavior of an AI application towards all stakeholders, consideration of the needs of users, reliable, understandable and safe functioning as well as the protection of sensitive data are key requirements for the trustworthy use of an AI. Lack of awareness of ethical and legal issues and the risks of dual use can lead to the opposite of what is intended. In this context, the possibility of automated decisions by an AI sheds new light on the individual liability and responsibility of people. In order to make AI systems humane and trustworthy, there needs to be a social understanding of what guardrails, principles, and boundaries should be applied, how AI applications should be developed, how autonomously they should be allowed to act, and how we want to control them.

Fig. 10.39 Cathy O'Neill is an American computer scientist who criticizes the use of AI in business. Image credits in Appendix A.3

At a 2018 presentation, the Google Duplex voice assistant (Sect. 7.5.2) behaved deceptively human on the phone (Leviathan and Matias 2018). Google Duplex is supposed to take over the often tiresome phone calls for making appointments with a hairdresser or restaurant. However, the areas of application are much broader, as call centers are naturally also very interested in this technology. This is where a transparency problem arises. In principle, voice assistants on the phone should say right at the beginning that they are an automaton and on what behalf they are acting. Then they should give us the option of being connected to a human instead. Preferably with an estimate of the waiting time. Moreover, such a redirection should be possible at any time during the dialog. In general, it is indispensable that AI systems create transparency right from the start and that it remains absolutely comprehensible for humans when and how they communicate with a machine and how they can control the interaction. Certainly, it will be very exciting when no human is involved at all, but two voice assistants meet directly. Such situations are not out of the question in the future and must of course be considered and clarified at an early stage. When people delegate decisions to machines, who is responsible?

On 16 February 2017, the EU Parliament has demanded by resolution civil law regulations for robots (Parliament 201). In terms of liability issues, they should be given the status of an electronic person. In fact, civil law has gaps for robots and for digital agents that can make decisions under uncertainty. Uncertainty means that they can decide between alternatives by optimizing multiple criteria without their behavior being programmed in a predictable way. This is true, for example, of the voice assistants mentioned previously, which act as quasi-digital proxies on the phone with a human or another digital agent to make appointments, make bookings, or sign contracts. Under current law, the user is responsible for any damage caused by a robot or machine. Under the EU Parliament's current thinking, this is no longer considered appropriate if the machine escapes the influence of its owner due to independently learned behavior patterns. In its resolution, the European Parliament assumes that self-learning systems and autonomous robots can no longer be considered tools of a manufacturer, operator or user. In particular, the binding nature of agreements, contractual liability and liability in the event of damage must be reconsidered and redefined here. One idea could be to introduce a special liability insurance for robots (e-persons) whose prices are calculated depending on the level of autonomy and the risk of the application.

The question of how AI applications can be used responsibly and reliably has been the subject of intense social and scientific discussions in the international arena for some time now. It impresses positively impressive how many AI guidelines have been developed by states, NGOs and companies in recent years. The most important ones are shown in Fig. 10.40.

At the European level, the EU Commission has established a High-Level Expert Group on Artificial Intelligence. In April 2019, this group formulated recommendations for assessing the trustworthiness of AI applications (Expert-Group 2019). An important concern is to avoid discriminatory actions and information asymmetries between humans and the AI algorithm and to consider these directly in the design of the AI system (ethics-by-design). Starting from general ethical and legal guidelines,

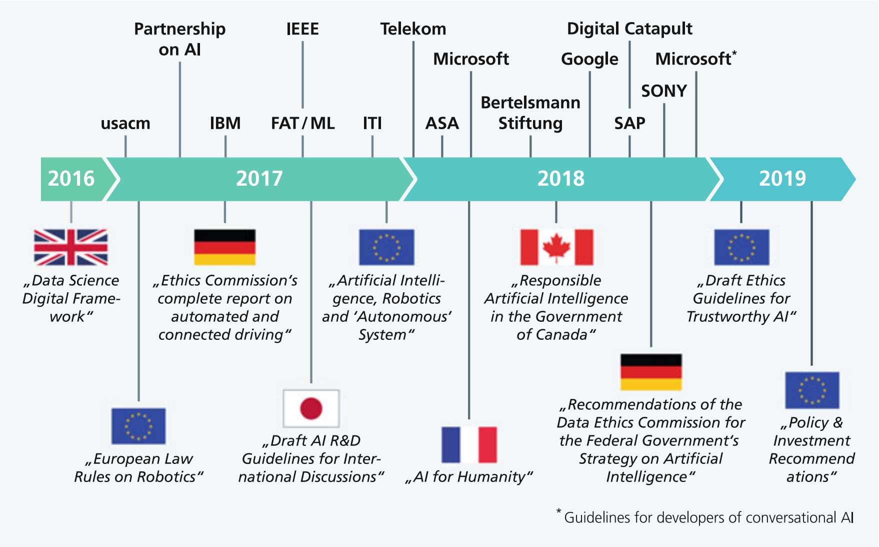

Partnership on AI
IEEE
Telekom
Digital Catapult
Microsoft
Google
Microsoft*
SONY
Bertelsmann Stiftung
usacm
IBM
FAT / ML
ITI
ASA
SAP
2016
2017
2018
2019
„Data Science Digital Framework“
„Ethics Commission's complete report on automated and connected driving“
„Artificial Intelligence, Robotics and 'Autonomous' System“
„Responsible Artificial Intelligence in the Government of Canada“
„Draft Ethics Guidelines for Trustworthy AI“
„European Law Rules on Robotics“
„Draft AI R&D Guidelines for International Discussions“
„AI for Humanity“
„Recommendations of the Data Ethics Commission for the Federal Government's Strategy on Artificial Intelligence“
„Policy & Investment Recommendations“
* Guidelines for developers of conversational AI

Fig. 10.40 Various AI guidelines presented by governments, organizations, and companies in recent years

requirements and implementation methods are suggested. The document ends with a list of 60 concrete questions to that are intended to evaluate the development, implementation and use of AI systems. However, the questions are still far from being adequate for a practical deployment, for example as a test catalog or a standard. They make clear how challenging it is to translate our values and ethical principles into technology in a verifiable way.

10.5.1 How to Build Trustworthy AI Systems?

We all know the TÜV sticker on cars, which must be renewed every two years. It is clear to every car owner that he cannot drive to an accredited inspector for an inspection without functioning lights, indicators and engine. There are still some technical challenges to be solved before we can think of approving self-driving cars. In this context, it has not yet been decided whether a passenger car model will be inspected by the TÜV only once, or after each software update, as is the case with TESLA. In order to establish trust in these and other AI applications, it must be verifiable that they observe the ethical and legal framework conditions in their specific application context and function safely and sufficiently reliably.

In a certification procedure, one can check and confirm compliance with standardized requirements. "Standardized" means that a test catalog precisely defines the requirements and the criteria of the test, thus giving the seal of approval its

Fig. 10.41 Aspects of certification of Artificial Intelligence systems. Wordcloud from the text of (Cremers et al. 2019)

meaning. For such a catalog, the questions of the EU's High-Level Expert Group certainly provide a good orientation framework. The certification is awarded by an independent authority, such as a TÜV or a Dekra. In this respect, an AI certification serves the general purpose of averting injustice—in the sense of violations of the law—and ethically unjustified consequences from society. With an AI certification carried out in Europe, we can set a quality and value standard "Made in Europe" that places people at the center and creates trust and acceptance among companies as well as users and societal stakeholders (Fig. 10.41).

Conversely, however, it must be avoided that overregulation and excessively high certification standards have an inhibiting effect on innovation or become outdated too quickly due to the dynamics of technological progress and are thus soon no longer appropriate. The methods and application possibilities of Artificial Intelligence are continuously and profoundly evolving that certification must reflect this dynamics. In view of technological innovations and upheavals, our values and laws are also changing, and with them the guard rails for the certification of AI.

10.5.2 How to Test Deep Neural Networks?

There are some serious differences on the technical side between a classical software and an AI system with a model trained on data. A classical software is modular and built from human-readable instructions. This gives the software developer the ability to validate the software largely automatically through a pipeline of component testing, integration testing, and system testing. Artificial neural networks, on the other hand, have millions to billions of parameters that have been optimized using large amounts of data. Any constellation of values of the inputs and parameters can be responsible for a concrete result, and their interactions are usually highly complex. Furthermore, the results can be generated randomly according computed probabilities, as is the case for text created by language models. All these possible combinations are difficult to test. They can also not be taken apart and tested separately according to the "divide and conquer" principle. In traditional software, there are just well-defined control flows; in Machine Learning models, there are data flows.

For example, if you want to predict bone fractures from X-ray images, it may be that one of the hospitals has a different setting for the X-ray equipment, which often incorrectly leads to bone fracture diagnoses. Then the learning procedure may detect this machine in the data and come up with the diagnosis of bone fracture more often for images from this machine. So the algorithm makes good predictions with respect to the training data, but nevertheless causes errors (C. S. Smith 2020).

In fact, imbalanced data is one of the most common causes of an unsatisfactory application. In addition, AI applications must be retrained in good time during operation or even continue to learn on their own on an ongoing basis (reinforcement learning, Chap. 8). After such updates, the behavior of the AI must be tested again. Appropriate processes must be set up for this purpose. It is by no means only data scientists who make mistakes when they build their first AI pilots. Even experienced AI engineers at Amazon, Google, Microsoft, etc. have made mistakes. In hindsight, one wonders if this could have been noticed earlier. But research is responsive and the state of the art in Machine Learning is advancing rapidly. A test catalog must therefore be regularly updated.

A rather mundane source of errors in DNN are disturbances in the acquisition of the data. These can be, for example, measurement errors in the sensors, or discrepancies in the nature of the inputs between the training set and the test set. We have seen that such errors can greatly reduce the performance of DNN (Sects. 5.12.1 and 7.9.1). The problem can be reduced by including inputs with measurement errors into the training set.

Another important area of concern for DNNs are examples that are intentionally constructed to deceive the DNN. A DNN trained to recognize people might find that features such as color, texture, or background patches are just as strong features for person recognition as the items we normally consider relevant, such as mouth or eye shapes. However, this also means that a very small change in input in seemingly "irrelevant" areas can cause the DNN to recognize something different. For example, we saw in Sect. 5.12.2 that traffic signs can be modified by small stickers to misclassify them fundamentally. Similarly, speech recognition systems can be made to completely misinterpret speech input by almost inaudible jamming signals (Sect. 7.9.2). Finally, robotic controls are also susceptible to small manipulations of input (Sect. 8.5.6). Although there are available countermeasures, they are only effective for known attacks.

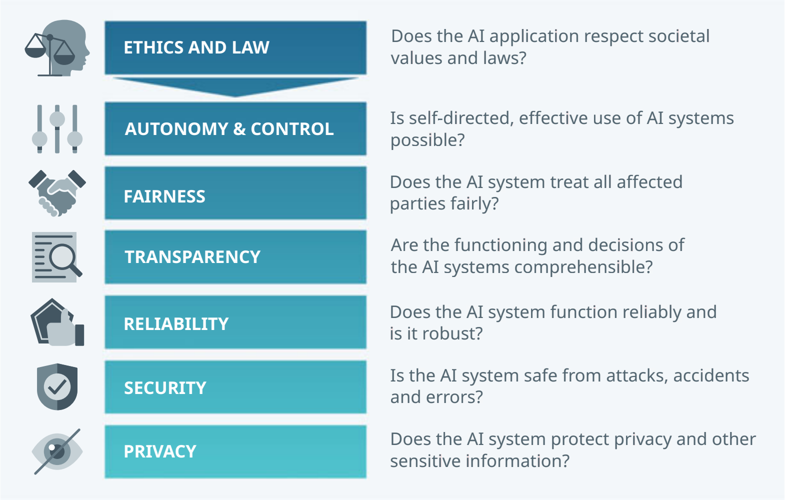

ETHICS AND LAW
Does the AI application respect societal values and laws?
AUTONOMY & CONTROL
Is self-directed, effective use of AI systems possible?
FAIRNESS
Does the AI system treat all affected parties fairly?
TRANSPARENCY
Are the functioning and decisions of the AI systems comprehensible?
RELIABILITY
Does the AI system function reliably and is it robust?
SECURITY
Is the AI system safe from attacks, accidents and errors?
PRIVACY
Does the AI system protect privacy and other sensitive information?

Fig. 10.42 Action areas of the Bonn AI test catalog. For each of these action areas, a series of questions and requirements for an AI system are formulated

In a test catalog, it is necessary to define the action areas from an ethical-legal and technical point of view. For the so-called "Bonn Catalog", which the Initiative KI.NRW has developed, the areas of fairness, transparency, autonomy and control, data protection as well as security and reliability were defined in a white paper in May 2019 to create a basic framework for an AI certification (Cremers et al. 2019). While the "security" item covers the usual aspects of operational security, the "reliability" item refers to the special testing challenges of complex AI models such as deep neural networks. Figure 10.42 shows the action items with their guiding questions for the AI system.

10.5.3 *Is Self-Determined, Effective Use of an AI System Possible?*

An AI system should not restrict the autonomy of individuals and social groups. With this in mind, the design, development, and operation of an AI application must consider or test the extent to which users might develop excessive trust in the application, form emotional attachments (Sect. 9.4), or be unacceptably influenced or guided in their decision-making.

For example, "User Intent Recognition" detects human emotions and intentions in real time and can be directly exploited if necessary. Personalized filtering systems create filter bubbles that can limit the objectivity and criticality of their users by providing biased news and information. AI applications in the medical context do not take a Hippocratic oath. As long as sufficient time is allowed for the physician to make a judgment, AI systems, according to the current understanding, must therefore not decide alone on the treatment of a patient, nor should they lead the physician to uncritically adopt their suggestions. Applications in the financial industry, where bots react to price developments in a matter of seconds, show that there must be safe intervention options in emergencies. Is there an emergency stop for such an AI system, and how does the user recognize an incorrect development?

In time-critical areas of operation, such as autonomous driving (Sect. 8.5), there is often not enough time to involve humans. It is important that an AI system only operates under the conditions for which it has been trained. If unfamiliar situations occur, the overall system must be transitioned to a safe state. Such a case could occur, for example, if a road is covered with snow, but the training data came only from snow-free days. The system must then be able to quickly park the car at the side of the road. So we need to understand the capabilities and limitations of the machine (see Sects. 5.11, 5.12, 6.10, 7.9, 8.5.4). Faultlessness can rarely be guaranteed; even with fully autonomous driving, we need to discuss this issue.

10.5.4 Does the AI System Treat all Affected Parties Fairly?

Discrimination against vulnerable individuals is a risk that must be thought of when using AI. Dynamic pricing and automated assessments can put certain populations at a disadvantage. This doesn't even have to happen explicitly. There are many cases where the underlying data is the cause. This is illustrated by a now prominent example from Amazon. The goal was to automate the evaluation of candidates in recruiting and ensure consistent criteria. Unfortunately, the software clearly favored men. The problem lay in the company's historical training data. As more men were invited to interview in person over the past few years, characteristics that are classically more associated with men came into focus in the model. In addition, a person's qualifications was evaluated based on whether or not he or she was hired, rather than on whether or not he or she was later successful on the job. Thus, the preferences of hiring managers were adopted in the trained algorithm. Further, women and people of color were severely underrepresented in the sample.

Attempts at improvement failed, and in 2017 the system was abandoned. So you always have to ask whether the training data reflect the required population and vulnerable subgroups without bias, and whether the quality of the data is equally good everywhere. But even then, it may happen in operation that the environment changes. Then retraining must be done with updated data. This also means that you have to think carefully about when an AI system goes into productive use, how much autonomy it is granted, and how frequently the system is checked.

Fairness is the fundamental requirement to treat the same social circumstances equally (Fig. 10.43), unless a different approach would be objectively justified. In particular, this means that individuals must not be discriminated against or given a better position because of their membership in a particular social group, their

Fig. 10.43 Decisions made by an AI system should be fair in their effects, i.e., not unreasonably disadvantage any affected parties. Image credits in Appendix A.3

religion, or their gender. Which social groups are protected and which are not must be determined by the legislature or social consent. AI applications learn from historical data. These are not necessarily bias-free. For example, if the data comprises disadvantages for women, the AI component will adopt these biases. In addition, certain groups may be underrepresented in the data base. This is known as bias, a distortion in the data. Bias can also lead to decisions that are unfair.

To address this issue, fairness must be operationalized and made measurable. First, one must identify those groups that should not be disadvantaged. For all identified groups, the results should be comparable. Comparable can mean "equal hit probability," but there are many other concepts as well. In addition, individual fairness can be used without considering groups, i.e., equal treatment of similar individuals. Which definition is the right one for the specific application has to be decided context-dependently and communicated to the user for transparency reasons.

10.5.5 Are the Functioning and Decisions of AI Comprehensible?

The transparency of an AI application can be decisive for its acceptance in many application areas. Two aspects must be distinguished here. First, information must be available on the correct handling of the AI application. Second, there are

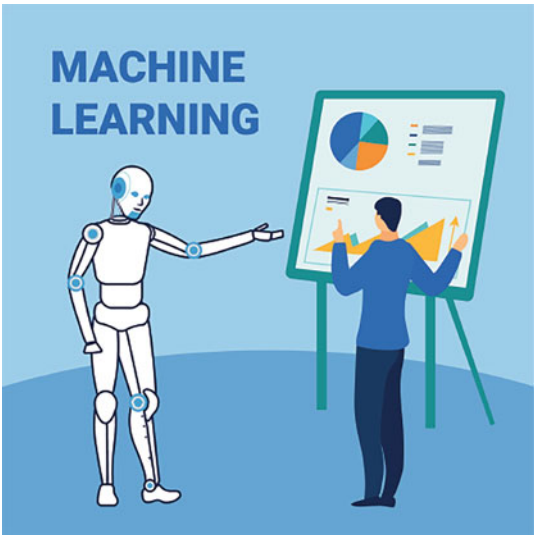


Fig. 10.44 An AI system should be able to explain how its results are obtained in a way that makes the basic features of the result comprehensible. Image credits in Appendix A.3

requirements for interpretability, traceability, and reproducibility of results, which require insights into how the AI application works (Fig. 10.44).

It must be clear to every user when he or she is communicating with an AI application. For this, it must be made clear what an AI application is. In addition, the stakeholders must be adequately familiarized with the utilization of the application. This includes an understanding of what purpose the application serves, what it does, and what the potential risks are. If the AI application is used in situations for which it has not been trained, there may be negative consequences. This means that AI applications must automatically monitor their reliability (Sect. 5.11).

