



Gerhard Paaß · Dirk Hecker

# Künstliche Intelligenz

Was steckt hinter der  
Technologie der Zukunft?

EBOOK INSIDE



Springer Vieweg

---

# Künstliche Intelligenz

---

Gerhard Paaß • Dirk Hecker

# Künstliche Intelligenz

Was steckt hinter der Technologie der Zukunft?



Springer Vieweg

Gerhard Paaß  
Fraunhofer-Institut für Intelligente Analyse-  
und Informationssysteme IAIS  
Sankt Augustin, Deutschland

Dirk Hecker  
Fraunhofer-Institut für Intelligente Analyse-  
und Informationssysteme IAIS  
Sankt Augustin, Deutschland

ISBN 978-3-658-30210-8

ISBN 978-3-658-30211-5 (eBook)

<https://doi.org/10.1007/978-3-658-30211-5>

Die Deutsche Nationalbibliothek verzeichnetet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Springer Fachmedien Wiesbaden GmbH, ein Teil von Springer Nature 2020

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von allgemein beschreibenden Bezeichnungen, Marken, Unternehmensnamen etc. in diesem Werk bedeutet nicht, dass diese frei durch jedermann benutzt werden dürfen. Die Berechtigung zur Benutzung unterliegt, auch ohne gesonderten Hinweis hierzu, den Regeln des Markenrechts. Die Rechte des jeweiligen Zeicheninhabers sind zu beachten.

Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Planung: Sybille Thelen Titelbild: sdecoret – stock.adobe.com

Springer Vieweg ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.

Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

---

## Grußwort von Minister Prof. Dr. Andreas Pinkwart

Liebe Leserinnen, liebe Leser,

getrieben von Neugier und Enthusiasmus für neue Technologien verfolgen Forscher und Entwickler seit vielen Jahrzehnten das Ziel, Menschen und Maschinen miteinander zu verbinden. Jenseits der Faszination, die die künstliche auf die menschliche Intelligenz ausübt, steht heute mit zunehmender Anwendungsbreite nicht mehr Utopie, sondern vielmehr die konkrete Rolle im Mittelpunkt, die Künstliche Intelligenz (KI) auf dem Weg zu einer digitalen Wirtschaft und Gesellschaft einnimmt. KI wird zu einem Treiber des Wandels.

Die KI-Forschung hat inzwischen eine Leistungsstärke erreicht, die selbst bei komplexen Aufgabenstellungen beeindruckende Ergebnisse erzielt. Und auch in unserem Alltag ist KI längst angekommen: Ob wir unser Smartphone per Spracheingabe nach dem Wetter fragen, Texte mit einem Mausklick in andere Sprachen übersetzen lassen oder automatisch über den Betrugsvorwurf mit unserer Kreditkarte informiert werden – all dies sind bereits etablierte Anwendungen Künstlicher Intelligenz. Aber die Potenziale und technologischen Möglichkeiten sind um ein Vielfaches höher.

Für Unternehmen bietet KI enorme Chancen, die es mit Mut und Kreativität beherzt zu ergreifen gilt. Wo Freude an Innovationen und die unternehmerische Bereitschaft, sich neuen Herausforderungen zu stellen, zusammentreffen, dort können diese Chancen für bessere Produkte, effiziente und sichere Prozesse sowie neue Geschäftsmodelle genutzt werden. Was hier möglich ist, zeigen uns unter anderem die zahlreichen agilen Start-ups, die mit kreativen Ideen die Technologie vorantreiben. Das autonome Fahren wird die Mobilität verändern und Sicherheit und Komfort auf ein neues Level heben. Auch im Gesundheitssektor und der E-Health-Branche verspricht KI konkreten Nutzen durch Verbesserungen in der Diagnostik, der Früherkennung von Krankheiten oder der Entwicklung neuer Medikamente.

Zu den enormen Chancen gehören aber auch elementare Fragen, die durch die Entwicklungen in der Künstlichen Intelligenz aufgeworfen werden: Was bedeutet es, wenn die Grenzen zwischen Mensch und Maschine verschwimmen? Welche Vorkehrungen sind für eine konsequente Bindung von KI-Anwendungen an ethische Standards zu treffen? Wer trägt die Verantwortung für vom Menschen an Maschinen delegierte Entscheidungen?