The second type of transparency concerns the inner processes of the AI application and specifically the ML model. Users want to know how certain a result is, which alternatives exist, or what other result was missed. Developers want to understand exactly how a result was reached, and in the case of an error they want to find out why an incorrect prediction came up. There is a tension between higher accuracy or robustness on the one hand and the explainability of models on the other. AI engineers know the dilemma: Black-box models are in many cases more accurate or robust than, for example, rule-based models, but they are only interpretable to a limited extent. However, explainability can be partially achieved by downstream methods, such as local explanatory models or an analysis of the input/output behavior of models (Sect. 6.9.1). Model interpretability is an active area of research, and many efforts are being made to better understand the learning processes of black-box models, as well as to visualize their internal processes and make the resulting decisions plausible.

10.5.6 Are AI Systems Secure from Attack, Accident, and Error?

Security in the sense of protection against attacks (Security, Sect. 5.12.2) and protection against dangers arising from the application itself (Safety) are at least as important for AI applications as for other information and technical systems. Both safety concepts affect the entire application in which the AI component is embedded. Hazards can manifest as functional failure or severe functional change of the AI component, as well as unauthorized information leakage. Hazards must be detected by the system as early as possible. The AI component must then be sidestepped. Instead, control can be handed over to the user, or the system can be transitioned by conventionally programmed modules to a safe state. In the self-driving car, for example, the autopilot must be deactivated in the event of poor visibility, snowfall, and heavy traffic if insufficient training material was available for these situations (Sect. 8.5.3). The driver is informed and takes control himself.

10.5.7 Do AI Components Work Reliably and Perform Robustly?

Traditional testing of the security of an application must be supplemented by dedicated tests of the AI component, and especially of the model that has been machine-learned from the data. Very different quality criteria can be examined here: The performance or accuracy of the model (Sect. 3.5), the assessment of ML model uncertainties (Sect. 5.11), robustness to noise or malicious inputs (see Sects. 5.12, 6.10, 7.9, 8.5.4), but also resource utilization and response times. Novel events that are atypical and thus unexpected for humans can lead to potentially critical situations because they are not rehearsed, especially in direct human-machine interaction.

Reliability also includes the avoidance or early detection of possible data or model manipulations. If such attempts of deception (Fig. 10.45) occur, this must be recognized and it must be possible to quickly return to an error-free original version.

A correct implementation of the training routines and the application of the trained model is another requirement for a professional software. The tests to be performed for this purpose should be established in the field of Machine Learning and adapted to the respective application. If model weaknesses are discovered, appropriate correction mechanisms, including the use of a fallback plan, must be implemented. A convincing example of this is the testing of self-driving cars through extensive, realistic simulations at Waymo (Sect. 8.5.3). Especially in production operation, the reliability of the AI application must be ensured at all times. This implies that the functioning must be checked at appropriate intervals.

Fig. 10.45 Artificial Intelligence systems must be robust against attempted attacks and malicious deception. Image credits in Appendix A.3

When AI systems are properly implemented, they help to bring an objectivity and neutrality to data-driven, unbiased decision making by delivering consistent quality. It cannot be denied that humans also very often make subjective and biased decisions. For example, in studying over 1000 court decisions in the U.S., it was found that the likelihood of a favorable decision was greater at the beginning of the workday or after a meal break than later in the sequence of cases, and this effect was highly statistically significant. Thus, from the defendant's perspective, it is highly advantageous to appear at the beginning of the session (i.e., either at the beginning of the day or immediately after lunch break) (Danziger et al. 2011).

10.5.8 Does AI Protect Privacy and Other Sensitive Information?

AI applications are capable of interfering with a wide range of legal positions (Fig. 10.46). Particularly frequently, these are intrusions into individuals' privacy or the right to informational self-determination. For example, AI applications often process sensitive information, personal or private data, such as voice recordings, photos or videos. It must therefore be ensured that the relevant data protection regulations, such as the European General Data Protection Regulation (GDPR) and the Federal Data Protection Act (BDSG), are complied with.

AI applications may not only represent a risk to individual privacy. It could also be possible to discover trade secrets from AI models and usage data that require ethical or legal protection. One example is machine data, which contains information about process capacity utilization or error rates completely independent of the question of which person operated the machine.

An AI system is trained with data from individual cases, and the estimated parameters can be viewed as a kind of "average" over the individual cases. The parameters alone are used in a forecast. However, in certain cases it is possible to infer back to the individual cases from these parameter values. If the individual cases are persons, then data protection regulations do not permit the unrestricted use of such a model.

Fig. 10.46 There must be effective protection against the tendency of the Internet industry to capture and monetize all accessible user data and media with the help of AI. Image credits in Appendix A.3

Differential privacy aims at maximizing the informativeness of neural networks while minimizing the probability of identifying the individual cases used for training. In the context of Machine Learning, appropriate algorithms have been developed, which are now also available in TensorFlow, a toolbox for DNN (Galen 2019). The intuition for defining differential privacy is that an individual's privacy cannot be compromised by a statistical release if their data is not in the database. For this reason, the goal with Differential Privacy is to provide each individual with approximately the same privacy that would result from removing their data from the training set. That is, the DNN should not be overly dependent on any one individual's data. One can thus greatly reduce the probability of an individual being at risk and obtain strong and robust guarantees of this property for the trained model.

10.5.9 The Challenges for an AI Seal of Approval

A seal of approval for human-centered AI that has been created by social consensus and provides clarity about the characteristics of an application in the listed fields of action is certainly more than desirable. At the same time, the tester will always have discretionary powers. It is therefore all the more important that the examiners are able to demonstrate sufficient competence. Accreditation standards must be developed for this purpose. Moreover, a basic catalog can only formulate minimum requirements in general terms. They must be differentiated according to industry or application. For example, a test catalog for the medical context would need to have a different focus than one for autonomous driving or for industrial applications.

An AI seal of approval should be understood as a living test certificate that develops in parallel with the technology, our knowledge, and our values. We should constantly monitor that applicable law and our principles and values are respected and that the technology used is robust and reliable. To ensure this, a cross-disciplinary exchange on Artificial Intelligence between computer science, law, philosophy and politics is necessary.

One problem that certification cannot fully address is the possibilities of dual use. An AI application can be used for both positive and negative purposes. Every company and developer should be aware of these risks and reduce them where possible. Applications should be "by design" to be testable to the specified extent and difficult to apply beyond their intended purpose. The maxim should be to develop applications that preserve basic human rights by design and, if possible, do not harm people. Regardless of a concrete application, it is helpful to formulate an AI code of ethics in the company and to pay attention to diversity and heterogeneity in the composition of AI teams.

10.6 Summary

The changes coming as a result of Big Data and AI are fundamental and probably irreversible. They have implications for individuals and society: the social fabric, the value and design of work, human-machine interaction, and the free formation of political will and opinion.

Three areas of application were used to demonstrate the possibilities and influence of Artificial Intelligence. It is impressive how much AI will integrate into our lives. Whether in our own household, in the smart hospital of the future or in one of the classic German domains, the engineering-based manufacturing industry, AI will be used in a variety of ways. AI is expected to penetrate many other sectors of the economy as well.

Data has become an important asset, the raw material of digitization. American corporations in particular have acquired enormous amounts of data in recent years and are constantly monetizing it with great success. With the help of platforms and portals, they act as intermediaries between users and traditional providers who collect data and mediate services.

In the global platform economy, it is enormously important to create European alternatives. Google was cited as an example here: Originally started as a search engine, the company has expanded into numerous industries. If the current trend continues, Germany will be relegated to being a mere user or supplier of American and Chinese platforms, for example for mobility services, voice services and other intelligent capabilities of our machines. This could mean a massive restriction of our national sovereignty, as AI services and the associated central value creation of the future will be controlled abroad. To harness the potential of AI and shape it according to our values, Europe needs to play an active role in all areas of AI application. In particular, there is great potential for Germany in industrial analytics.

In this rapidly changing environment, Europeans should take a pioneering role and implement their moral and ethical ideas in intelligent systems.

Forecasts on the effects on the labor market are currently still very challenging. However, it can be seen that in addition to the further automation of manual work, the intellectual work of well-trained specialists also has considerable potential for substitution by AI. AI will take over many routine tasks and assist with more demanding activities. Analytical thinking, creativity, and the ability to solve problems in a team are key competencies of the future, which are only insufficiently provided by AI systems. Therefore, the corresponding educational programs must be modernized. In the future, there will be a massive demand for data scientists—AI designers who are able to think outside or at the edges of traditional processes and implement ideas that challenge existing structures.

We have discussed the question of the limitations of AI systems in terms of their robustness and vulnerability to malicious attacks in previous chapters. An AI system has no moral or ethical concerns. Its behavior follows the given loss or reward function and learned patterns in data that may be unrepresentative, of insufficient quality, or contain historical preferences and discriminations. In the application field of medicine, it becomes quite clear that ultimately humans are asked to weigh alternatives and make decisions (human in the loop).

For the trustworthy use of AI, we need a social dialog about the relationship between humans and machines as well as a reliable ethical-legal foundation. A seal of approval can establish a quality brand "AI Made in Germany/Europe" that makes responsibly designed, reliable and securely implemented applications recognizable and protects them over the long term. A seal of approval contributes to the acceptance of Artificial Intelligence in society.

References

Andreessen, M. (2011). Why software is eating the world. *The Wall Street Journal, 20*, C2 (2011)

Ärzteblatt (2018). GKV-Spitzenverband Nimmt Exoskelett Ins Hilfsmittelverzeichnis Auf. In *Ärzteblatt*.

Bastian, M. (2018). *600 Millionen US-Dollar Investition: Sensetime Ist Chinas Wertvollstes KI-Unternehmen*.

Bate, A. (2016). *ALPHA vs. The Pro - Judgement Day*. Raspberry Pi. https://www.raspberrypi.com/news/alpha-vs-pro-judgement-day/ (visited on 05 May 2022).

BearingPoint (2017). *Studie Smarte Gesundheit 2017*. https://www.bearingpoint.com/de-de/downloadformular/?item=8551&module=474592.

Bengio, Y. (2019). From system 1 deep learning to system 2 deep learning. In *Neural Information Processing Systems - NeurIPS'2019 Keynote*.

Biermann, K. (2015). Massenüberwachung: BND speichert 220 Millionen Telefondaten – jeden Tag. In *Die Zeit. Digital*. ISSN: 0044-2070. https://www.zeit.de/digital/datenschutz/2015-01/bnd-nsa-metadaten-ueberwachung?utmreferrer=https%3A%2F%2Fwww.startpage.com%2F (visited on 05 June 2022).

Biermann, K. (2019). Deutschland Will Zurückhacken. In *Zeit*. Online 07 December 2019.

BMBF (2018). *Strategie Künstliche Intelligenz Der Bundesregierung*. Bundesministerium für Bildung und Forschung. https://www.bmbf.de/files/NationaleKI-Strategie.pdf.

Brinker, T. J., Hekler, A., Enk, A. H., Klode, J., Hauschild, A., Berking, C., Schilling, B., Haferkamp, S., Schadendorf, D., Holland-Letz, T., et al. (2019). Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task. *European Journal of Cancer, 113*, 47–54.

Bryant, M. (2019). *How AI and machine learning are changing prosthetics*. https://www.medtechdive.com/news/how-ai-and-machine-learning-are-changing-prosthetics/550788/.

Cadwalladr, C., & Graham-Harrison, E. (2018). How Cambridge analytica turned Facebook 'Likes' into a lucrative political tool. In *Guard*.

Castro, D., McLaughlin, M., & Chivot, E. (2019). *Who is winning the AI race: China, the EU or the United States*. Washington DC Center for Data Innovation.

Cremers, A. B. et al. (2019). *Vertrauenssürdiger Einsatz von Künstlicher Intelligenz*. Fraunhofer Institut für Intelligente Analyse- und Informationssysteme, Sankt Augustin.

Dachwitz, I., Rudl, T., & Rebiger, S. (2018). Was Wir Über Den Skandal Um Facebook Und Cambridge Analytica Wissen. In *Netzpolitik.org*.

Danziger, S., Levav, J., & Avnaim-Pesso, L. (2011). Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences, 108*(17), 6889–6892.

Davenport, T., & Patil, D. (2020). Data scientist: The sexiest job of the 21st century. In *Harvard Business Review Oct 2012*.

Deuber, L. (2019). 'Überwacht, Überall - Wer in Der Volksrepublik Eine Neue Handynummer Will, Muss Künftig Vorher Sein Gesicht Scannen Lassen. In *Süddtsch. Ztg.*

Diamond, A. (2013). Executive functions. *Annual Review of Psychology, 64*, 135–168.

Diederich, J. (2021). *The psychology of artificial superintelligence*. Cognitive systems monographs (Vol. 42). Springer International Publishing. ISBN: 978-3-030-71841-1. https://doi.org/10.1007/978-3-030-71842-8. (Visited on 13 January 2023).

Dorloff, A. (2018). Mit Gesichtserkennung in Richtung Massenüberwachung. In *Dtschl.*

Dyrmann, M., Jørgensen, R. N., & Midtiby, H. S. (2017). RoboWeedSupport- detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. *Advances in Animal Biosciences, 8*(2), 842–847.

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandečič, D. (2014). IntroducingWikidata to the Linked DataWeb. In *International semantic web conference* (pp. 50–65). Springer.

Expert-Group (2019). *Ethics guidelines for trustworthy AI. report of high-level expert group on AI*. https://ec.europa.eu/digital-single-market/en/news/ethics-guidelinestrustworthy-ai.

Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., & Vechev, M. (2019). Dl2: Training and querying neural networks with logic. In *International Conference on Machine Learning* (pp. 1931–1941)

Galen, A. (2019). *Tensorflow privacy*. https://github.com/tensorflow/privacy.

García-Ajofrín, L. (2018). *Paying for fried chicken with the face: A journey around the biometric landscape in China*. https://outride.rs/en/paying-for-fried-chicken-withthe-face-a-journey-around-the-biometric-landscape-in-china/.

Görz, G., Schmid, U., & Braun, T. (Eds.) (2020). *Handbuch Der Künstlichen Intelligenz, 6. Auflage* (6th ed.) De Gruyter. 978-3-11-021808-4.

Grävemeyer, A. (2019). Künstlich Intelligente Diagnose Als Zweite Meinung. In *CT*.

Gromann, D., Espinosa Anke, L., & Declerck, T. (2019). Special issue on semantic deep learning. *Semantic Web, 10*(5), 815–822.

Guardian (2019). Paralysed man walks using mind-controlled exoskeleton. In *Guard.*

Hamrick, J. B. (2019). Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences, 29*, 8–16.

Hecker, D., Koch, D., Heydecke, J., & Werkmeister, C. (2016). Big-data-Geschäftsmodelle— Die Drei Seiten Der Medaille. *Wirtschaft für Management, 8*(6), 20–30.

Hecker, D., Döbel, I., Rüping, S., & Schmitz, V. (2017a). Künstliche Intelligenz Und Die Potenziale Des Maschinellen Lernens Für Die Industrie. *Wirtschaft für Management, 9*, 26–35.

Hecker, D., Petersen, U., Rauschert, A., Schmitz, V., Voss, A., & Döbel, I. (2017b). *Zukunftsmarkt Künstliche Intelligenz. Potenziale Und Anwendungen*. Fraunhofer- Allianz big data. https://www.bigdata.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/KI-Potenzialanalyse2017.pdf.

Hossain, M. T., Teng, S. W., Zhang, D., Lim, S., & Lu, G. (2018). Distortion robust image classification with deep convolutional neural network based on discrete cosine transform. Preprint. arXiv: 1811.05819.

I-Prognosis (2019). *I-PROGNOSIS aims to develop early and unobtrusive parkinson's disease detection tests based on the interaction of users with their everyday technological devices*. http://www.i-prognosis.eu/?pageid=59.

Ito, J. (2018). *Why westerners fear robots and the Japanese do not*. Wired.

Jordan, M. (2018). Artificial intelligence — The revolution hasn't HappenedYet. In *Medium*. https://medium.com/@mijordan3/artificial-intelligencethe-revolution-hasnt-happened-yet-5e1d5812e1e7.

Kahneman, D. (2011). *Thinking, fast and slow*. Macmillan.

Kanellos, M. (2016). *152,000 smart devices every minute in 2025: IDC outlines the future of smart things*. Forbes.com.

Keegan, M. (2020). *The top 10 most surveilled cities in the world*. US News & World Report. //www.usnews.com/news/cities/articles/2020-08-14/the-top-10-most-surveilled-cities-in-the-world (visited on 13 January 2023).

KillerRobots (2018). *Campaing toStop killer robots*. https://www.stopkillerrobots.org/.

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. In: *Proceedings of the National Academy of Sciences, 110*(15), 5802–5805.

Kropp, P., Theuer, S., & Fritzsche, B. (2018). *Immer Mehr Tätigkeiten Werden Durch Digitalisierung Ersetzbar*. IAB 2018, S SN 1861-14 35, S.32.

Kruppe, T. et al. (2019). *Digitalisierung: Herausforderungen Für Die Aus-UndWeiterbildung in Deutschland* (note 2195–5980). Instituts für Arbeitsmarkt-und Berufsforschung. http//doku.iab.de/stellungnahme/2019/sn0119.pdf

Lepies, J. (2017). Künstliche Intelligenz in Der Medizin: Wir Wollen Ärzte Nicht Arbeitslos Machen. In *Heise*. Online 09 July 2017.

Leviathan, Y., & Matias, Y. (2018). Google duplex: An AI system for accomplishing real-world tasks over the phone. https://blog.research.google/2018/05/duplex-ai-system-for-natural-conversation.html

Lewis, N. (2019). Why robots will soon be picking soft fruits and salad. In *CNN.com*.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 3749–3759.

Marcus, G. (2018). Deep learning: A critical appraisal. Preprint. arXiv: 1801.00631.

Marks, A. (2019). How AI is radically changing our definition of human creativity. In *Wired*.

Metz, C. (2016). Inside OpenAI, Elon Musk's wild plan to set artificial intelligence free. In *Wired.*.

Miners, Z. (2014). Google buys smart thermostat maker nest for a cool $3.2B. In *Computerworld*.

Mistreanu, S. (2018). Life inside China's social credit laboratory. In *Foreign policy*.

Mitchell, M. (2018). Artificial intelligence hits the barrier of meaning. In *N. Y. Times*.

Mukherjee, S. (2017). AI versus MD: What happens when diagnosis is automated? In *New Yorker, 3*.

Ng, A. (2016). What artificial intelligence can and can't do right now. In *Harvard Business Review, 9*.

Nicklaus, J. (2007). *Golf my way: The instructional classic, revised and updated*. Simon & Schuster.

O'Neil, C. (2016). *Weapons of math destruction: HowBig data increases inequality and threatens democracy*. Broadway Books.

Osborn, L. E., Dragomir, A., Betthauser, J. L., Hunt, C. L., Nguyen, H. H., Kaliki, R. R., & Thakor, N. V. (2018). Prosthesis with Neuromorphic multilayered E-dermis perceives touch and pain. *Science Robotics, 3*, 19.

Parliament, E. (2017). *European parliament resolution of 16 February 2017 with recommendations to the commission on civil law rules on robotics*. https://www.europarl.europa.eu/doceo/document/TA-8-2017-0051EN.html.

Pearl, J. (2018). Theoretical impediments to machine learning with seven sparks from the causal revolution. Preprint. arXiv: 1801.04016.

Petzinger, J. (2018). Europe's newest unicorn is a process-mining startup founded by german students. In *quartz 26.06.2018*.

Radebaugh, C., & Erlingsson, U. (2019). Introducing TensorflowPrivacy: Learning with differential privacy for training data. In *Medium*. Online 03 June 2019.

Richter, R. R., & Austin, T. M. (2012). Using MeSH (medical subject headings) to enhance PubMed search strategies for evidence-based practice in physical therapy. In *Physical Therapy, 92*(1), 124–132.

Scudellari, M. (2019). Bionic 'feeling' leg makeswalking easier, reduces phantom limb pain. In *IEEE Spectrum 2019*.

Severson, K. (2020). Thanks to A.I., Machines get a taste for the right kinds of food. In *N.Y. Times*.

Shoham, Y., Perrault, R., Brynjolfsson, E., Clark, J., Manyika, J., Niebles, J. C., Lyons, T., Etchemendy, J., Grosz, B., & Bauer, Z. (2018). The AI index 2018 annual report. In: *AI index steering committee*. Human-Centered AI Institute.

Slavin, E. (2017). Pentagon unveils Perdix micro-drone swarm. In *Stars stripes*. Online 01 October 2017.

Smith, C. (2020). Want to be better at sports? Listen to the machines. In: *N. Y. Times* .

Smith, C. S. (2020). Dealing with bias in artificial intelligence. In: *N. Y. Times*.

Tagesspiegel (2019). Amazon-Mitarbeiter Hören Alexa-Aufzeichnungen Ab. In *Tagesspiegel*.

TensorProb (2019). *TensorFlow probability*. https://www.tensorflow.org/probability (visited on 13 September 2019).

Tischbein, V. (2016). 98 Daten, Die Facebook Über Dich Weiß Und Nutzt, Um Werbung Auf Dich Zuzuschneiden. In *Netzpolitikorg*. Online 22 August 2016.

Valentino-deVries, J. (2018). Your apps know where you were last night, and they're not keeping it secret. In *N. Y. Times*.

Wang, H., Zheng, F., Chen, Z. Lu, Y., Gao, J., & Wei, R. (2018). A captcha design based on visual reasoning. In *2018 IEEE international conference on acoustics, speech signal process, ICASSP* (pp. 1967–1971).

Whittaker, Z. (2019). *OpenAI built a text generator so good, it's considered too dangerous to release*. Techcrunch. https://techcrunch.com/2019/02/17/openai-textgenerator-dangerous/.

Wikimedia (2019). *Malicious attack on Wikipedia—what we know, and what we're doing*. Wikimedia Foundation. https://wikimediafoundation.org/news/2019/09/07/malicious-attack-on-wikipedia-what-we-know-and-what-were-doing/ (visited on 12 January 2023).

Wrobel, S., & Hecker, D. (2019). Fraunhofer big data and artificial intelligence alliance. In *Digital transformation* (pp. 253–264). Springer.

Appendix A
Appendix

A.1 Mathematical Notation

$L(w)$	Loss function to be minimized during training by modifying w.
sigmoid(x)	$= \frac{1}{1+\exp(-x)}$
softmax(u)	$= \frac{(\exp(u_1),\ldots,\exp(u_k))}{\exp(u_1)+\cdots+\exp(u_k)}$ transforms any vector $u = (u_1, \ldots, u_k)$ into a probability vector.
TrainSet	Training data set $\left\{(x^{(1)}, y^{(1)}), \ldots, (x^n, y^n)\right\}$ of observations of input-output pairs $(x^{(i)}, y^{(i)})$.
x	Input vector or tensor.
w	Parameter vector, which is modified by optimization algorithms.
y	Output scalar, vector or tensor.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5

A.2 Glossary

5G	5G is the fifth generation of a mobile communications standard of wireless broadband technology for mobile Internet and mobile telephony. 5G is currently one of the most powerful mobile communications standards.
Accuracy	A performance measure of classification: $\text{Accuracy} = \frac{\text{number of correctly classified test samples}}{\text{number of all test samples}}$ Value between 0.0 (poor) and 1.0 (good)
Activation function	A nonlinear function as an operator in a network. The network subsequently can represent non-linear functional relations. • The function is applied to all components of a vector. • Often component of a fully connected layer. Example: $(u_1, \ldots, u_k) \rightarrow (\tanh(u_1), \ldots, \tanh(u_k))$. Other variants are ReLU and sigmoid.
Affine Transformation	Layer in a network. Multiplies an input vector x by a matrix W and adds a vector b, called bias $u = W * x + b \text{ thus } u_i = w_{i,1} * x_1 + \cdots + w_{i,n} * x_n + b_i$ • Converts straight lines to straight lines. • W determines a rotation and stretching transformation and b a global shift. • Basic building block of most deep neural networks.
Algorithm	A step-by-step guide to perform a task, usually by a computer. Computer algorithms can be simple (e.g., determine the sum of the two numbers by following steps …) or complex (e.g., determine the object in this photo by following steps …).
Artificial Intelligence (AI)	The term Artificial Intelligence, abbreviated AI, stands for computer systems that can imitate human intelligence.
Artificial Neural Network (ANN)	see Neural network.
Autoencoder	An artificial neural network used to learn an efficient encoding of the input. It consists of several layers. • The goal is to reproduce the input by the output. • Input and output vectors are of equal size. • The middle layer has only a short hidden vector. This forces the network to produce an effective compression of the input by the hidden vector.

Backpropagation	Procedure to compute the gradient $\frac{\partial L(x,y;w)}{\partial w}$ of a neural network loss function for a given training example (x, y). Because of the individual layers $f_i(x, w)$ of a network, the loss function is a nested function of the layers $L(x, y; w) = L(f_k(\cdots f_1(x, w) \cdots, w), y)$ Its derivative is calculated with the chain rule. • The loss L evaluates the deviation between predicted and observed output y. Therefore, for a given input x, the neural network must first predict the output y. • The derivative of the loss is “propagated back” from the output to the input layer. • In current frameworks for the definition of neural networks, the gradient is computed automatically.
Beam Search	A heuristic search algorithm. It considers the k best partial results so far during successive search. In machine translation, it is used in generating more probable output words sequentially.
BERT	A sequence model, which is able to compute contextual embeddings. As inputs it receives a sequence of tokens $v_1, \ldots, v_t$. These tokens are encoded by static token embeddings, which are learned during training. Embeddings of the positions of tokens in the sequence are added. Inside BERT multi-head self-attention is used to compute contextual embeddings of the tokens. BERT consists of 6 Transformer encoder blocks, which are executed in sequence. During pre-training with a large corpus of text about 15% of the input tokens are replaced with the token “[MASK]”. The training task is to predict these masked tokens using the contextual embeddings in the highest layer as input of a logistic regression model. After pre-training with a large general text corpus the model usually is fine-tuned for a specific task. To the text a first token “CLS” is prepended. Its embedding in the highest layer is used during fine-tuning to predict class information, e.g. the sentiment of a sentence.
Bidirectional LSTM	When predicting the property y_t of word x_t in a sequence $x_1, \ldots, x_T$ with a recurrent neural network (RNN), only the previous words are considered. The bidirectional RNN considers previous and subsequent words yielding better predictions: • The Forward RNN has the words $x_1, \ldots, x_{t-1}$ as inputs and predicts the hidden vectors $\overrightarrow{h}_t$ • The Backward RNN considers the words $x_T, x_{T-1}, \ldots, x_{t-1}$ as inputs and predicts the hidden vectors $\overleftarrow{h}_t$ • The logistic regression model for predicting y_t receives the concatenated hidden vectors of both models $[\overrightarrow{h}_t, \overleftarrow{h}_t]$ as input.

Black-box-model	A Machine Learning model or DNN that transforms input tensors into output tensors. The inner workings of the transformation are (almost) unknown and usually cannot be understood in detail by humans.
BLEU	Performance measure for evaluating machine translation. It compares the translation result with reference translations. The number of equal words, 2-grams, … , 4-grams is counted and aggregated to a value between 0.0 and 1.0. • higher value → better translation
Chatbot	A chatbot, or bot for short, is a text-based dialog system that exchanges messages with human users.
Classification	A type of supervised learning in which the target variable can take only finitely many discrete values. An example is the classification of the sentiment in a text as positive, neutral, or negative.
CNN	See Convolutional Neural Network
Computer vision	Computer programs that allow interpretation of the content of images or videos. Familiar tasks are object recognition and the description of an image by text.
Contextual Embedding	An embedding vector for a word or token, which takes into account the context of the word. This is especially important for words like “bank”, which can have different meanings depending on the context.
Convolutional Neural Network (CNN)	A neural network, which is usually applied to 2- or 3-dimensional sensor input (e.g. pixel matrix of an image) and generates a prediction, e.g. the class of an object in the image. It contains • Convolution layer: a small 2-dimensional matrix (kernel) is moved over the input; it computes a value for each position. Usually many kernels are used in parallel. • Pooling layer: Aggregation of neighboring values, e.g. by maximum. • Often residual connections are used to simplify optimization. These layers can be stacked multiple times. The last layer is usually a logistic regression for classification.
Cross-attention	Computes an attention between embeddings $x_1, \dots, x_T$ of an input sequence and embeddings $z_1, \dots, z_m$ of an output sequence. The calculations are performed analogously to the self-attention, with the difference that the query vectors $q_j = Q * z_j$ are calculated from the embeddings of the output sequence, while the key and value vectors $k_t = K * x_t$ and $v_t = V * x_t$ are calculated from the embeddings of the input sequence. It is used to map information from the input sequence to the output sequence, e.g. for translation with the Transformer.
Data	A set of sample data, also called observations or instances. • Each sample data contains inputs (and outputs, if any) for the Machine Learning procedure, e.g., images, speech signals, texts, measured values. All types of data are converted to numbers, usually tensors, for processing.

Deep Learning	The process by which a deep neural network is adapted to the training data through stepwise optimization. This involves determining internal representations for the hidden vectors that recode the inputs in each layer of the DNN. Finally the output is computed. The parameter vector w is modified during training such that the value of the loss function gets small and the model fits well to the training data.
Deep Learning Toolkit	System for specification and execution of deep neural networks: CNTK, Pytorch, TensorFlow, ... • operators and connections are specified in a simple way. • gradients are determined automatically. • execution on clusters of CPUs and GPUs.
Deep Neural Network (DNN)	A neural network with many layers. • A deep neural network can approximate very complex functions. • Generalization: network with arbitrarily connected operators (connection graph). The operators compute an output tensor from one or more input tensors. The connection graph should be a directed acyclic graph.
Deep Q-Network	In reinforcement learning, the Q-Function represents the expected value of a state-action pair: • The Deep Q-Network $\hat{Q}(s_t, a_t; w)$ with parameter w is a deep neural network to approximate the Q-Function. After training, it can be used to determine the best action a for a given state s_t.
Digitization	Digitization aims to transform analog objects and processes in the real world into digital data and processes. The digital transformation goes hand in hand with the Internet of Things, in which people, machines and devices are networked. By 2020, 50 billion devices are expected to be networked and exchange data intelligently with each other.
Directed Acyclic Graph (DAG)	Directed Graph without cycles but possibly parallel paths.
Directed Graph	A graph with edges having a direction. Neural networks often not only have a single sequence of successive layers, but form a directed graph of operators. • The nodes (operators) of the graph model individual computations. • The directed edges of the graph correspond to data flowing from one operator to another. These edges can be parallel. Mostly, these are directed acyclic graphs. These model structures can be represented in toolboxes like TensorFlow.
DNN	Deep neural network
Dropout	An approach to regularization. • It assumes a hidden vector h_t within a model. • In a dropout layer, a certain percentage (e.g. 50%) of the components of h_t are randomly selected and set to 0.

Embedding Network	Calculates static embeddings for the words in a sequence. • The static embedding of a word is used as network input. • From this input, the words in the neighborhood of the word are predicted as well as possible. • In this process, parameters of the embeddings are optimized so that the prediction is as good as possible. The embeddings of word with similar meaning have a low vector distance. An example network is Word2Vec.
Embedding	Vector representing a word or token of a language. The embeddings of semantically similar words should have a small vector distance. • Embeddings can also computed for larger objects, (e.g. sentences, paragraphs, documents, images) to represent their contents. Embeddings are used for most of the recent deep neural networks.
Ensemble Methods	Ensemble methods use a set of similar models for the same training data to solve a prediction task. The models have random or systematic differences. • Bootstrap or bagging randomly changes the training set to train different models. • Boosting gradually focuses on the difficult prediction cases. A prediction is computed by averaging the model predictions. It can be shown for both cases that the prediction variance is lower than for single models. The methods can be applied to arbitrary Machine Learning methods, i.e., also to neural networks.
Episode in reinforcement learning	A sequence of observed triples (state, reward, action) in reinforcement learning resulting from a model run: $(s_1, r_1, a_1), (s_2, r_2, a_2), \ldots, (s_n, r_n, a_n)$
Epoch	Part of an optimization where all training examples are used once.
Expert System	An AI system that represents content as logical facts and rules, which are usually entered manually. One can then derive properties of objects by logical reasoning. Was widely used in the previous century.
Exploding Gradient	During training the gradient of a model can become very large, e.g. after k applications of a recurrent network, a parameter w enters the gradient as w^k. • $w > 1$: the value of w^k becomes extremely large and an exploding gradient results. • an optimization step will change the parameter vector too much and the optimization may not converge Workaround: constrain the length of the gradient.
Exploration	In order to find the best policy in reinforcement learning, it is necessary that a large part of the state-action set is evaluated during training. Therefore, new episodes should not always be determined with the optimal actions according to the currently best policy, but actions should also sometimes be chosen randomly
Fully Connected Layer	A layer of a neural network that performs an affine transformation involving all inputs and outputs. Subsequently the output is transformed by a nonlinear activation function g: $y = g(W * x + b)$

F-value	A performance measure of classification with respect to a given class c: $F_c = \frac{2 * \text{precision}_c * \text{recall}_c}{\text{precision}_c + \text{recall}_c}$ Harmonic mean of precision_c and recall_c. Value between 0.0 (bad) and 1.0 (good).
Generalization	Meaningful and adequate transfer of relationships in existing data to new data.
Generative adversarial network	Generative adversarial networks consist of two neural networks operating against each other. The generator network creates objects, e.g. images. The discriminator network receives real images or generated images in random order. It must decide whether the images are real or generated. • Typically, the generator network maps a vector of random values to objects, e.g. images. The goal of the generator is to produce images that are not rejected by the generator. • The discriminator is trained to distinguish the images created by the generator from the real images of the training set. As a result, the generator can produce images that the discriminator can no longer distinguish from the true images in the training set.
G-Force	G-forces are forces that act on the human body or a vehicle due to strong changes in magnitude and/or direction of velocity. In particular, fighter pilots and astronauts are subjected to high G-forces during air combat or rocket launches. High G-forces can only be withstood by humans for a short period of time.
GPT language model	A language model to predict the next token v_t in a sequence $v_1, v_2, \ldots$. It has the following modules: • A Transformer decoder block without cross-attention. It computes a self-attention for the embedding vectors of the for the previously generated tokens $v_1, \ldots, v_t$ and generates a new contextual embeddings for these tokens. • A logistic regression model uses the embedding of the righmost token v_t in the highest decoder block to predict the probability vector for the next token v_{t+1}. The model is repeatedly applied to predict the tokens of a text. The GPT2-version has 1.7 billion parameters, the GPT-3 model has 175 billion parameters.
Gradient Descent Optimization	A method for parameter optimization for a loss function $L(w)$ given training data $\text{TrainSet} = \left\{(x, y)^{(1)}, \ldots, (x, y)^{(n)}\right\}$: 1. The value of the parameter vector $w^{(0)}$ is initialized (e.g., randomly). 2. The gradient $\frac{\partial L(w)}{\partial w}(w^{(t)})$ is calculated for the current parameter $w^{(t)}$. This yields a vector $\frac{\partial L(w)}{\partial w}(w^{(t)})$ that is equal in length to $w^{(t)}$. 3. The gradient is multiplied by the learning rate λ (small positive number) and subtracted from the current parameter $w^{(t)} w^{(t+1)} := w^{(t)} - \lambda \frac{\partial L(w)}{\partial w}(w^{(t)})$

	This step is repeated until the changes in the value of $L(w)$ become very small. λ is a hyperparameter.
Gradient of $L(w)$	The direction (vector) $\frac{\partial L(w)}{\partial w}(w^{(0)}) = \left(\frac{\partial L(w)}{\partial w_1}(w^{(0)}), \ldots, \frac{\partial L(w)}{\partial w_k}(w^{(0)})\right)'$ in which the function $L(w) : \Re^k \to \Re$ of parameter w has the steepest ascent at point $w^{(0)}$. $\frac{\partial L(w)}{\partial w_i}(w^{(0)})$ is the partial derivative of $L(w)$ with respect to w_i at point $w^{(0)}$.
Graphics Processing Unit (GPU)	A processor originally specialized and optimized for computing graphics. • Often contains several thousand computational units and has a very high computational capacity. • Can also perform the computations for training a DNN. • Current neural network frameworks (e.g., TensorFlow) can use the graphics processor for parallel computations.
HAL 9000	British author and physicist Arthur C. Clarke described the fictional computer Hal 9000 in his book "2001: A Space Odyssey." Aboard the spacecraft Discovery, the supercomputer HAL (Heuristically programmed Algorithmic Computer) controls and monitors mission processes and purpose. In addition to spacecraft control, HAL also impresses by playing chess, recognizing speech, and responding in natural language.
Hidden vector	Internal vector in a multilayer neural network. No observed values are available for this vector, but it is computed internally during model prediction.
Hyperparameter	A hyperparameter is a parameter whose value is set before the learning process begins, e.g., the number of layers in a neural network or the learning rate. • Unlike normal parameters, hyperparameter values are not determined by training but often by trial and error.
Hyperparameter Optimization	Procedure for selecting the values of hyperparameters. • A simple approach trains multiple models on the training data with different values of the hyperparameters and compares model performance on an additional validation set of annotated examples.
Information Extraction	The extraction of information from text, in particular: • The extraction of named entities (objects with names) and the determination of the type of entities (Named Entity Recognition). • The extraction of relations between entities (Relation Extraction).