In Nordrhein-Westfalen wollen wir der Zukunftstechnologie KI den Weg ebnen und haben uns dabei ambitionierte Ziele gesetzt: Unser Bundesland soll zu den zehn stärksten europäischen Standorten für die Entwicklung und Anwendung Künstlicher Intelligenz gehören. Mit der Kompetenzplattform KI.NRW schaffen wir ein Netzwerk für Forschung und Technologietransfer, unterstützen die Unternehmen bei der Umsetzung von Anwendungen und Strategien Künstlicher Intelligenz und nehmen dabei auch ethische Fragen in den Blick, etwa mit der Entwicklung eines „KI-Gütesiegels“ zur Zertifizierung von KI-Lösungen.

Jede der Chancen, die mit Künstlicher Intelligenz verbunden ist, beginnt mit Verständnis und Expertise. Die Technologie zu begreifen, die unser Leben so stark prägen wird, ist von essentieller Bedeutung. Wir brauchen Expertinnen und Experten für die Entwicklung lernender Systeme und wir brauchen mündige Anwenderinnen und Anwender, die ihre Funktionsweisen verstehen, ihren Nutzen und ihre Grenzen abwägen können. Je mehr Wissen wir über die Technologie erwerben, desto besser können wir alle die digitale Zukunft mit Ideenreichtum, Innovationskraft und Verantwortung gestalten.

Das vorliegende Buch zielt genau auf diese Grundvoraussetzung für einen erfolgreichen Umgang mit Künstlicher Intelligenz und verbindet die in diesem Sinne „richtige“ Frage nach dem „Was steckt dahinter?“ mit dem erfreulichen Anspruch, einen verständlichen Einstieg in die Thematik zu schaffen. Bei der Lektüre der Antworten und ihrem Transfer in den praktischen Umgang mit KI wünsche ich viel Freude.

Prof. Dr. Andreas Pinkwart

Minister für Wirtschaft, Innovation, Digitalisierung und Energie des Landes Nordrhein-Westfalen

---

# Vorwort

---

## Über dieses Buch

Künstliche Intelligenz (KI) ist das Schlagwort unserer Zeit. Als zentraler Treiber der Digitalisierung verändert sie Gesellschaft, Wirtschaft und nahezu alle anderen Lebensbereiche in grundlegender Weise. Die Geschwindigkeit dieses Prozesses ist im Vergleich zu früheren gesellschaftlichen und technischen Veränderungen fast beispiellos. Durchbrüche in der Entwicklung sogenannter tiefer künstlicher neuronaler Netze auf hochleistungsfähigen Rechnern haben diese rasante technologische Entwicklung ausgelöst.

Schon Anfang der 50er-Jahre war dem KI-Pionier Alan Turing klar, dass man für viele Fragestellungen Computer nicht händisch bis in jedes Detail programmieren kann. Es müsste eine zügigere Methode geben, Computer zu programmieren: diese Methode ist das maschinelle Lernen. Die Techniken, mit denen Computer aus vorhandenen Daten lernen, ihr Verhalten zu verbessern, sind inzwischen so leistungsfähig, dass sie an vielen Stellen im Alltag und in unserem Berufsleben zum Einsatz kommen.

Der Vielfalt der Anwendungsgebiete sind kaum Grenzen gesetzt. Intelligente Maschinen nehmen die Umwelt wahr, machen Prognosen, geben Empfehlungen und treffen automatisiert Entscheidungen. Sie nehmen uns Routinearbeiten ab und unterstützen uns bei verantwortungsvollen Tätigkeiten. Die Mensch-Maschine-Beziehung ändert sich hin zu einem Partnerschaftsmodell. Intelligente Systeme entlasten uns, erweitern unsere Fähigkeiten und erhöhen unsere Lebensqualität, führen aber auch zu tiefgreifenden Änderungen der Gesellschaft.