Instance of Data	Element, example of training or test set
Internet of Things (IoT)	Internet of Things (IoT) refers to a further development in digitization. In the Internet of Things, products and devices are connected both with each other and directly with the Internet itself. Everyday objects are equipped with processors or sensors and exchange information about their use, environment and user behavior. The IoT thus has a high potential for optimizing the performance and service of the networked devices.
Keras	Keras is a neural network library written in Python. • Contains numerous neural network building blocks which can be used in a simple way. • Acts as a control module for TensorFlow. • The computations are performed automatically on different computing devices, e.g. GPUs.
Kernel of a Convolutional Neural Network	In a convolution layer, a small matrix is moved over a large tensor (e.g., pixel matrix) of inputs. Generates an output value for each position of the input matrix.
Language Model	Statistical model with a parameter w. Predicts the next word v_t of a text from all previous words $v_1, \ldots, v_{t-1}$ by estimating the probability of the next word $p(v_t \| v_1, \ldots, v_{t-1}; w)$.
Layer of a Neural Network	An operator in a neural network that receives an input (e.g., input vector) and transforms it into an output (e.g., hidden vector) through simple calculations. Usually the next layer receives the output of the previous layer as input.
Learning Rate	Small positive real number λ determining the step size in gradient descent optimization.
Likelihood	Assumes a model $y \sim p(y \| x; w)$ which from the input x predicts the distribution of y and depends on a parameter w. The likelihood of a parameter w for a given training set $\text{TrainSet} = \left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\right\}$ of independent observations $(x^{(i)}, y^{(i)})$ is the probability of the whole training set $\text{lik}(w) = p(\text{TrainSet} \| w) = p(y^{(1)} \| x^{(1)}; w) * \ldots * p(y^{(n)} \| x^{(n)}; w)$ A large value of $\text{lik}(w)$ indicates that the model $p(y \| x; w)$ can reproduce the training data well.
Linear Transformation	Multiplies an input vector x by a matrix W and $u = W * x$, so $u_i = w_{i,1} * x_1 + \cdots + w_{i,n} * x_n$ • Converts straight lines to straight lines. • The matrix W determines the rotation and stretching of the transformation. • Part of the affine transformation $u = W * x + b$.

Logistic Regression	The logistic regression model calculates the probability vector for an input vector x by $p(y\|x; w) = \text{softmax}(Ax+b)$, where $w = vec(A, b)$ is the parameter vector. The i-th component of the vector $p(y\|x; w)$ can be interpreted as the probability of the i-th class. Choosing the class with the highest probability gives the associated classifier. • The model is often used as the last layer in deep neural networks for classification.
Log-Likelihood Loss	For a training set TrainSet $= \left\{(x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\right\}$ of independent observations $(x^{(i)}, y^{(i)})$ the Maximum Likelihood loss function $L(w) = \log p(\text{TrainSet}\|w)$ is $L(w) = -\left(\log p(y^{(1)}\|x^{(1)}; w) + \cdots + \log p(y^{(n)}\|x^{(n)}; w)\right)$ Maximizing $L(w)$ with respect to w yields a model $p(y\|x; w)$ which can optimally reproduce the training set.
Long Short-Term Memory (LSTM)	A language model to compute the probability for next element in a sequence. Input x_t represents the current sequence element, there is a hidden vector h_{t-1}, and a memory vector c_{t-1}. $u_t = [x_t, h_{t-1}]$ is the concatenation of x_t and h_{t-1}. The LSTM has three components: • store gate: adds modified inputs u_t to memory vector, controlled by transformed inputs u_t. • forget gate: sets single memory vector components to 0.0, controlled by transformed inputs u_t. • output gate: reads memory vector components controlled by transformed inputs u_t. The components are transformed to the new hidden vector h_t. Outputs: Memory vector c_t and hidden vector h_t. Has 5 fully connected layers with trainable parameters.
Loss function	Real-valued function $L(\text{TrainSet}; w)$ that, for the given data TrainSet, which measures how good a model with the current parameter w can reproduce the data. Usually Maximum Likelihood loss functions are used. • A smaller value of $L(\text{TrainSet}; w)$ indicates that there is only a smaller difference between predicted and observed outputs. • The loss function for multiple training examples is usually the sum of the loss values for individual training examples For a fixed set of training data TrainSet, the loss function is often written as a function $L(w)$ of the parameter vector w.
Lot size 1	In manufacturing, lot size 1 describes the ability of producing individualized products at the price of mass products. The manufacturing processes are so efficient and highly automated that it now makes no significant difference in cost whether one produces an individual customer product or a series product.
Machine Learning	Automatically extracts information from data that is relevant to the user using a trainable model.
Machine Translation	Automatic translation of text from one language to another.

Matrix	A rectangular arrangement of real numbers, e.g. $\begin{pmatrix} 1.7 & -6.0 & -9.5 \\ -1.2 & 0.3 & 0.7 \\ 2.8 & 4.7 & 3.8 \end{pmatrix}$ A matrix is a tensor.
Maximum Likelihood Loss	A loss function arising from the Maximum Likelihood principle. To avoid very small numbers, the probability of the training set is transformed with the monotonous function $\log(\cdot)$ and multiplied by -1. For a training set $\text{TrainSet} = \left\{(x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\right\}$ of independent observations $(x^{(i)}, y^{(i)})$ the negative log-probability of the whole training set is $$L(w) = -\left(\log p(y^{(1)} \mid x^{(1)}; w) + \cdots + \log p(y^{(n)} \mid x^{(n)}; w)\right)$$ A small value of $L(w)$ indicates that the model $p(y \mid x; w)$ can reproduce the training data well. Minimizing $L(w)$ with respect to w yield the same results as maximizing $\text{lik}(w)$.
Maximum Likelihood Principle	Uses a model that depends on a parameter w and determines the probability or likelihood $\text{lik}(w)$ of the training data as if the data had been generated by the probabilistic model $y \sim p(y \mid x; w)$ • Maximum Likelihood principle: change the parameter vector w so that the likelihood $\text{lik}(w)$ of the training data becomes maximal.
Minibatch	A small, randomly selected subset of m, e.g. $m = 100$ of the available training examples.
Multi-head cross-attention	Parallel execution of different cross-attention operators with different sets of K, Q, V matrices
Multi-head self-attention	Parallel execution of different self-attention operators with different sets of K, Q, V matrices
Model	A function that assigns an output (e.g. class label) to an input (e.g. image, tensor). A model usually has a vector w of 'free' parameters whose value is optimized based on the training data. • More generally, is assumed that for a given parameter w and input x the conditional distribution $p(y \mid x; w)$ controls the generation of output y.
Model application	Using a trained model to predict/analyze data not used in training.
Multilayer Neural Network, Multilayer Feedforward Network	A neural network consisting of multiple layers stacked on top of each other. • Each layer forms a function $x^{[i+1]} = f_i(x^{[i]})$ which transforms a tensor $x^{[i]}$ with the function $f_i(\cdot)$ into another tensor $x^{[i+1]}$. • The network performs these functions sequentially $x^{[j]} = f_{[j-1]}(\cdots f_1(x^{[1]}) \cdots)$ • It can be used for classification (with logistic regression as last layer) or for regression (with a linear last layer).
Named Entity	Named object (name) in the text, names for entities (people, places, products, etc.)

Natural language processing (NLP)	Process in which the computer attempts to “understand” different aspects of spoken or written language and to react meaningfully. It must understand both the sentence structure (syntax) and the meaning (semantics) of the words. Today, this is usually done with Machine Learning and Deep Neural Networks.
Neural Network	A computer program consisting of several layers, which transforms an input (e.g. image) into an output (e.g. object class) by simple calculations. The next layer usually receives the output of the previous layer as input. The connection of the layers can also have a mesh structure. A neural network has a vector w of unknown parameters, which is modified by optimization procedures. The goal is to predict the corresponding outputs as well as possible from the observed inputs.
N-gram	Small subsequence of fixed length n from a larger sequence • letter N-gram: sequence of N consecutive letters from a text. • word N-gram: sequence of N consecutive words from a text.
Nonlinear Function	A function $g(x)$ that is not an affine function $f(x) = Ax + b$.
Operator	Component in a neural network which performs computations. • Has one or more tensors as input and one or more tensors as output.
Overfitting	A model may overfit the existing data and reproduce non-systematic, random variations in the data. • Such a model generally has higher prediction errors on new data than on the training data and thus generalizes poorly. • Regularization can be used as a countermeasure.
Parameter	A vector of numbers in a neural network (or other Machine Learning model) that affects the output of the model. This vector is modified during training by optimization methods. The aim is to reduce the difference between the predicted outputs and observed training data.
Perplexity	Measure of the performance of the language model $p(v_i \mid v_1, \dots, v_{i-1})$: the inverse probability of the test set $v_1, \dots, v_N$, normalized by the number of words of the test set: $p(v_1, \dots, v_N)^{-1/N}$ • Smaller is better.
Policy Gradient Model	A model for reinforcement learning. It employs a probability distribution $\hat{\pi}(a_t \mid s_t; w)$ over the actions as a policy that can be used to select the next action. • Determines the probability of different actions and selects actions using a random mechanism. • Is implemented by a neural network with parameter w. The parameter w is optimized during training such that the average reward is maximal.

Policy in Reinforcement Learning	A function $\pi : s_t \rightarrow a_t$ that assigns an action a_t to each state s_t during reinforcement learning • The policy can also be random, i.e., select an action according to a probability distribution. This requires special methods: e.g. policy gradient model. • sometimes called strategy.
Pooling Layer	A component in a Convolutional Neural Network that computes an aggregation (e.g., maximum) of neighboring elements in a tensor. The output tensor thus contains fewer elements.
Precision	Performance measure of classification with respect to a given class c: $\text{Precision}_c = \frac{\text{number of correctly assigned test examples of class } c}{\text{number of all test examples assigned to class } c}$ Value between 0.0 (bad) and 1.0 (good).
Probability vector	Vector $p = (p_1, \ldots, p_k)$ of real numbers: • Each element is not negative: $p_i \geq 0$. • The sum of all elements is 1: $1.0 = p_1 + \cdots + p_k$ p_i can be interpreted as the probability (plausibility) of the i-th of k alternatives/classes.
Q-Function in Reinforcement Learning	Assigns a real value $Q(s, a)$ to each state-action pair (s, a) in reinforcement learning: • $Q(s, a)$ to is the maximum discounted future reward achievable for (s, a) with an optimal policy $\pi^* : a \rightarrow s$. • $\gamma < 1$ is the discount factor that controls how the weight of future rewards r_{t+i} is reduced. $Q(s_t, a_t) = \max_{\pi^*} \left(\gamma^0 r_{t+1} + \gamma^1 r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} \right)$
Recall	Performance measure of classification with respect to a given class c: $\text{Recall}_c = \frac{\text{number of correctly assigned test examples of class } c}{\text{number of all actual test examples of class } c}$ A value between 0.0 (bad) and 1.0 (good).
Recurrent Neural Network (RNN)	A language model for predicting the next element (word) v_t of a sequence $v_1, \ldots, v_n$: • input: previous element v_{t-1} and previous hidden vector h_{t-1}. • output: next hidden vector h_t and probabilities for all possible next elements v_t at position t. • loss function: same as for logistic regression model, is summed over the entire sequence.

Regression	A prediction model $y = f(x, w)$ where the target variable y is continuous $y = f(x, w)$. y can be a scalar or a tensor. • Usually a Gaussian distribution $p(y\|x, w)$ is assumed, where $f(x, w)$ is the expected value of y. Then for a training set TrainSet $= \left\{(x^{(1)}, y^{(1)}), \cdots, (x^{(n)}, y^{(n)})\right\}$ of independent observations $(x^{(i)}, y^{(i)})$ the Maximum Likelihood loss function $L(w) = \log p(\text{TrainSet}\|w)$ is $L(w) = \\|p(y^{(1)} - f(x^{(1)}; w)\\|^2 + \cdots + \\|p(y^{(n)} - f(x^{(n)}; w)\\|^2$ Minimizing $L(w)$ with respect to w yields a model $f(x, w)$ which can optimally reproduce the training set.
Regularization	Measure to reduce overfitting. There are a number of alternatives: • Reduction of model complexity (fewer layers, fewer parameters). • Additional term of the loss function that “pulls” the values of the parameters toward 0.0. • Dropout • Minibatch normalization, etc.
Reinforcement Learning	A learning task, where the relevant learning signal is available only after a series of steps. Example: win, loss or draw in chess • for a state s_t, the agent (the algorithm) chooses an action a_t. • This results in a new state s_{t+1} and a numerical reward r_{t+1}. The goal is to choose the actions so that the sum of all rewards is maximized. • In deep reinforcement learning, a neural network is constructed that predicts the optimal action in each state.
Residual Connection	An approach to improve the convergence of optimization during training. As a bypass the input of a layer is copied and added to the output. This allows to train deep neural networks with hundreds of layers.
Reward for Reinforcement Learning	For each time step t, a real number r_t, e.g., an amount of money. The goal of reinforcement learning is to maximize the sum of rewards.
Scalar	A real number, e.g. $-1.7 \in \Re$. A scalar is a tensor.

Self-attention	A layer in a network to compute a contextual-embedding for each token. As inputs it receives a sequence of embeddings $x_1, \ldots, x_T$ of the tokens $v_1, \ldots, v_T$. The inputs are multiplied with the “query” $q_t = Q * x_t$, the “key” $k_t = K * x_t$, and the “value’ $v_t = V * x_t$ matrix. For each input embedding x_j the following computations are repeated • The normalized scalar product $\alpha_t = q_j^{'} k_t / \sqrt{d}$ computes the association or attention between q_j and k_t, where $1/\sqrt{d}$ is a normalizing factor with respect to the length d of q_j. • $(\gamma_1, \ldots, \gamma_T) = \text{softmax}(\alpha_1, \ldots, \alpha_T)$ is the normalized attention, adding up to 1.0. • $\tilde{x}_j = \gamma_1 v_1 + \cdots + \gamma_T v_T$ is the new contextualized embedding of the token v_t at position j. The new embedding $\tilde{x}_j$ is context-sensitive or contextual because it includes information from the other embeddings $x_1, \ldots, x_T$. The computation is repeated for all $j = 1, \ldots, T$.
Self-supervised Learning	Learning task where the data is not annotated. Usually the relationship between the components of the input (vector, sequence, image) is modeled.
Sequence Model	Deep Neural network to process a sequence of items, e.g. words, speech phonemes, musical notes, etc.
Sequence-to-Sequence Model	A model for translating an input sequence into a corresponding output sequence. An important application is the translation of a text from one language to another. • The input sequence is encoded into a hidden vector by an Encoder RNN. • The hidden vector is transformed into the output sequence by a Decoder RNN. The logistic regression model calculates the probability of the output words at each position. Training criterion: generate the actual output sequence words with the highest possible probability.
Softmax Function	A function that transforms any vector $u = (u_1, \ldots, u_k)$ into a probability vector. $\text{softmax}(u) = \frac{(\exp(u_1), \ldots, \exp(u_k))}{\exp(u_1) + \cdots + \exp(u_k)}$
Stochastic Gradient Descent	Variant of gradient descent optimization in which the gradient is computed only for a minibatch of data and not for all training data. • Low computational cost. • Gradient varies randomly around the correct value, therefore the method can overcome local minima. The procedure usually converges much faster than computing the gradient for the entire training set. Hyperparameters are the size of the minibatch and the learning rate.

Stochastic Policy	A nondeterministic policy in reinforcement learning that randomly selects an action a in a state s according to a probability distribution $\pi(a\mid s)$ over the actions. • Given a state s_t, randomly generates an action $a_t \sim \pi(a\mid s_t)$. • Often better than a deterministic policy: e.g., in the game rock-paper-scissors.
Supervised learning	There is a target variable y, whose values are to be predicted based on the input variable x. • There are pairs (x, y) for the input variable x and the target variable y in the training set. • The model describing is the probability distribution $p(y\mid x; w)$ of y given input x is trained by modifying the parameter w.
Tensor	A one-, two-, three-, or higher dimensional arrangement of real numbers, e.g. $(a_{i,j,k})_{i=1,\ldots,r;j=1,\ldots,s;k=1,\ldots,t} \in \Re^{r\times s\times t}$ also called multidimensional array. Examples are scalars, vectors, matrices, etc.
Test Set	A set of observed and possibly manually labeled (annotated) examples $(x, y)^{(1)}, \ldots, (x, y)^{(r)}$ that is not used for training, but only to determine the performance of a model. • The test set may only be used once at the end of model building and the results may not be used to improve the model. If the test set has the same distribution as the future application data, the accuracy on the test data is transferable to the application data.
Token	Tokens are elements of a limited vocabulary of frequent words, word parts and letters. As arbitrary words can be represented by tokens they avoid the problem of missing words in the application of NLP models. Can be generated with specific modules, e.g. Byte-Pair Encoding.
Training	Modifies the parameter vector w of the model so that for the examples of the training set, the difference between predicted outputs (e.g., class labels) and observed outputs becomes as small as possible.
Training Set	A set of observed and possibly manually annotated training examples (x, y) forming the TrainSet $= \left\{(x, y)^{(1)}, \ldots, (x, y)^{(n)}\right\}$, which is **only** used for training. Typically, the largest part of the data (60–90%).
Transfer Learning	A model is trained with several datasets: • First the model is trained for an unsupervised learning task with a very large set of training data. • Then, the trained model is fine-tuned for a different, but similar, learning task. The training set is usually relatively small. If the model has many parameters it can then often transfer knowledge from the initial learning task to the new learning task and yield a model with high accuracy, even though only a small amount of training data was available for fine-tuning.

Transformer	A sequence processing network, which transforms an input sequence into an output sequence, e.g. for translating a text to another language. It has the following modules: • k Transformer encoder blocks sequentially transform the input sequence to contextual-embeddings. • k Transformer decoder blocks predict the embedding of the next token v_t of the output sequence by using the previous tokens $v_1, \dots, v_{t-1}$ as inputs. By cross-attention the information from the last embedding layer of the Transformer encoder is included. • A logistic regression model predicts the probability of the next output token using the output embedding at position t of the Transformer decoder. The output tokens are predicted sequentially. The original Transformer has $k = 6$ encoder and decoder blocks.
Transformer decoder block	A module of the Transformer decoder consisting of several layers: • A multi-head self-attention layer transforming embeddings $(z_1, \dots, z_m)$ of the output sequence into contextual-embeddings $(\tilde{z}_1 \dots, \tilde{z}_m)$ • A multi-head cross-attention layer using the embeddings $(\tilde{x}_1 \dots, \tilde{x}_T)$ of the last encoder layer to adapt the embeddings of the output sequence $(\tilde{z}_1 \dots, \tilde{z}_m)$. The results of the different multi-head cross-attention cross-attention operators are concatenated. • A fully connected layer. • Around the layers, residual connections are used, which improve optimization convergence. • After each layer the output is standardized by batch normalization. The Transformer decoder is applied to the subsequence $v_1, \dots, v_m$ of already predicted tokens. The output layer of the Transformer decoder is used to predict the next token v_{m+1} of the output sequence.
Transformer encoder block	A module of the Transformer encoder consisting of two layers: • A multi-head self-attention layer transforming input embeddings $(x_1, \dots, x_T)$ into contextual-embeddings $(\tilde{x}_1 \dots, \tilde{x}_T)$. The results of the different self-attention heads are concatenated. • A fully connected layer. • Around both layers, a residual connection is used, which improves optimization convergence. • After each layer the output is normalized by batch normalization. These Transformer encoder blocks are building blocks of a large number of NLP models, e.g. BERT.
Unsupervised Learning	The data for training the model is not labeled. • There is no target variable, only input variables. The statistical relationship between the variables is learned. • Self-supervised learning: learning task where the data is not annotated. The relationship between components of the input (vector, sequence, image) is modeled.

Validation set	A separate set of observed and, if necessary, manually labeled (annotated) training examples used only for measuring model quality during training. Used for selecting hyperparameters (e.g., number of layers).
Vanishing Gradient	After k applications of a recurrent network, a parameter w enters the gradient as w^k. • $w < 1$: the value of w^k becomes extremely small and a vanishing gradient results. • Optimizer does not consider long-range correlations when w is small. Remedy: special models such as LSTM.
Vector	A column of numbers, e.g. $\begin{pmatrix} 2.7 \\ 0.23 \\ -8.9 \end{pmatrix}$. A vector is a tensor.

A.3 List of Images and Their Sources

In this section, the image credits for the photos and illustrations used in the book have been compiled. All illustrations not specifically referenced here originate from the authors and were mostly revised by Svenja Niehus. The copyright of the self-created illustrations is held by the authors and Fraunhofer IAIS. Exceptions are illustrations that contain parts of other illustrations with Creative Commons license. For these images, the corresponding Creative Commons licenses of the partial images apply. Links to the Creative Commons image licenses are listed in the following image credits:

License	Link to license text
CC PDM 1.0	Public Domain
CC0 1.0	https://creativecommons.org/publicdomain/zero/1.0/deed.en
CC BY 2.0	https://creativecommons.org/licenses/by/2.0/
CC BY-ND 2.0	https://creativecommons.org/licenses/by-nd/2.0/
CC BY-SA 2.0	https://creativecommons.org/licenses/by-sa/2.0
CC BY-SA 2.0 de	https://creativecommons.org/licenses/by-sa/2.0/de/deed.en
CC BY 2.5	https://creativecommons.org/licenses/by/2.5/deed.tr
CC BY 3.0	https://creativecommons.org/licenses/by/3.0/
CC BY-SA 3.0	https://creativecommons.org/licenses/by-sa/3.0
CC BY-SA 3.0 AU	https://creativecommons.org/licenses/by-sa/3.0/au/
CC BY-SA 4.0	https://creativecommons.org/licenses/by-sa/4.0/
Pixabay License	https://pixabay.com/service/license/

Regulations of the German Copyright Law

German copyright law provides protection to authors with German citizenship for all their works, regardless of whether and where the works were published https://www.gesetze-im-internet.de/urhg/__120.html. According to the paragraph 51 of the German copyright law the reproduction, distribution and public communication of a published work for the purpose of quotation is permitted, provided that the extent of the use is justified by the particular purpose. In particular, this is permissible if

1. individual works are included in an independent scientific work after publication to explain the content,
2. passages of a work are cited after publication in an independent language work.

The right to quote according to sentences 1 and 2 includes the use of an illustration or other reproduction of the quoted work, even if this is itself protected by a copyright or related right https://www.gesetze-im-internet.de/urhg/__51.html.

Cover image	https://stock.adobe.com/de/images/white-cyborg-finger-about-to-touch-humanfinger-3d-rendering/204325271
Photo G. Paaß	Fotoatelier Herff, Bonn. http://www.atelier-herff.de/
Photo D. Hecker	Fraunhofer IAIS, 2019
Fig. 1.1	Runner: https://www.istockphoto.com/de/foto/die-nahaufnahme-derf%C3%BC%C3%9Fe-des-mannes-running-und-training-auf-laufbandgm924938754-253831773 cropped Staircase: Nilfanion: View down the spiral staircase in Haldon Belvedere, a folly on Haldon Hill, near Exeter, Devon. https://commons.wikimedia.org/wiki/File:Spiral_staircase_in_Haldon_Belvedere.jpg CC BY-SA 3.0 Woman and child: Adobe Stock: https://stock.adobe.com/de/images/mother-anddaughter-in-their-home-they-are-looking-at-coloring-book/167651778 cropped Guitarist: https://en.wikipedia.org/wiki/Ben_Carey Author: Justin M. Martinez, U.S. Marine Corps public domain cropped Mathematics: https://de.wikipedia.org/wiki/Edward_Frenkel#/media/File:Edward_Frenkel.jpg Author: Søren Fuglede Jørgensen Matematikeren Edward Frenkel under et foredrag om geometrisering af sporformler på University of California, Berkeley, 16. September 2010. CC BY-SA 3.0 cropped Interpersonal: Meaghan got hurt sliding (sort of) into home. Finn gave her a hug. She was safe. https://commons.wikimedia.org/wiki/File:There%27s_no_crying_in_baseball!_(4549295140).jpg by Jax House CC BY-SA 2.0 cropped Self-reflective: Maria Rantanen self reflection https://www.flickr.com/photos/idhren/4176198989 CC BY-SA 2.0 cropped Thunderstorm: Bidgee: Cloud to ground lightning strike during a dry thunderstorm near Wagga Wagga, Australia. https://en.wikipedia.org/wiki/Dry_thunderstorm#/media/File:Cloud_to_ground_lightning_strikes_south-west_of_Wagga_Wagga.jpg CC BY-SA 3.0 cropped.

Fig. 1.3	cat: https://stock.adobe.com/de/images/beautiful-short-hair-cat-lyingon-the-bed-at-home/247439873?asset_id=247439873 cropped.
Fig. 1.4	Monkey: Male chimpanzee by EBFoto https://stock.adobe.com/de/images/male-chimpanzee/103531598 cat: https://stock.adobe.com/de/images/beautiful-short-hair-cat-lyingon-the-bed-at-home/247439873?asset_id=247439873 cropped. Woman: Cheerful young African woman smiling by mimagephotos https://stock.adobe.com/de/images/cheerful-young-african-woman-smiling/56513248 cropped
Fig. 1.5	Photo Adobe Stock: https://stock.adobe.com/de/images/mother-and-daughter-intheir-home-they-are-looking-at-coloring-book/167651778 cropped
Fig. 1.6	Albert Einstein 1921. https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png. License: Public Domain. Images of features: Lee et al. (2011).
Fig. 1.7	Albert Einstein 1921. https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png. License: Public Domain.
Fig. 1.9	https://www.istockphoto.com/de/foto/transparente-fahrzeuggm154411679-16663939
Fig. 1.10	Car: Oliver Kurmis Sportwagen ‘Porsche 911’, Modell SC 1986 https://de.wikipedia.org/wiki/Automobil#/media/File:1986_Porsche_911_SC.jpg CC BY 3.0. Images of features: Lee et al. (2011).
Fig. 1.11	Self-created graphic with: Photo of Turing: Turing as a 16-year-old. Public domain https://de.wikipedia.org/wiki/Alan_Turing#/media/Datei:Alan_Turing_Aged_16.jpg Expert system Dendral: Lindsay et al. (1993). Photo McCarthy: Professor “Uncle John” McCarthy discusses LISP, by null0 from Singapore, CC BY 2.0 https://commons.wikimedia.org/wiki/File:John_McCarthy_(computer_scientist)_Stanford_2006_(272020300).jpg?uselang=de Photo cat/Krakenimages.com: https://stock.adobe.com/de/images/beautiful-shorthair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873 cropped.
Fig. 1.12	OCR: https://stock.adobe.com/de/images/businessman-taking-photo-of-billwith-mobile-phone/216276055 Chess: https://stock.adobe.com/de/117711879 Face localization: https://stock.adobe.com/de/300883540 changed Share price forecast: Adobe Stock: https://stock.adobe.com/de/images/stock-market-data/83882611 Handwriting recognition: Self-created graphic
Fig. 2.1	Krizhevsky et al. (2012) with self-created graphic
Fig. 2.2	upper row and lower left is public domain. https://commons.wikimedia.org/wiki/File:Firm_red_skin_lump.jpg https://commons.wikimedia.org/wiki/File:Crusty_skin_lump.jpg https://commons.wikimedia.org/w/index.php?title=Special:Search&limit=50&offset=0&profile=default&search=skin+cancer&advancedSearch-current=&ns0=1&ns6=1&ns12=1&ns14=1&ns100=1&ns106=1#/media/File:Melanomawith_color_differences.jpg https://en.wikipedia.org/wiki/Nevus#/media/File:Spitz_nevus.jpg

	Lower row middle: Jmarchn Becker’s nevus, right shoulder https://en.wikipedia.org/wiki/Nevus#/media/File:Becker’s_nevus_of_shoulder.JPG CC BY-SA-3.0 cropped.
	lower row right: Mathew Bellemare https://commons.wikimedia.org/wiki/File:51_105_Leg_Nevus_(149029661).jpeg CC BY-SA 3.0 cropped.
Fig. 2.3	Frames from video https://deepmind.com/blog/article/alphafold
Fig. 2.5	https://www.istockphoto.com/de/foto/3d-rendering-piktogrammspracherkennungssystem-von-blauem-grund-gm683287514-125386755
Fig. 2.7	Photos: ©WTO/ Cuika Foto. World Trade Organization Heads of Delegation meeting, 11 December https://de.wikipedia.org/wiki/Simultandolmetschen#/media/Datei:Heads_of_Delegation_meeting,_11_December_(38130168095).jpg CC BY-SA-2.0 cropped
Fig. 2.8	By Rosemaryetoufee “Watson Jeopardy demo” https://kids.kiddle.co/Image:Watson_Jeopardy_demo.jpg CC BY-SA 4.0 cropped
Fig. 2.9	By Myrabella https://de.wikipedia.org/wiki/Schlacht_bei_Hastings#/media/Datei:Bayeux_Tapestry_scene55_William_Hastings_battlefield.jpg public domain
Fig. 2.12	IBM Research IBM Project Debater with Dan Zafrir https://www.flickr.com/photos/ibm_research_zurich/45818968004 CC BY-ND 2.0
Fig. 2.13	By Jürg Vollmer “Schachweltmeister Garri Kasparov aus Baku/Aserbeidschan” 26.08.2009. CC BY-SA 2.0. https://www.flickr.com/photos/maiakinfo/3859735342/
Fig. 2.14	Buster Benson 8:36pm Match 3 of AlphaGo vs Lee Sedol. The confidence of the human commentary is fascinating. CC BY-SA 2.0. https://www.flickr.com/photos/erikbenson/25717574115/
Fig. 2.15	https://stock.adobe.com/de/images/people-play-poker-at-the-table-in-thecasino/294179973
Fig. 2.16	Own snapshots from public domain Stella emulator
Fig. 2.17	Cropped from https://www.theverge.com/2018/7/4/17533898/deepmind-aiagent-video-game-quake-iii-capture-the-flag
Fig. 2.18	https://www.theverge.com/2018/8/6/17655086/dota2-openai-bots-professionalgaming-ai
Fig. 2.19	https://commons.wikimedia.org/wiki/File:BeerBottlePass.JPG public domain
Fig. 2.20	Anthony Kendall: This is VW’s entry in the Darpa Grand Challenge. https://www.flickr.com/photos/80546870@N00/87625236 CC BY-SA 2.0
Fig. 2.21	Adobe Stock/Andrei https://www.istockphoto.com/de/foto/waymoselbstfahrendes-auto-f%C3%BChrt-tests-auf-einer-stra%C3%9Fe-in-dern%C3%A4he-von-googles-b%C3%BCros-gm1162994767-319188671
Fig. 2.22	Construction worker in manhole. Image: Waymo https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-carsdeep-learning-neural-net-interview
Fig. 2.23	https://de.wikipedia.org/wiki/Portrait_of_Edmond_de_Belamy#/media/Datei:Edmond_de_Belamy.pngpublicdomain
Fig. 2.24	Karras (2019) https://thispersondoesnotexist.com/ public domain. See https://news.ycombinator.com/item?id=19144280

Fig. 3.1	https://www.istockphoto.com/de/foto/eine-gruppe-von-studenten-auf-dieanatomie-klasse-in-hoch-schule-gm519589014-90619453
Fig. 3.2	“Defying Gravity” by Andrew Newill is licensed under CC BY-SA 2.0 https://ccsearch.creativecommons.org/photos/157c3159-14ef-40ff-bdc1-c394ea5006ac
Fig. 3.3	https://stock.adobe.com/de/search?serie_id=206970868&asset_id=206970923
Fig. 3.4	Self-created graphic. House: https://www.pexels.com/photo/home-real-estate-106399/ free to use.
Fig. 3.7	Self-created graphic based on self-rendered MNIST-Data
Fig. 3.8	https://www.researchgate.net/figure/Selected-difficult-examples-from-theMNIST-database_fig2_220914543
Fig. 3.10	Self-created graphic based on self-rendered MNIST-Data
Fig. 3.22	Edna Winti: Above it all. CC BY 2.0 https://www.flickr.com/photos/ednawinti/45068960405/
Fig. 4.4	Self-created graphics except: Perceptron: The Mark I Perceptron, an early pattern recognition system, at the Cornell Aeronautical Laboratory. Fair Use license. https://en.wikipedia.org/wiki/Perceptron#/media/File:Mark_I_perceptron.jpeg Krizhevsky: Nvidia Tesla 2075 by Raysonho @ Open Grid Scheduler/Grid Engine – Eigenes Werk CC0 1.0 https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG
Fig. 4.3	By Sethwoodworth: Marvin Minsky was visiting the OLPC offices and picked up a Firefox wrist band. https://en.wikipedia.org/wiki/Marvin_Minsky#/media/File:Marvin_Minsky_at_OLPCb.jpg CC BY 3.0
Fig. 4.10	Self-created graphic. House: https://stock.adobe.com/de/290172089
Fig. 4.13	Eviatar Bach: “Geoffrey Hinton giving a lecture about deep neural networks at the University of British Columbia” https://en.wikipedia.org/wiki/Geoffrey_Hinton#/media/File:Geoffrey_Hinton_at_UBC.jpg cropped. CC BY-SA 3.0
Fig. 4.14	https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG CC0 public domain
Fig. 4.21	Self-rendered letters from the notMNIST dataset. https://www.kaggle.com/jwjohnson314/notmnist CC0 Public Domain
Fig. 4.27	Self-created graphic. Chess https://www.istockphoto.com/de/foto/opening-gameof-chess-gm586376700-100670363
Fig. 4.31	AnotherSamWilson: Bayesian optimization of a function (black) with Gaussian processes (purple). Three acquisition functions (blue) are shown at the bottom. https://en.wikipedia.org/wiki/Bayesian_optimization#/media/File:GpParBayesAnimationSmall.gif CC BY-SA 4.0
Fig. 4.33	Noy et al. (2020)
Fig. 4.34	https://de.wikipedia.org/wiki/Neuronales_Netz#/media/Datei:Neurons_big1.jpg public domain
Fig. 5.1	Self drawn bounding boxes: photos: https://www.istockphoto.com/de/foto/bengal-kitten-in-blumenwiesegm905117504-249578201 https://stock.adobe.com/de/images/domestic-animalschicken-dog-and-cat-eating-together-as-best-friend/221661292
Fig. 5.2	Brain: _DJ_ human brain on white background CC BY-SA 2.0 https://www.flickr.com/photos/flamephoenix1991/8376271918 Eye: Human eye https://www.istockphoto.com/de/foto/menschlichesauge-gm91170533-6467713

Fig. 5.3	Self-created graphic: cat: https://stock.adobe.com/de/images/beautiful-short-haircat-lying-on-the-bed-at-home/247439873?asset_id=247439873
Fig. 5.4	Kirsch: https://en.wikipedia.org/wiki/Russell_A._Kirsch#/media/File:NBSFirstScanImage.jpg public domain Viola&Jones: https://de.wikipedia.org/wiki/Viola-Jones-Methode#/media/Datei:Face_detection.jpg Jimmy answering questions.jpg: Wikimania2009 Beatrice Murch derivative work: Sylenius (Diskussion) – Jimmy answering questions.jpg. Die Viola-Jones-Methode hat drei der vier Gesichter erkannt. CC BY 3.0 cropped Leopard: https://www.flickr.com/photos/116174993@N02/30708142115 "EA0A7014 Kopie" by harald 65 is licensed under CC BY 2.0 cropped
Fig. 5.5	cat1: Daniel Spiess: Cats. If you want to use this photo, please credit DJ Spiess and link to www.fermentarium.com https://www.flickr.com/photos/deegephotos/6190172658 (CC BY-SA 2.0) cropped cat2: Rainer Stropek: Cat "Cat looking directly into the camera" https://www.flickr.com/photos/rainerstropek/16075613156 (CC BY 2.0) cropped, cat3: Xiahong Chen: cat https://www.flickr.com/photos/101210291@N05/9665358037/ (CC BY-SA 2.0) cropped, cat4:@ S@ndrine: Mon 'tit père aux aguets! https://www.flickr.com/photos/neelsandrine/32019326474/ (CC BY 2.0) cropped, cat5: Mike Steinhoff: invisibility cloak https://www.flickr.com/photos/-aismist/24154229240 (CC BY 2.0) cropped, cat6: Mokeneco: sweet https://www.flickr.com/photos/quinnanya/110987103 (CC BY-SA 2.0) cropped, Bridge1: Dave Kavanagh: "Quin bridge" https://www.flickr.com/photos/142089943@N07/27647871434 public domain cropped, Bridge2: teofilo: Bridge https://www.flickr.com/photos/teofilo/2780744203 (CC BY 2.0) cropped, Bridge3: Brandon Giesbrecht: University Bridge https://www.flickr.com/photos/naturegeak/6242747880 (CC BY 2.0) cropped, Bridge4: Neil Turner: Railway bridge. This carries the transpennine route over the River Calder on the edge of Dewsbury. https://www.flickr.com/photos/neilt/3351424070 (CC BY-SA 2.0) cropped, Bridge5: Nick Amoscato: Bridge https://www.flickr.com/photos/namoscato/15024333421 (CC BY 2.0) cropped Bridge6: Neo_II London Bridge https://www.flickr.com/photos/neo_ii/5344966712 (CC BY-SA 2.0) cropped
Fig. 5.7	https://de.wikipedia.org/wiki/Laplace-Filter#/media/Datei:Laplace_beispiel.png public domain
Fig. 5.14	https://en.wikipedia.org/wiki/Yann_LeCun#/media/File:Yann_LeCun__2018_(cropped).jpg cropped. "Conference of French computer scientist Yann LeCun, director of Facebook AI Research, at the École Polytechnique" Jérémy Barande CC BY-SA 2.0
Fig. 5.15	Wu et al. (2018)
Fig. 5.18	He (2016)
Fig. 5.17	Self-created graphic Li et al. (2018).
Fig. 5.20	Self-created graphic. Woman: https://stock.adobe.com/de/images/cheerful-young-africanwoman-smiling/56513248