So kann KI einen Beitrag zur Bewältigung von großen gesellschaftlichen Herausforderungen wie beispielsweise Mobilität und Gesundheit leisten. Um aber fundiert über Nutzen, Chancen und Risiken von KI diskutieren zu können, muss der Nutzer verstehen, wie intelligente Systeme im Prinzip funktionieren. Dafür muss er die wichtigsten Konzepte der dahinterstehenden Technologie des maschinellen Lernens begreifen. Zudem wird in immer mehr Einsatzgebieten deutlich, dass eine sorgfältige Gestaltung notwendig ist, damit KI im Einklang mit unseren gesellschaftlichen Werten und unserer Vorstellung von Souveränität steht. Hier müssen auch Nicht-Informatiker sachverständig und fundiert mithören können.

Der Zweck des vorliegenden Buches ist es, die neuen Möglichkeiten des maschinellen Lernens in unterschiedlichen Einsatzgebieten, wie beispielsweise dem autonomen Fahren, der medizinischen Diagnose oder der Analyse der Bedeutung von Sprache anschaulich aufzuzeigen. Dabei werden das Fachvokabular aber auch Konzepte, Methoden und Netzarchitekturen mit vielen Grafiken und Bildern erklärt. Mathematische Zusammenhänge werden, wenn hilfreich, formuliert, und immer nachvollziehbar erläutert. Es stellt sich heraus, dass die verwendeten Verfahren aus sehr einfachen Operationen, wie Addition und Multiplikation, zusammengesetzt sind. Diese gewinnen ihre Leistungsfähigkeit, indem sie auf sehr große Zahlenpakete und mehrfach hintereinander angewendet werden. Möchte man die technischen Kap. 3, 4, 5, 6, 7, 8 und 9 detailliert nachvollziehen, so reicht ein mathematisches Verständnis auf Oberstufenniveau.

Das Buch befähigt Entscheider, aber auch interessierte Laien, bei der Gestaltung von intelligenten Systemen mitreden zu können und Anforderungen in ihren Auswirkungen besser einzuschätzen. Datenanalysten, Studierenden, Ingenieuren, Forschern, die sich neu in das Fachgebiet einarbeiten möchten, bietet das Buch den idealen Einstieg in weiterführende Fachliteratur.

---

## Danksagung

Dieses Buch wurde erst ermöglicht durch die motivierende und fachlich anregende Umgebung des Fraunhofer-Instituts für Intelligente Analyse- und Informationssysteme IAIS in Sankt Augustin. Wir danken allen Kolleginnen und Kollegen sowie den Menschen aus unserem persönlichen Umfeld, die uns bei diesem Buchprojekt unterstützt haben – sei es durch fachliche Diskussionen, Lektorate einzelner Kapitel und hilfreiche Anmerkungen: Robert Babatz, Niklas Beck, Katharina Beckh, Sven Giesselbach, Harald Grund, Monika Hommes-Rüdiger, Birgit Kirsch, Franz Lutter, Dominik Paaß, Julia Paaß und Benjamin Schaarwächter. Mirco Lange und Dr. Henning Petzka haben darüber hinaus das gesamte Buch durchgearbeitet und wertvolle Hinweise für Verbesserungen der Darstellung und der Argumentation geliefert, herzlichen Dank dafür.

Ein ganz besonderer Dank gilt Frau Dr. Angi Voss. Sie fand nicht nur aufmunternde Worte in schwierigen Schreibphasen, sondern hat auch mit ihrer langjährigen Erfahrung als Autorin, ihren antreibenden Fragen und inspirierenden Hinweisen ganz wesentlich zu diesem Buch beigetragen.

Viel Aufwand wurde auch in die mehr als 400 Schaubilder, Grafiken und Fotos investiert. Von den ersten Ideen und Grafiken von Julia Paaß bis hin zur endgültigen und professionellen Gestaltung durch Svenja Niehues, die auch viele inhaltliche Vorschläge für das verständliche Design der Grafiken machte. Vielen Dank für Eure Ausdauer und die langen gemeinsamen Diskussionen am Bildschirm.

Der größte Dank gebührt aber unseren Familien, die uns während der langen Zeit des Schreibens die nötigen Freiräume gegeben haben. Insbesondere möchte ich, Gerhard Paaß, meiner Frau Margret Paaß danken, die mit viel Geduld und Zuspruch einen großen

Anteil am Gelingen dieses Buches hatte und von der Planung bis zur Korrektur der Druckfahnen eine unentbehrliche Hilfe war. Und ich, Dirk Hecker, danke meiner Familie Katrin Berkler und Lara Hecker, die selbst in Urlauben die Arbeit an diesem Buch unterstützt und auch in vielen anderen Situationen so viel Verständnis aufgebracht haben. Ohne Euren Zuspruch und Eure Unterstützung hätten wir dieses Buch nicht zustande gebracht.

Vielen Dank für all Eure Unterstützung!

---

# Inhaltsverzeichnis

<b>1 Was ist intelligent an Künstlicher Intelligenz?</b>	1
1.1 Menschliche Intelligenz hat viele Dimensionen . . . . .	1
1.2 Woran erkennt man Künstliche Intelligenz . . . . .	2
1.3 Computer lernen. . . . .	3
1.4 Tiefe neuronale Netze können Objekte erkennen . . . . .	6
1.5 Wie kann man Künstliche Intelligenz verstehen? . . . . .	8
1.6 Die Geschichte der Künstlichen Intelligenz. . . . .	10
1.7 Zusammenfassung . . . . .	12
Literatur. . . . .	13
<b>2 Was kann Künstliche Intelligenz?</b>	15
2.1 Objekterkennung in Bildern. . . . .	15
2.1.1 Medizinische Diagnose . . . . .	17
2.1.2 Vorhersage der 3D-Struktur von Proteinen . . . . .	18
2.2 Spracherkennung . . . . .	19
2.3 Maschinelle Übersetzung. . . . .	20
2.4 Die Beantwortung natürlichsprachiger Fragen . . . . .	22
2.5 Dialoge und persönliche Assistenten . . . . .	25
2.6 Brettspiele . . . . .	27
2.6.1 Das Strategiespiel Go. . . . .	28
2.6.2 Künstliche Intelligenz gewinnt gegen fünf Poker-Professionals . . . . .	29
2.7 Videospiele. . . . .	29
2.7.1 Atari 2600 Spielekonsole. . . . .	30
2.7.2 Capture the Flag. . . . .	30
2.7.3 Das Echtzeit-Strategiespiel Dota2 . . . . .	32
2.8 Selbstfahrende Autos . . . . .	33
2.8.1 Weiterentwicklung der selbstfahrenden Autos. . . . .	34
2.9 Der Computer als kreatives Medium . . . . .	35
2.9.1 Neue Bilder komponieren . . . . .	36
2.9.2 Geschichten erfinden . . . . .	38
2.10 Allgemeine Künstliche Intelligenz . . . . .	39