Fig. 5.21	Huang et al. (2017) Self-created graphic. Horse: https://stock.adobe.com/de/images/pferd-weisser-hintergrund/113638638
Fig. 5.22	Touvron et al. (2019) with translations
Fig. 5.23	Zeiler and Fergus (2014)
Fig. 5.24	Zeiler and Fergus (2014)
Fig. 5.25	Krizhevsky et al. (2012)
Fig. 5.26	Van Horn et al. (2018)
Fig. 5.27	Esteva et al. (2017)
Fig. 5.28	Grover (2018)
Fig. 5.29	Redmon and Farhadi (2017)
Fig. 5.30	Chen et al. (2019, p. 841)
Fig. 5.32	Ronneberger et al. (2015), edited
Fig. 5.34	Zhu et al. (2019)
Fig. 5.35	Eslami et al. (2018)
Fig. 5.36	Phillips et al. (2018)
Fig. 5.37	Wang et al. (2019)
Fig. 5.38	Self-created graphic. dog1: Mariofan13—Own work Apricotfarbener Königspudel https://en.wikipedia.org/wiki/Hypoallergenic_dog_breed#/media/File:K%C3%B6nigspudel_Apricot.JPG CC BY-SA 4.0 cropped dog2: Chris Barber: A terrier-type mixed-breed dog. Rosie https://en.wikipedia.org/wiki/File:Terrier_mixed-breed_dog.jpg (CC BY 2.0) dog3: Jason Oatman: Sambo, a black Labrador Retriever https://en.wikipedia.org/wiki/File:Black_Labrador_Retriever_portrait.jpg (CC BY-SA 3.0) cropped cat: Bertil Videt: Van Cat, male kitten. https://tr.wikipedia.org/wiki/Dosya:Turkish_Van_Cat.jpg (CC BY-SA 3.0) cropped
Fig. 5.39	Self-created graphic. dog1: Mariofan13 - Own work Apricotfarbener Königspudel https://en.wikipedia.org/wiki/Hypoallergenic_dog_breed#/media/File:K%C3%B6nigspudel_Apricot.JPG CC BY-SA 4.0 cropped cat: Rainer Stropek: Cat Cat looking directly into the camera https://www.flickr.com/photos/rainerstropek/16075613156 (CC BY 2.0) cropped, dog 2: Jason Oatman: Sambo, a black Labrador Retriever https://en.wikipedia.org/wiki/File:Black_Labrador_Retriever_portrait.jpg (CC BY-SA 3.0) cropped Bertil Videt: Van Cat, male kitten. https://tr.wikipedia.org/wiki/Dosya:Turkish_Van_Cat.jpg (CC BY-SA 3.0) cropped
Fig. 5.42	Teye et al. (2018)
Fig. 5.43	Teye et al. (2018)
Fig. 5.44	Hendrycks and Dietterich (2019)
Fig. 5.45	Hendrycks and Dietterich (2019)
Fig. 5.46	Eykholt et al. (2018)
Fig. 5.47	Eykholt et al. (2018)
Fig. 5.48	Szegedy et al. (2013)
Fig. 6.1	Bovee and Thill "Business Technology" https://www.flickr.com/photos/bovee_thill/39475182975 License: CC BY 2.0

Fig. 6.4	Self-created graphic. https://www.svenkreiss.com/blog/word2vec-on-databricks/
Fig. 6.5	Lime: public domain from https://simple.wikipedia.org/wiki/Key_lime#/media/File:Key_lime.jpg Almju – Naranja de gran tamaño cosechada en el Bajo Andarax, Almería. https://es.wikipedia.org/wiki/Naranja_(fruta)#/media/Archivo:Naranja_Almeria.JPG CC BY-SA 4.0 Evan-Amos Lemon external surface and cross-section https://commons.wikimedia.org/wiki/File:Lemon-Whole-Split.jpg public domain
Fig. 6.8	Jérémy Barande: “Kick-off / Deep Learning”, par Yoshua Bengio de l’Université de Montréal - Data Science Summer School organisée à l’École Polytechnique du 28 août au 1er septembre. https://commons.wikimedia.org/wiki/Category:Yoshua_Bengio?uselang=de#/media/File:Yoshua_Bengio_-_2017.jpg CC BY-SA-2.0
Fig. 6.18	Music notes: https://www.cantus.org/notenarchiv/files/d872-4.pdf, public domain DNA: https://stock.adobe.com/de/images/digital-illustration-of-a-dna-model3d-rendering/140436292 Sound: https://stock.adobe.com/de/images/blue-audio-waveformbackground/85619893
Fig. 6.24	ITU/R.Farrell https://commons.wikimedia.org/wiki/File:J%C3%BCrgen_Schmidhuber.jpg CC BY 2.0
Fig. 6.31	Text. Source: https://de.wikipedia.org/wiki/Joseph_von_Fraunhofer
Fig. 6.33	Stock Market Data by James Thew https://stock.adobe.com/de/images/stock-market-data/83882611
Fig. 6.34	Strobelt et al. (2016)
Fig. 6.38	Text Sutskever et al. (2014)
Fig. 6.39	Table: Sutskever et al. (2014)
Fig. 6.40	Self-created graphic Source: Sutskever et al. (2014)
Fig. 6.42	Self-created graphic: https://devblogs.nvidia.com/introduction-neural-machinetranslation-gpus-part-3/
Fig. 6.43	Table: Source: Bahdanau et al. (2015)
Fig. 6.44	Source: Bahdanau et al. (2015)
Fig. 6.45	Source: http://wmt.ufal.cz/
Fig. 6.46	Britz et al. (2017)
Fig. 6.48	Self-created graphic: Source: https://ricardokleinklein.github.io/2017/11/16/Attentionis-all-you-need.html Source: Vaswani et al. (2017)
Fig. 6.65	Self-created graphic: Vaswani et al. (2017)
Fig. 6.47	Text: Sennrich et al. (2016)
Fig. 6.68	Self-created graphic. Source: https://ricardokleinklein.github.io/2017/11/16/Attention
Fig. 6.69	Popel and Bojar (2018)
Fig. 6.70	Businessman standing and holding speaker to speak in the auditorium. Source https://stock.adobe.com/de/285635048
Fig. 6.71	Self-created graphic: Source: Jia et al. (2019)
Fig. 6.50	Source: Rectified linear unit and Gaussian error linear unit activation functions by Ringdongdang https://en.wikipedia.org/wiki/Activation_function#/media/File:ReLU_and_GELU.svg licensed by CC BY-SA 4.0

Fig. 6.56	Table MultiNLI (2017)
Fig. 6.57	Table Devlin et al. (2018)
Fig. 6.58	Source: https://en.wikipedia.org/wiki/Genghis_Khan public domain
Fig. 6.59	California National Guard: California's Soldiers and Airmen help extinguish raging wildfire. 24.08.2012. CC BY 2.0.https://www.flickr.com/photos/caguard/7880896072
Fig. 6.61	https://en.wikipedia.org/wiki/Robin_Spielberg#/media/File:Robin_Spielberg.jpg Photograph of Robin Spielberg Larry Kosson CC BY-SA 3.0
Fig. 6.73	Kudugunta et al. (2019)
Fig. 6.74	Self-created graphic: Shazeer et al. (2017)
Fig. 6.75	https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html
Fig. 6.63	Self-created graphic Nayak (2019)
Fig. 6.76	http://cocodataset.org/#explore?id=195696, http://cocodataset.org/#termsofuse
Fig. 6.77	Self-created graphic using DJHeini: Some friends playing frisbee on the beach. IMG_0531 https://www.flickr.com/photos/74837817@N00/4880595155 CC BY-SA 2.0 cropped
Fig. 6.78	Sharma et al. (2018)
Fig. 6.81	Self-created graphic: Ribeiro et al. (2016)
Fig. 6.82	Lapuschkin et al. (2019)
Fig. 6.83	Table: Karpukhin et al. (2019)
Fig. 6.85	Self-created graphic: https://gluebenchmark.com/leaderboard
Fig. 7.2	Rburtonresearch: This is a picture of a wave file captioned with pressure and time as the 'X' and 'Y' coordinates. https://en.wikipedia.org/wiki/Sound#/media/File:The_Elements_of_Sound_jpg.jpg translated CC BY-SA 4.0
Fig. 7.3	https://de.wikipedia.org/wiki/Fourier-Analysis#/media/Datei:Fourier_transform_time_and_frequency_domains_(small).gif. public domain
Fig. 7.6	Source: https://archive.org/details/SwitchboardCorpusSample
Fig. 7.7	Dragon: https://de.wikipedia.org/wiki/Dragon_NaturallySpeaking#/media/Datei:Dragon_Naturally_Speaking_Logo.png public domain Amazon Echo: Michael Sheehan The black Amazon Echo Dot (second generation) sitting idle on a wood surface. https://www.flickr.com/photos/hightechdad/33309161691/ CC BY 2.0 Ibm shoebox: https://www.wired.com/2013/11/tech-time-warp-of-the-weekibms-speech-savvy-computer-1986/ Tangora: https://www.wired.com/2013/11/tech-time-warp-of-the-week-ibmsspeech-savvy-computer-1986/ https://stackoverflow.com/questions/35169706/audio-feature-for-deeplearning/39624704 Tangora and Shoebox Courtuosy of IBM.
Fig. 7.9	Source: Chan et al. (2016)
Fig. 7.12	Self-created graphic. Source: Li et al. (2019)
Fig. 7.15	Source: https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html
Fig. 7.16	Self-created graphic: Chung et al. (2017)
Fig. 7.17	Source: http://en.wikipedia.org/wiki/File:Signal_Sampling.png public domain
Fig. 7.21	Source: "Happy woman talking on the phone" by Rawpixel Ltd https://www.flickr.com/photos/147875007@N03/30849323977 was cropped and is licensed under CC BY 2.0

Fig. 7.22	Sources: (left) Maurizio Pesce “Android Assistant on the Google Pixel XL smartphone” https://www.flickr.com/photos/pestoverde/29526761674/ CC BY 2.0. cropped. (middle): Frmorrison at English Wikipedia “This is the Amazon Echo appliance that is basically a talking speaker” https://en.wikipedia.org/wiki/Virtual_assistant#/media/File:Amazon_Echo.jpg CC BY-SA 3.0. (right) “Ausblick auf iOS 7” by FalkoMD is licensed under CC BY-SA 2.0 https://www.flickr.com/photos/95190793@N08/9339721523 cropped
Fig. 7.25	Table. Source: https://web.stanford.edu/~jurafsky/ws97/manual.august1.html
Fig. 7.28	Source: https://arxiv.org/pdf/1609.06782.pdf
Fig. 7.29	Source: Monfort et al. (2019)
Fig. 7.30	Image frame from Video: https://www.youtube.com/watch?v=x9AhZLDkbyc
Fig. 7.31	Self-created graphic: Frames out of video Henry Stober The Big 5 https://vimeo.com/239953264 (CC BY-SA 3.0)
Fig. 7.32	Self-created graphic. Frames out of video Source: Henry Stober The Big 5 https://vimeo.com/239953264 (CC BY-SA 3.0)
Fig. 7.33	Simonyan and Zisserman (2014)
Fig. 7.34	Self-created graphic Zhou et al. (2018)
Fig. 7.35	Zhou et al. (2018)
Fig. 7.36	Sun et al. (2019)
Fig. 7.37	Sun et al. (2019)
Fig. 7.38	Schönherr et al. (2019)
Fig. 8.1	https://www.istockphoto.com/de/foto/junge-w%C3%A4rmt-seineh%C3%A4nde-in-der-n%C3%A4he-von-lagerfeuer-im-waldgm670835208-122726195
Fig. 8.2	Source: Self-driving car https://www.istockphoto.com/de/foto/frau-mit-smartphone-inautonome-auto-selbstfahrer-fahrzeug-autopilot-kfz-technikgm829192848-134881835 Andyhill8 – Andy Hill https://en.wikipedia.org/wiki/Stockbroker#/media/File:Stockbroker.jpg public domain, cropped
Fig. 8.3	Chess: Mukumbura: Chess pieces with colorful bokeh https://commons.wikimedia.org/wiki/File:Chess_pieces_bokeh.jpg CC BY-SA 2.0. Seaquest: Own snapshots taken from public domain Stella Emulator
Fig. 8.4	Self-created graphic. Katrina.Tuliao: Phillippine stock market board https://commons.wikimedia.org/wiki/File:Philippine-stock-market-board.jpg (CC BY 2.0) cropped Andy Hill: Stockbroker https://commons.wikimedia.org/wiki/File:Stockbroker.jpg public domain, cropped
Fig. 8.5	Snarc: http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minskyedmonds-american/ Shakey: SRI International Shakey in 1972 CC BY-SA 3.0 https://en.wikipedia.org/wiki/Shakey_the_robot#/media/File:SRI_Shakey_with_callouts.jpg Dickmann: https://www.lifehacker.com.au/2016/02/creator-of-the-worlds-firstself-driving-cars-ernst-dickmanns/ Backgammon: Bdevel, took this picture on Sept. 19 2005 and grant its use under the GNU Free Documentation License. https://de.wikipedia.org/wiki/Datei:Backgammon_board.jpg Chess: https://pixabay.com/de/photos/schach-schachfiguren-schachspiel-335141/ Pixabay License

	Darpa: Anthony Kendall This is VW's entry in the Darpa Grand Challenge. https://www.flickr.com/photos/80546870@N00/87625236 CC BY-SA 2.0 Go: https://stock.adobe.com/de/editorial/chinese-go-player-ke-jie-reacts-duringhis-first-match-with-google-s-artificial-intelligence-program-alphago-at-thefuture-of-go-summit-in-wuzhen/155279016?prev_url=detail
Fig. 8.11	Pong, Breakout: Own snapshots of the public domain stella simulator.
Fig. 8.12	Osband et al. (2016)
Fig. 8.13	Own snapshots of the public domain stella simulator.
Fig. 8.16	Mnih et al. (2015) cropped
Fig. 8.17	Source: Own snapshots of the public domain stella simulator.
Fig. 8.18	james bamber 4th UK Rock Paper Scissors Championships https://commons.wikimedia.org/wiki/File:Rps2010_2.jpg public domain
Fig. 8.20	Levine (2018)
Fig. 8.21	Richard S. Sutton 2016 Steve Jurvetson from Menlo Park, USA – Deep Thinkers on Deep Learning https://de.wikipedia.org/wiki/Richard_S._Sutton#/media/Datei:Richard_Sutton,_October_27,_2016.jpg CC BY 2.0
Fig. 8.22	Source: Roman Fuchs: Path to Angels Landing – Zion National Park. https://de.wikipedia.org/wiki/Datei:Path_to_Angels_Landing.jpg CC BY 3.0 cropped
Fig. 8.23	Source: Schulman et al. (2015)
Fig. 8.24	Schulman et al. (2015)
Fig. 8.26	Source: https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister1904-140710.html
Fig. 8.27	World Economic Forum: https://commons.wikimedia.org/wiki/File:Sebastian_Thrun_World_Economic_Forum_2013.jpg DAVOS/SWITZERLAND, 23JAN13 – Sebastian Thrun, Research Professor of Computer Science, Stanford University, USA talks during the interactive session 'Connected Transportation: Hype or Reality' at the Annual Meeting 2013 of the World Economic Forum in Davos, Switzerland, January 23, 2013. Copyright by World Economic Forum. swiss-image.ch/Photo Urs Jaudas CC BY-SA 2.0
Fig. 8.28	Fridman (2019)
Fig. 8.29	Fridman (2019)
Fig. 8.30	Fridman (2019)
Fig. 8.31	Fridman (2019)
Fig. 8.32	Self-created graphic Giacaglia (2019)
Fig. 8.33	Self-created graphic: Sources: https://www.machinedesign.com/motion-control/saved-sensor-vehicle-awareness-self-driving-age, https://commons.wikimedia.org/wiki/File:C3top_svg.svg
Fig. 8.34	https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungengm938038454-256530428 https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungengm938038442-256530421
Fig. 8.35	Anguelov (2019)
Fig. 8.36	Anguelov (2019)

Fig. 8.37	Source: Fridman (2019)
Fig. 8.38	Pinto et al. (2017)
Fig. 8.39	Source: Chen et al. (2019)
Fig. 9.1	Self-created graphic https://commons.wikimedia.org/wiki/Category:Paintings_by_Vincent_van_Gogh_by_title?uselang=de public domain Image1: “Vincent van Gogh (1853–1890), Irises, May 1890” by Tulip Hysteria/Go to albums is licensed under CC PDM 1.0 https://ccsearch.creativecommons.org/photos/c71e11ef-3a38-4201-8b62-347c7a5fef05 , Image2: “Self Portrait 1887” is licensed under CC PDM 1.0 https://commons.wikimedia.org/wiki/File:Vincent_van_Gogh_-_Self-Portrait_-_1954.326_-_Art_Institute_of_Chicago.jpg?uselang=de public domain, Image3: Vincent van Gogh - Self-portrait with bandaged ear (1889, Courtauld Institute) licensed under CC PDM 1.0 https://en.wikipedia.org/wiki/Self-Portrait_with_Bandaged_Ear#/media/File:Vincent_van_Gogh_-_Self-portrait_with_bandaged_ear_(1889,_Courtauld_Institute).jpg , Image4: “Roses (1890) by Vincent Van Gogh. Original from the MET Museum. is licensed under CC BY 2.0 https://en.wikipedia.org/wiki/File:Roses_-_Vincent_van_Gogh.JPG , Image5: “Absinthe_Van Gogh Museum” by E V E is licensed under CC BY-SA 2.0 https://de.wikipedia.org/wiki/Datei:Van_Gogh_-_Still_Life_with_Absinthe.jpg Image6: Portrait of Félix Rey, January 1889, Pushkin Museum; public domain https://en.wikipedia.org/wiki/Vincent_van_Gogh#/media/File:Felix_Rey_portrait_&_sketch.jpg
Fig. 9.3	Source: Karras et al. (2018)
Fig. 9.4	Karras et al. (2018)
Fig. 9.5	Karras et al. (2018)
Fig. 9.6	Brock et al. (2019)
Fig. 9.7	Karras et al. (2018)
Fig. 9.8	Source: Radford et al. (2016)
Fig. 9.9	Source: Isola et al. (2017)
Fig. 9.10	Isola et al. (2017)
Fig. 9.11	Isola et al. (2017)
Fig. 9.12	Zhu et al. (2017)
Fig. 9.13	Wolf (2018)
Fig. 9.14	Elgammal et al. (2017)
Fig. 9.16	Elgammal et al. (2017)
Fig. 9.17	Zhang et al. (2017)
Fig. 9.18	Zhang et al. (2017)
Fig. 9.19	Vardaan (2019). Frames of a video
Fig. 9.22	Self-created graphic Fan et al. (2018)
Fig. 9.27	Source: Vig (2019)
Fig. 9.28	Dathathri et al. (2020)
Fig. 9.29	Text Source: Payne (2019)
Fig. 9.30	Self-created graphic, Source: Payne (2019)
Fig. 9.31	Source: Payne (2019)
Fig. 9.33	Source: Huang et al. (2019)