2.11 Zusammenfassung . . . . .	40
Literatur . . . . .	40
<b>3 Einige Grundbegriffe des maschinellen Lernens . . . . .</b>	<b>45</b>
3.1 Die wichtigsten Arten des maschinellen Lernens . . . . .	45
3.1.1 Überwachtes Lernen . . . . .	45
3.1.2 Unüberwachtes Lernen . . . . .	46
3.1.3 Bestärkungslernen . . . . .	47
3.2 Programmieren und Lernen . . . . .	48
3.2.1 Modelle transformieren eine Eingabe in eine Ausgabe . . . . .	48
3.2.2 Algorithmen arbeiten schrittweise eine Liste von Anweisungen ab. . . . .	50
3.2.3 Ein Lernproblem: Die Erkennung von Ziffern. . . . .	50
3.2.4 Vektoren, Matrizen und Tensoren . . . . .	51
3.3 Lernen eines Zusammenhangs . . . . .	53
3.3.1 Schema für das Lernen: Modell, Verlustfunktion und Optimierung . . . . .	53
3.3.2 Detaillierter Ablauf des Lernens . . . . .	53
3.4 Ein simples Modell: Die logistische Regression . . . . .	56
3.4.1 Berechnung eines Punktwertes . . . . .	56
3.4.2 Die gleichzeitige Berechnung aller Punktweite. . . . .	58
3.4.3 Lineare Transformation . . . . .	59
3.4.4 Die Softmaxfunktion erzeugt einen Wahrscheinlichkeitsvektor . . . . .	60
3.4.5 Das logistische Regressionsmodell . . . . .	61
3.5 Die Güte des Modells . . . . .	62
3.5.1 Maßstab für die Modellgüte: Die Wahrscheinlichkeit der kompletten Trainingsdaten . . . . .	62
3.5.2 Wie misst man den Lernerfolg: Die Verlustfunktion . . . . .	63
3.5.3 Verdeutlichung für zwei Klassen und zwei Eingabemerkmale . . . . .	65
3.6 Optimierung, oder wie findet man die besten Parameterwerte . . . . .	66
3.6.1 Der Gradient zeigt in Richtung des steilsten Anstiegs . . . . .	67
3.6.2 Der Gradient für mehrere Dimensionen . . . . .	67
3.6.3 Der Gradient der Verlustfunktion . . . . .	69
3.6.4 Schrittweise Minimierung durch Gradientenabstieg . . . . .	71
3.6.5 Die Lernrate bestimmt die Länge eines Optimierungsschritts . . . . .	72
3.6.6 Gradientenabstieg mit Minibatch benötigt viel weniger Rechenaufwand . . . . .	72
3.7 Überprüfung des Lernerfolges . . . . .	74
3.7.1 Anwendung des Modells auf neue Daten . . . . .	74
3.7.2 Überprüfung der Genauigkeit auf der Testmenge . . . . .	75
3.7.3 Präzision und Recall für Klassen unterschiedlichen Umfangs . . . . .	76
3.8 Zusammenfassung . . . . .	77
Literatur . . . . .	78

<b>4 Tiefes Lernen kann komplexe Zusammenhänge erfassen . . . . .</b>	79
4.1 Beim XOR-Problem gibt es Interaktionen zwischen den Merkmalen . . . . .	79
4.2 Nichtlinearitäten erzeugen gebogene Trennflächen . . . . .	82
4.3 Tiefe neuronale Netze sind Stapel von nichtlinearen Schichten . . . . .	87
4.3.1 Vektoren und Tensoren repräsentieren die umgeformten Inhalte . . . . .	88
4.4 Training von TNN mit Backpropagation-Verfahren . . . . .	90
4.5 Toolkits erleichtern die Formulierung und das Training von TNN . . . . .	93
4.5.1 Parallele Berechnungen beschleunigen das Training von TNN . . . . .	93
4.5.2 Toolkits erleichtern die Arbeit mit TNN . . . . .	94
4.6 Wie mache ich das Netz besser? . . . . .	96
4.6.1 Iterative Konstruktion eines guten Modells mit der Validationsmenge . . . . .	96
4.6.2 Unteranpassung und Überanpassung führen zu höheren Fehlern . . . . .	97
4.6.3 Ein Beispiel für Überanpassung . . . . .	98
4.6.4 Regularisierungsverfahren reduzieren den Generalisierungsfehler . . . . .	100
4.7 Unterschiedliche Anwendungen erfordern Netze unterschiedlichen Aufbaus . . . . .	103
4.7.1 Mehrschichtiges Feedforward-Netz . . . . .	104
4.7.2 Convolutional Neural Network (CNN) . . . . .	105
4.7.3 Rekurrentes neuronales Netz (RNN) . . . . .	105
4.7.4 Bestärkungslernen-Netz . . . . .	105
4.7.5 Generatives adversariales Netz (GAN) . . . . .	106
4.7.6 Autoencoder-Netze erzeugen eine komprimierte Darstellung . . . . .	106
4.7.7 Architekturen für bestimmte Medien und Anwendungsbereiche . . . . .	106
4.8 Die Konstruktion eines tiefen neuronalen Netzes ist ein Suchprozess . . . . .	108
4.8.1 Auswahl der Anzahl der Parameter und der Hyperparameter . . . . .	108
4.8.2 Der Standard-Prozess der Modellsuche führt zu besseren Modellen . . . . .	109
4.8.3 Automatische Suche von Modellarchitekturen und Hyperparametern . . . . .	111
4.9 Biologische neuronale Netze funktionieren anders . . . . .	114
4.10 Zusammenfassung und Trends . . . . .	116
Literatur . . . . .	117
<b>5 Bilderkennung mit tiefen neuronalen Netzen . . . . .</b>	119
5.1 Was bedeutet eigentlich Bilderkennung? . . . . .	119
5.1.1 Arten der Objekterkennung in Bildern . . . . .	119
5.1.2 Inspirationen aus der Biologie . . . . .	120
5.1.3 Warum ist eine Bilderkennung schwierig? . . . . .	123
5.2 Die Bestandteile eines Convolutional Neural Networks . . . . .	123
5.2.1 Der Kernel einer Convolution-Schicht analysiert kleine Bildbereiche . . . . .	123