Fig. 9.35	Kanijoman: ex-machina-movie https://www.flickr.com/photos/23925401@N06/20167701293 (CC BY 2.0)
Fig. 9.37	Shum et al. (2018)
Fig. 9.38	Self-created graphic similar to Shum et al. (2018)
Fig. 9.41	https://www.flickr.com/photos/websummit/22762564236 CC BY 2.0 Picture credit: Diarmuid Greene/SPORTSFILE/Web Summit Description – 4 November 2015; Attendees from Coder Dojo, Fionnan O'Ceallaigh, Lucas Perez, and Lexi Schoene, interacting with 'Pepper' from Aldebaran Robotics in the Town Hall during Day 2 of the 2015 Web Summit in the RDS, Dublin, Ireland.
Fig. 9.42	Guo et al. (2018)
Fig. 9.43	Source: right: https://thispersondoesnotexist.com/, left: Elgammal et al. (2017)
Fig. 9.44	Source: https://towardsdatascience.com/openai-gpt-2-writes-alternate-endingsfor-game-of-thrones-c9be75cd2425
Fig. 10.1	self-created graphic, uses: The famous red eye of HAL 9000 by Cryteria – https://de.wikipedia.org/wiki/HAL_9000#/media/Datei:HAL9000.svg CC BY 3.0
Fig. 10.2	Image 2008: Raised hands in honor of a musical show on stage by 9parusnikov https://stock.adobe.com/de/266551332, Image 2012: People taking photographs with touch smart phone during a music entertainment public concert by verve https://stock.adobe.com/de/290346853 Image 2016: Group of fans taking selfie with smart phone while watching match in stadium. https://www.istockphoto.com/de/foto/zuschauergruppe-jubelt-imstadion-gm1131336410-299526968
Fig. 10.4	No more copyright protection 50 years after creation (1922) https://www.imdb.com/title/tt0013099/mediaviewer/rm320567040?ref_=ttmi_mi_all_sf_12
Fig. 10.5	Lung CA seen on CXR James Heilman, https://commons.wikimedia.org/wiki/File:LungCACXR.PNG (CC BY-SA 3.0)
Fig. 10.6	Magnetic resonance image of a knee by Test21 https://de.wikipedia.org/wiki/Magnetresonanztomographie#/media/Datei:Knie_mr.jpg CC BY-SA 3.0
Fig. 10.7	Doctor making a vaccination by guerrieroale https://stock.adobe.com/de/images/doctor-making-a-vaccination/150155835
Fig. 10.8	https://stock.adobe.com/de/images/young-disable-man-in-the-roboticexoskeleton-sitting-in-wheelchair-in-the-rehabilitation-clinic/227997068
Fig. 10.10	Author: BMW Werk Leipzig BMW plant in Leipzig, Germany: Spot welding of BMW 3 series car bodies with KUKA industrial robots. CC BY-SA 2.0 de https://en.wikipedia.org/wiki/Smart_manufacturing#/media/File:BMW_Leipzig_MEDIA_050719_Download_Karosseriebau_max.jpg
Fig. 10.12	Dyrmann et al. (2017)
Fig. 10.13	public domain https://commons.wikimedia.org/wiki/File:FEMA_-_34156_-_Preliminary_Damage_Assessment_Team_workin_in_Tennessee.jpg
Fig. 10.14	Grendelkhan: A Waymo self-driving car on the road in Mountain View, making a left turn. (Side view.) https://it.wikipedia.org/wiki/File:Waymo_self-driving_car_side_view.gk.jpg CC BY-SA 4.0
Fig. 10.15	Osborn et al. (2018)
Fig. 10.16	David Berkowitz Nest thermostat – Consumer Electronics Show – CES 2013 – Las Vegas https://www.flickr.com/photos/davidberkowitz/8381790944/ CC BY 2.0 cropped
Fig. 10.18	Wang et al. (2019)
Fig. 10.19	Kanellos (2016)

Fig. 10.20	Sean MacEntee: Amazon UK. https://www.flickr.com/photos/smemon/6500597907/ CC BY 2.0 cropped
Fig. 10.21	self-created graphic, Hosseini (2018)
Fig. 10.24	Shoham et al. (2018)
Fig. 10.25	self-created graphic, Source Kropp et al. (2018)
Fig. 10.26	https://commons.wikimedia.org/wiki/File:DJPatil.jpg public domain
Fig. 10.28	Datei-Nr.: 283969663 Competition between humans and robots in tug of war concept. by Elnur https://stock.adobe.com/de/images/competition-betweenhumans-and-robots-in-tug-of-war-concept/283969663
Fig. 10.29	Stock adobe.com: 215544030. Data Security For Doctor Providing Telemedicine https://stock.adobe.com/de/images/data-security-for-doctor-providing-telemedicine/215544030
Fig. 10.30	Roncheng's "civilized families" are displayed on public noticeboards like these. (Simina Mistreanu) https://foreignpolicy.com/2018/04/03/life-inside-chinas-social-credit-laboratory/
Fig. 10.31	Dirk Ingo Franke: Six surveillance cameras overlooking a gas station next to the Autobahn A9 3.Juni 2013. https://de.wikipedia.org/wiki/%C3%9Cberwachung#/media/Datei:%C3%9Cberwachungskameras_Autohof_in_Th%C3%BCringen_an_A9_03.06.2013_10-07-08.JPG CC BY 3.0 cropped
Fig. 10.32	https://en.wikipedia.org/wiki/Perdix_(drone)#/media/File:Perdix-drone.jpg public domain
Fig. 10.33	Adobe Stock Datei 102117631. https://stock.adobe.com/de/images/technological-singularity/102117631
Fig. 10.34	https://stock.adobe.com/de/images/artificial-intelligence-and-data-concept/258774440
Fig. 10.35	Self-created graphic. Source:http://biomine.cs.helsinki.fi/
Fig. 10.37	John F. Kennedy (1963), photo: Cecil W. Stoughton public domain. https://de.wikipedia.org/wiki/John_F._Kennedy#/media/Datei:John_F._Kennedy,_White_House_color_photo_portrait.jpg
Fig. 10.38	https://stock.adobe.com/de/images/male-golfplayer-on-professional-golf-course-golfer-with-golf-club-taking-ashot/242337141 added arrows.
Fig. 10.39	Cathy O'Neil bei der Vorstellung ihres Buches bei Google Cambridge 2016 https://de.wikipedia.org/wiki/Cathy_O%E2%80%99Neil#/media/Datei:Cathy_O'Neil_at_Google_Cambridge.jpg by GRuban – Eigenes Werk, CC BY-SA 4.0
Fig. 10.41	Generated with the words of Cremers et al. (2019) using https://www.wordclouds.com/
Fig. 10.42	Self-created graphic. Source Cremers et al. (2019)
Fig. 10.43	Humanoider Roboter als Justitia mit einem Schwert und einen Waageby Alexander Limbach. 229515942 https://stock.adobe.com/de/images/humanoider-roboter-als-justitia-mit-einem-schwert-und-einen-waage/229515942
Fig. 10.44	Stock-Illustration-ID:1145748016 https://www.istockphoto.com/de/vektor/maschinentechnik-flachbanner-gm1145748016-308484501
Fig. 10.45	Richard Patterson: Hacker that looks to have been caught in the act. CC BY 2.0. https://www.flickr.com/photos/136770128@N07/40492737110/
Fig. 10.46	https://www.istockphoto.com/de/foto/computer-netzwerk-sicherheit-cybersicherheit-gro%C3%9Fe-daten-schutz-verschl%C3%BCsselunggm1130248375-298854279

References

Anguelov, D. (2019). *Taming the long tail of autonomous driving challenges. Vortrag Im Rahmen Des MIT Kurses "Deep Learning for Self-Driving Cars",* 15 January 2019, Cambridge. https://www.youtube.com/watch?v=Q0nGo2-y0xY

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. ArXiv Prepr. ArXiv14090473 ICLR. arXiv: 1409.0473.

Britz, D., Goldie, A., Luong, M.-T., & Le, Q. (2017). Massive exploration of neural machine translation architectures. In *Conference on Empirical Methods in Natural Language Processing* (pp. 1442–1451).

Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *ICLR*.

Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP* (pp. 4960–4964). IEEE.

Chen, T., Liu, J., Xiang, Y., Niu, W., Tong, E., & Han, Z. (2019). Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity, 2*(1), 11.

Chung, J. S., Senior, A., Vinyals, O., & Zisserman, A. (2017). Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition CVPR* (pp. 3444–3453). IEEE.

Cremers, A. B., et al. (2019). *Vertrauenswürdiger Einsatz von Künstlicher Intelligenz*. Fraunhofer Institut für Intelligente Analyse-und Informationssysteme, Sankt Augustin.

Dathathri, S., Madotto, A., Lan, J., Hung, J., Frank, E., Molino, P., Yosinski, J., & Liu, R. (2020, March 3). Plug and play language models: A simple approach to controlled text generation. ArXiv Prepr. ArXiv191202164. arXiv: 1912.02164.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. ArXiv Prepr. ArXiv181004805. arXiv: 1810.04805.

Dyrmann, M., Jørgensen, R. N., & Midtiby, H. S. (2017). RoboWeedSupport-Detection of weed locations in leaf occluded cereal crops using a fully convolutional neural network. *Advances in Animal Biosciences, 8*(2), 842–847.

Elgammal, A., Liu, B., Elhoseiny, M., & Mazzone, M. (2017). Can: Creative adversarial networks, generating "art" by learning about styles and deviating from style norms. ArXiv Prepr. ArXiv170607068. arXiv: 1706.07068.

Eslami, S. A., Rezende, D. J., Besse, F., Viola, F., Morcos, A. S., Garnelo, M., Ruderman, A., Rusu, A. A., Danihelka, I., Gregor, K., et al. (2018). Neural scene representation and rendering. *Science, 360*(6394), 1204–1210.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature, 542*(7639), 115–118.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2018). Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1625–1634).

Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. ArXiv Prepr. ArXiv180504833. arXiv: 1805.04833.

Fridman, L. (2019). *Self driving cars: State of the art 2019.* https://www.youtube.com/watch?v=sRxaMDDMWQQ

Giacaglia, G. (2019). Self-driving cars. medium.com.

Grover, P. (2018). *Evolution of object detection and localization algorithms*. https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad

Guo, X., Zhu, B., Polanía, L. F., Boncelet, C., & Barner, K. E. (2018). Group-level emotion recognition using hybrid deep models based on faces, scenes, skeletons and visual attentions. In *Proceedings of 20th ACM International Conference on Multimodal Interaction* (pp. 635–639).

He, K. (2016). Deep residual networks-deep learning gets way deeper - Tutorial. In ICML.

Hendrycks, D., & Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*.

Hosseini, H., & Schmidt, H. (2018). *Wert Der Plattform-Ökonomie Steigt Im Ersten Halbjahr Um 1 Billion $.* https://www.platformeconomy.com/blog/wert-der-plattform-okonomie-steigt-im-ersten-halbjahr-um-1-billion

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., & Eck, D. (2019). Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*. ICLR.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4700–4708).

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1125–1134).

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., & Wu, Y. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. ArXiv Prepr. ArXiv190406037. arXiv: 1904.06037.

Kanellos, M. (2016). 152,000 smart devices every minute in 2025: IDC outlines the future of smart things. Forbes.com.

Karpukhin, V., Levy, O., Eisenstein, J., & Ghazvininejad, M. (2019). Training on synthetic noise improves robustness to natural noise in machine translation. ArXiv Prepr. ArXiv190201509. arXiv: 1902.01509.

Karras, T. (2019). *This person does not exist.* https://thispersondoesnotexist.com/

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097–1105).

Kropp, P., Theuer, S., & Fritzsche, B. (2018). Immer Mehr Tätigkeiten Werden Durch Digitalisierung Ersetzbar. IAB 2018, S SN 1861-14 35, S.32.

Kudugunta, S. R., Bapna, A., Caswell, I., Arivazhagan, N., & Firat, O. (2019). Investigating multilingual Nmt representations at scale. ArXiv Prepr. ArXiv190902197. arXiv: 1909.02197.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K.-R. (2019). Unmasking clever Hans predictors and assessing what machines really learn. *Nature Communications, 10*(1), 1–8.

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM, 54*(10), 95–103.

Levine, S. (2018). *Policy gradients. CS 294-112: Deep reinforcement learning.* http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf

Li, F.-F., Johnson, J., & Young, S. (2018). *CNNs for visual recognition. Lecture 2: Image classification pipeline*. Stanford UNIV. http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture02.pdf

Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Nguyen, H., & Gadde, R. T. (2019). Jasper: An end-to-end convolutional neural acoustic model. ArXiv Prepr. ArXiv190403288. arXiv: 1904.03288.

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A., & Lederberg, J. (1993). DENDRAL: A case study of the first expert system for scientific hypothesis formation. *Artificial Intelligence, 61*(2), 209–261.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.

Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S. A., Yan, T., Brown, L., Fan, Q., Gutfreund, D., Vondrick, C., et al. (2019). Moments in time dataset: One million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 42*(2), 502–508.

MultiNLI. (2017). *The multi-genre NLI corpus*. https://cims.nyu.edu/~sbowman/multinli/ visited on 31.10.2020.

Nayak, P. (2019). Understanding searches better than ever before. Google Blog October 25.

Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., & Zelnik, L. (2020). Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics* PMLR (pp. 493–503).

Osband, I., Blundell, C., Pritzel, A., & Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems* (Vol. 29).

Osborn, L. E., Dragomir, A., Betthauser, J. L., Hunt, C. L., Nguyen, H. H., Kaliki, R. R., & Thakor, N. V. (2018). Prosthesis with neuromorphic multilayered E-dermis perceives touch and pain. *Science Robotics, 3*(19).

Payne, C. (2019). MuseNet. OpenAI Blog.

Phillips, P. J., Yates, A. N., Hu, Y., Hahn, C. A., Noyes, E., Jackson, K., Cavazos, J. G., Jeckeln, G., Ranjan, R., Sankaranarayanan, S., et al. (2018). Face recognition accuracy of forensic examiners, superrecognizers, and face recognition algorithms. *Proceedings of the National Academy of Sciences, 115*(24), 6171–6176.

Pinto, L., Davidson, J., Sukthankar, R., & Gupta, A. (2017). Robust adversarial reinforcement learning. ArXiv Prepr. ArXiv170302702. arXiv: 1703.02702.

Popel, M., & Bojar, O. (2018). Training tips for the transformer model. *Prague Bulletin of Mathematical Linguistics, 110*(1), 43–70.

Radford, A., Metz, L., & Chintala, S. (2016, January 7). Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR 2016*. https://doi.org/10.48550/arXiv.1511.06434. arXiv: 1511.06434 [cs] (visited on 29 May 2022).

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7263–7271).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. ArXiv Prepr. ArXiv160605386. arXiv: 1606.05386.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention* (pp. 234–241). Springer.

Schönherr, L., Kohls, K., Zeiler, S., Holz, T., & Kolossa, D. (2019). Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. In *Network and Distributed System Security Symposium NDSS 2019*.

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. ArXiv Prepr. ArXiv150602438. arXiv: 1506.02438.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics Volume 1: Long Papers* (pp. 1715–1725).

Sharma, P., Ding, N., Goodman, S., & Soricut, R. (2018). Conceptual captions: A cleaned, Hypernymed, image Alt-Text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics Volume 1: Long Papers* (pp. 2556–2565).

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., & Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. ArXiv Prepr. ArXiv170106538. arXiv: 1701.06538.

Shoham, Y., Perrault, R., Brynjolfsson, E., Clark, J., Manyika, J., Niebles, J. C., Lyons, T., Etchemendy, J., Grosz, B., & Bauer, Z. (2018). The AI index 2018 annual report. In *AI Index Steering Committee Human-Centered AI Initiative*

Shum, H.-Y., He, X.-d., & Li, D. (2018). From Eliza to XiaoIce: Challenges and opportunities with social Chatbots. *Frontiers of Information Technology & Electronic Engineering, 19*(1), 10–26.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems* (pp. 568–576).

Strobelt, H., Gehrmann, S., Huber, B., Pfister, H., Rush, A. M., et al. (2016). Visual analysis of hidden state dynamics in recurrent neural networks. ArXiv Prepr. ArXiv160607461. arXiv: 1606.07461.

Sun, C., Myers, A., Vondrick, C., Murphy, K., & Schmid, C. (2019). Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 7464–7473).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (pp. 3104–3112).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. ArXiv Prepr. ArXiv13126199. arXiv: 1312.6199.

Teye, M., Azizpour, H., & Smith, K. (2018). Bayesian uncertainty estimation for batch normalized deep networks. ArXiv Prepr. ArXiv180206455. arXiv: 1802.06455.

Touvron, H., Vedaldi, A., Douze, M., & Jégou, H. (2019). Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems* (pp. 8252–8262).

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., & Belongie, S. (2018). The Inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8769–8778).

Vardaan, director (2019, April 26). *AI to steal fashion model jobs? New AI able to generate entire bodies of people who don't exist.* https://www.youtube.com/watch?v=8siezzLXbNo (visited on 09 February 2019).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008). https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vig, J. (2019). OpenAI GPT-2: Understanding language generation through visualization. *Data Science Medium March* 5.

Wang, C., Li, M., & Smola, A. J. (2019). Language models with transformers. ArXiv Prepr. ArXiv190409408. arXiv: 1904.09408.

Wolf, S. (2018). CycleGAN: Learning to translate images (without paired training data). *Datascience 19112018.*

Wu, L., Fisch, A., Chopra, S., Adams, K., Bordes, A., & Weston, J. (2018). Starspace: Embed all the things! In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 32). 1.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European Conference on Computer Vision* (pp. 818–833). Springer.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. N. (2017). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 5907–5915).

Zhou, L., Zhou, Y., Corso, J. J., Socher, R., & Xiong, C. (2018). End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8739–8748).

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2223–2232).

Zhu, Y., Sapra, K., Reda, F. A., Shih, K. J., Newsam, S., Tao, A., & Catanzaro, B. (2019). Improving semantic segmentation via video propagation and label relaxation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8856–8865).

Index

A
Accuracy of classification, 72, 430
Action in reinforcement learning, 281, 283
Activation function, 78, 430
ActivityNet data, 271
Actor-Critic, 301
Adaptive error injection, 303
Adversarial attack, 150, 232
Affine transformation, 57, 430
Agent in reinforcement learning, 281
Agriculture, 378
AI as a Service, 387
Alexa, 239
Alexa chatbot, 24, 239, 261, 364
AlexNet, 16, 126
Algorithm, 48, 430
Alibaba, 390
ALPHA drone control, 403
AlphaGo, 27
AlphaZero, 27
Amazon, 388
Analogy, 168
Annealing, 107
Apple, 388
Architecture search, 106
Aristo, 24, 210
Art and AI, 381
Artificial General Intelligence (AGI), 38, 405
Artificial Intelligence (AI), 1, 430
  strong, 405
  symbolic, 11, 211
  weak, 38, 404
Artificial Neural Network (ANN), 75, 430
ASAP hyperparameter optimization, 108
Assistant
  conversation, 259
  intelligent personal, 259
  question-answering, 259
  task, 259
  voice, 259
Associative learning, 409
Atari 2600, 29, 293
Attention, 193, 409
  cross, 193
  multi-head, 200
  self, 196
Attention economy, 355
Attention weights, 192
Autoencoder, 430
Autoencoder network, 101
AutoKeras, 107
Awareness, 38

B
Backpropagation, 80, 85, 431
Back-translation, 220
Backward RNN, 431
Bagging, 145
Baidu, 390
Baidu, Alibaba, and Tencent (BAT), 390
Batch normalization regularization, 97
Bayesian neural network, 146
Beamforming, acoustic, 274
Beam Search, 189, 431
Bellman equation, 289
Bias, 57
Bias in the data, 419

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

G. Paaß, D. Hecker, *Artificial Intelligence*,
https://doi.org/10.1007/978-3-031-50605-5

Bidirectional Encoder Representations from Transformer (BERT), 196, 431
Bidirectional LSTM, 184, 431
Bionics and AI, 380
Black box attack, 152
Black-box model, 432
Blender chatbot, 353
BLEU value for translation quality, 190, 432
BookCorpus data, 204
Boosting, 434
Bootstrap, 145, 434
BOS symbol for beginning of sentence, 188
Bot, 29
Bounding box, 113, 135
Breakout game, 293
BST data, 354
Business Expert, 396
Byte-Pair Encoding (BPE), 196

C
Cambridge Analytica, 402
Capture the Flag, 30
Carcraft simulator, 310
CelebA-HQ data, 323
Chain rule, 85
Charade data, 267
Chatbot, 259, 432
  Alexa, 261
  ELIZA, 259
  Gunrock, 261
  Meena, 355
  Tay, 261
  XiaoIce, 348
Chess, 27
CIFAR-10 data, 108, 129
Cityscapes data, 140
Class, 44, 54
Classification, 44, 432
  accuracy, 72
Classifier, 54
CNTK toolkit, 89
Communication and AI, 378
Compute cluster, 88
Computer generated painting, 35, 320, 331
Computer generated story, 37, 335, 340
Computer vision, 432
ConceptNet, 262
Conceptual Captions data, 225
Conditional GAN, 324
CoNLL 2003 data, 184, 205
Connectionism, 11, 82
Consciousness, 409
Context-sensitive embedding, 196, 200, 443
Contextual embedding, 196, 200, 432, 443
Context vector, 193
Conversation assistant, 259
Conversation-turns Per Session (CPS), 349
Convolution
  (2+1)D, 269
  3D, 267
Convolutional layer, 116, 432
Convolutional layer, one-dimensional, 250
Convolutional Neural Network (CNN), 100, 116, 432
  dilated, 255
Coreference resolution, 262
Cosine distance, 166
Counterfactual reasoning learning, 409
Creative Adversarial Network (CAN), 330
Cross-attention, 193, 213, 432
  masked, 214
Cross-Industry Standard Process for Data Mining (CRISP), 104
Crowdworker, 16
Curved surface, 78
Cyberattack, 404
Cyborg text generation, 335
Cyc knowledge base, 208, 407
CycleGAN, 329

D
DARPA Grand Challenge race, 32
Data, 432
Data analyst, 396
Data augmentation, 129
Data Manager, 396
Data Scientist, 395
Data strategy, 383
DCGAN deep convolutional GAN, 322
(2+1)D convolution, 269
Debater dialog system, 25
Decision tree, 228
Decoder block of transformer, 214
Decoder of transformer, 214
Deep Blue chess program, 27
Deep convolutional GAN, 322
Deep Learning, 7, 433
Deep Learning Toolkit, 433
Deep Neural Network (DNN), 6, 82, 433
Deep policy network, 299
Deep Q-Network, 433
Derivative of a function, 65
Deterministic environment, 284
Dialog act, 262
Differential privacy, 423
Digital twin, 375

Digitization, 433
Dilated Convolutional Neural Network, 255
Dimensions of intelligence, 1
Directed Acyclic Graph (DAG), 433
Directed Graph, 433
Discounted future reward, 287
Discount factor, 287, 441
Discrete Laplace operator, 119
Discriminator, 320
Discriminator network, 320
DNA, 18
Dota2 game, 31, 304
Drone, 403
Dropout regularization, 96, 433
Dual learning, 220
Dual use AI system, 424

E
Early stopping, 92
Economy of Scale, 386
Economy of Speed, 386
ELIZA chatbot, 259, 349
Embedding, 162, 434
  context-sensitive, 196, 200
  contextual, 200, 432
  input, 197
  position, 197
Embedding Network, 434
Embedding vector, 162
Encoder block of transformer, 201
Encoder of transformer, 200
Ensemble Method, 145, 434
Ensemble of models, 190
Environment in reinforcement learning, 283
EOS symbol for end of sentence, 188
Episode in reinforcement learning, 284, 434
Epoch, 70, 88, 434
Error, Top5, 125
Exoskeleton, 372
Expert system, 434
Exploding gradient, 176, 434
Exploration, 434
Exploration in reinforcement learning, 284, 292
Explotation in reinforcement learning, 284

F
Facebook, 388
Facebook prediction of human features, 402
Face recognition, 141
Fast Thinking, 408
Few-shot learning, 343
Financial industry and AI, 379
FLOPS floating point operations, 128
Flow Machines music generation, 343
Forget gate, 179
Forward RNN, 431
Fourier transform, 241
Fully connected layer, 79, 434
F-value of a class, 72, 435

G
Gate, 178
Gene, 18
Generalization, 54, 72, 435
Generalized additive model, 228
Generative adversarial network (GAN), 101, 435
  conditional, 324
  CycleGAN, 329
  progressive, 324
  StackGAN, 332
Generative Pretrained Transformer (GPT-2), 338
Generator, 320
Generator network, 320
G-Force, 435
G5G, 430
Global explanatory model, 228
Global parameter vector, 57
Go game, 27, 302
Google, 388
Google, Amazon, Facebook and Apple (GAFA), 388
Google Assistant, 258
Google Duplex, 26
Google Home, 24, 239
GoogLeNet, 126
GPT language model, 435
GPT-3, 343
Gradient, 65, 66
  negative, 66
Gradient clipping, 178
Gradient descent
  minibatch, 70
  stochastic, 70
Gradient descent optimization, 435
Graphics processing unit (GPU), 88, 436
Gunrock, 261
Gym environment simulation, 293

H
HAL 9000 chatbot, 363, 436
Hallucination, 355
Hidden vector, 79, 436
High-Level Expert Group for Artificial Intelligence, 413
Hperparameter optimization, 129
Hybrid Artificial Intelligence, 408
Hyperbolic tangent, 79
Hyperparameter, 69, 91, 436
Hyperparameter optimization, 106, 107, 270, 436

I
Image captioning, 224
ImageNet data, 16, 123
Image recognition, 4, 16, 113
Image segmentation, 136
Image transformation, 326
Image understanding, 113
iNaturals dataset, 134
Inception module, 126
Industry 4.0, 374
Information bottleneck, 101
Information extraction, 183, 436
Input embedding, 197
Instance of Data, 437
Intelligence
  dimensions of, 1
Intelligent assistants
  personal, 24
Intelligent personal assistant, 259
Internet and AI, 380
Internet of Things (IoT), 368, 384, 437
Internet-of-Things platform, 376

J
Jasper, 250

K
Keras toolkit, 90, 437
Kernel
  stride, 119
Kernel of a convolution, 116, 437
Key vector, 198
Knowledge base, 208, 407

L
Labeled Faces in the Wild data, 142
Labor market and AI, 392
LAMBADA benchmark, 339
Language model, 173, 437
  masked, 203
Large-Scale Scene Understanding data (LSUN), 322
Layer of a neural network, 6, 82, 437
Learning, 4
  associative, 409
  counterfactual reasoning, 409
  deep, 7
  dual, 220
  machine, 7, 43
  reinforcement, 45
  self-supervised, 45
  semi-supervised, 134
  simulation, 409
  supervised, 44
  unsupervised, 44
Learning of representations, 234
Learning rate, 69, 437
Least-squares loss function, 62
LibriSpeech data, 243, 249, 252, 254
Lidar sensor, 306
Likelihood, 61, 437
LIME, 229
Linear model, 228
Linear transformation, 57, 437
Lipreading, 254
Listen-Attend-Spell (LAS), 246
Local explanation model, 229
Localization of objects, 113, 135
Loebner Prize, 265
Logistic regression, 59, 438
Log-likelihood loss function, 62
Long Short-Term Memory (LSTM), 178, 438
Loss function, 52, 438
  least-squares, 62
  log-likelihood, 62
  maximum entropy, 62
Lot size 1, 438
L1 regularization, 95
L2 regularization, 95
LSTM cell, 180

M
Machine Learning, 7, 43, 438
Machine translation, 20, 186, 212, 438
Markov-Chain Monte-Carlo (MCMC), 147
Masked cross-attention, 214
Masked language model, 203
Matrix, 49, 439
Maximum entropy loss function, 62
Maximum Likelihood loss, 439

Maximum Likelihood principle, 61, 439
Max-pooling, 121
Max-unpooling, 138
Mean reciprocal rank (MRR), 212
Mean squared distance, 99
Medical image data, 17
Meena chatbot, 355
Megatron, 342
Mel Frequency Cepstral Coefficient (MFCC), 241
Mental simulation, 410
Microtargeting, 402
Minibatch, 69, 439
Minibatch gradient descent, 70
Mixture of Experts, 222
MNIST data, 48
Model, 8, 47, 59, 439
Model application, 71, 439
Model ensemble, 145
Model prediction, 71
Model predictive control, 309
Model uncertainty, 145
Moments-in-Time data, 266
Montezuma's Revenge game, 297
MS-COCO dataset, 225
MS-MARCO dataset, 212
Multi-head attention, 200
Multi-head cross-attention, 439
Multi-head self-attention, 439
Multilayer feedforward network, 82, 99, 439
Multilayer Neural Network, 439
Multilayer RNN, 181
MuseNet music generation, 344
Music Transformer, 346

N
Named entity, 439
Named entity recognition (NER), 183, 205
Natural language processing (NLP), 157, 440
Negative gradient, 66
Neocognitron, 116
Nest and Google, 382
Neural Network, 440
  Bayesian, 146
  deep, 82
Neuron, 108
N-gram, 170, 440
Noise Contrastive Estimation, 165
Nonlinear function, 78, 440
notMNIST data, 93
Number packet, 6

O
Object classification in images, 4, 16, 113, 116
Object localization, 135
Object localization in images, 113
Object segmentation, 114, 136
One-dimensional convolutional layer, 250
Operator, 8, 440
Opinion mining, 206
Optical flow, 269
Optimization, 51
Optimization step, 53
Output gate, 179
Overfitting, 92, 440

P
Padding, 125
Painting, computer generated, 35, 320, 331
Parameter, 6, 52, 440
Parameter sharing, 119, 122
Parameter vector, 52
  global, 57
Parkinson's disease, 370
Partial derivative, 66
Penalization of large parameter values, 95
Penn Treebank data, 182, 339
Pepper robot, 353
Perception algorithm, 308
Perceptron, 10
Perplexity, 182, 440
Personal assistant
  XiaoIce, 348
Personal intelligent assistants, 24
Phrase, 187
Pixel, 3
Platform economy, 386
Plug and play language model (PPLM), 342
Pluribus poker system, 28
Poker, 28
Policy, 100
Policy for reinforcement learning, 45, 286, 441
Policy gradient, 299
Policy gradient model, 440
Policy, stochastic, 298
Pong game, 293
Pooling layer, 121, 432, 441
Portrait photo, 36
Position embedding, 197
Posteriori probability, 146
Precision of a class, 72, 441
Prediction, 51
Predictive Maintenance, 375

Print media, 377
Privacy Preserving Machine Learning, 399
Probability vector, 58, 441
Process model, 375
Progressive GAN, 324
Prosody speech melody, 255
Protein, 18
Proximal policy optimization (PPO), 301
PyTorch toolkit, 89

Q
Q-Function in reinforcement learning, 287, 441
Q-Network, 289
Quake III Arena Capture the Flag, 30
Query vector, 198
Question answering, 22, 206
Question-answering assistant, 259

R
Radar sensor, 305
Ranking algorithm for web search, 211
Recall of a class, 72, 441
Receptive field, 117
Rectified linear unit (ReLU), 79
Recurrent Neural Network (RNN), 100, 171, 441
Regression, 44, 442
Regression model, 99
    logistic, 59
Regression problem, 99
Regularization, 94, 442
    batch normalization, 97
    Dropout, 96
    L1, 95
    L2, 95
Reinforcement learning, 27, 45, 100, 281, 442
Relation, 160
Replika chatbot, 353
Representation, learning of, 234
Residual block, 127
Residual connection, 127, 201, 216, 442
ResNet, 127
ResNeXt, 129
Resolution of images, increase, 327
Reward for reinforcement learning, 45, 442
Reward in reinforcement learning, 281, 283
Reward, discounted future, 287
RGB input, 125
RNN, multilayer, 181
Roadrunner game, 293
Robot control, 302
Route planning, 308
Rule set, 228

S
Scalar, 50, 442
Scalar product, 56
Scissors-rock-paper game, 297
Score, 54
Seaquest game, 293
Self-attention, 196, 443
Self-driving car, 32, 305
Self-supervised learning, 45, 443
Semi-supervised learning, 134
Sensor for self-driving car, 305
Sentiment analysis, 206, 352
Separating hyperplane, 77
Sequence, 100, 171
Sequence model, 443
Sequence-to-sequence model, 188, 443
Serious games, 370
Sigmoid function, 79
Simulation learning, 409
Simultaneous translation, 21, 220
Singularity, technological, 405
Siri chatbot, 24, 364
Slow Thinking, 408
Smart home, 365
Smart speaker, 260
Social Credit, 400
Softmax function, 58, 443
Software agent, 29
Software architect, 397
Sparse Transformer, 344
Speech generation, 255
Speech melody, 255
Speech recognition, 19, 239
Speech signal, 241
Speech synthesis, 25, 255
SQuAD question answering data, 23, 206
StackGAN, 332
StarCraft 2 video game, 32
State in reinforcement learning, 281, 283
Static embedding, 162
Statistical interaction, 76
Stochastic environment, 284
Stochastic gradient descent, 70, 443
Stochastic policy, 298, 444
Store gate, 179
Story, computer generated, 37, 335, 340
Stride of a kernel, 119
Strong Artificial Intelligence, 405
Student model, 134

Supervised learning, 44, 444
Surveillance and AI, 400
Surveillance in China, 400
SWAG data for text understanding, 209
Switchboard speech data, 19, 242, 252–254
Symbol, 159
Symbol grounding, 160, 411
Symbolic Artificial Intelligence, 11, 22, 211, 407
Synapse, 108
Synonym, 161

T
Tacotron, 258
Task assistant, 259
Tay chatbot, 261
Teacher model, 134
Technological singularity, 405
Tencent, 390
Tensor, 50, 444
TensorFlow toolkit, 89
Tesla, 309
Test example, 72
Testing AI systems, 421
Test set, 72, 444
Text-to-speech, 255
3D convolution, 267
3D generation, 334
3D structure of proteins, 18
Time series of numerical data, 185
Token, 444
Token embedding, 431
Top5 error, 16, 125
Training, 51, 444
Training data, 4
Training example, 4, 72
Training set, 4, 72, 444
Transfer learning, 133, 196, 204, 411, 444
Transformation
    affine, 57
    linear, 57
Transformer, 212, 445
    decoder, 213, 214
    decoder block, 445
    encoder, 200, 212
    encoder block, 445
Translation
    automatic, 186
    machine, 20
    simultaneous, 21, 220
Translation invariant, 119
Transportation and AI, 379
Transposed convolution, 140
t-SNE, 166
TTS text-to-speech, 25, 255
Turing Natural Language Generation (T-NLG), 342
Turing test, 2

U
Uncertain reasoning, 408
Uncertainty
    lack of data, 144
    measurement errors, 145
    model, 145
Underfitting, 92
U-Net, 139
Unicorn startup company, 374
Unsupervised learning, 44, 445

V
Validation set, 446
Value vector, 198
Vanishing gradient, 178, 446
Vector, 49, 446
Video
    camera, 305
    captioning, 266
    classification, 266
    content analysis, 266
    generation, 334
Vocabulary, 163
Voice assistant, 259

W
Watson, 22
Waymo, 309
Weak Artificial Intelligence, 38, 404
Wearables, 384
Web search, AI supported, 211
WebText data, 339
Weight decay, 95
White box attack, 152
WMT
    2014 En-De data, 218
    2014 En-Fr data, 194, 218
    2017 translation data, 195
Woebot chatbot, 353
Word Embedding, 434
Word error rate (WER), 19, 242
WordPiece tokenization, 197
Wordsmith text generation, 335
Working memory, 408
World knowledge, 208

X
XiaoIce chatbot, 348
XiaoIce personal assistant, 348
XOR problem, 75

Y
YOLO, 135
Yolo9000, 136