5.2.2	Convolution-Schicht mit vielen Kerneln reagiert auf viele Merkmale . . . . .	126
5.2.3	Die Pooling-Schicht wählt die wichtigsten Merkmale aus . . . . .	127
5.3	Ein einfaches Convolutional Neural Network für die Ziffernerkennung . . . . .	128
5.4	Der ImageNet Wettbewerb befeuert die Methodenentwicklung . . . . .	130
5.5	Fortschrittliche Convolutional Neural Networks . . . . .	132
5.5.1	AlexNet nutzt erfolgreich GPUs zum Training . . . . .	132
5.5.2	ResNet erleichtert die Optimierung durch Umgehungspfade . . . . .	133
5.5.3	DenseNet verwendet zusätzliche Umgehungspfade . . . . .	136
5.5.4	ResNeXt nutzt transformierte Bilder zum Training . . . . .	137
5.6	Analyse der CNN Ergebnisse . . . . .	138
5.6.1	Einzelne Kernel reagieren auf Merkmale unterschiedlicher Art und Größe . . . . .	138
5.6.2	Ähnlichen Bildern entsprechen benachbarte verdeckte Vektoren . . . . .	139
5.7	Transferlernen reduziert den Bedarf an Trainingsdaten . . . . .	140
5.8	Lokalisierung eines Objektes im Bild . . . . .	143
5.8.1	Objektlokalisierung durch Rechtecke . . . . .	143
5.8.2	Bildsegmentierung zur pixelgenaue Bestimmung von Klassen . . . . .	145
5.8.3	Max-Unpooling belegt ein vergrößertes Feld mit Werten . . . . .	146
5.8.4	Das U-Net erkennt zunächst Objekte und findet dann die zugehörigen Pixel . . . . .	147
5.9	Die 3D-Rekonstruktion einer Szene . . . . .	149
5.10	Gesichter können mit sehr hoher Genauigkeit zugeordnet werden . . . . .	150
5.11	Abschätzung der Genauigkeit von Modellprognosen . . . . .	152
5.11.1	Unsicherheit der Modellprognosen . . . . .	153
5.11.2	Der Bootstrap erzeugt eine Menge plausibler Modelle . . . . .	155
5.11.3	Bayessche neuronale Netze . . . . .	155
5.12	Zuverlässigkeit der Bilderkennung . . . . .	157
5.12.1	Der Einfluss von Bildverzerrungen . . . . .	157
5.12.2	Gezielte Konstruktion von falschklassifizierten Bildern . . . . .	160
5.13	Zusammenfassung und Trends . . . . .	163
	Literatur . . . . .	164
<b>6</b>	<b>Erfassung der Bedeutung von geschriebenem Text . . . . .</b>	<b>167</b>
6.1	Wie kann man die Bedeutung von Wörtern durch Vektoren darstellen? . . . . .	170
6.1.1	Das Konzept der Embeddingvektoren . . . . .	172
6.1.2	Berechnung von Embeddingvektoren mit Word2vec . . . . .	173
6.1.3	Die Approximation der Softmaxfunktion reduziert den Rechenaufwand . . . . .	175
6.2	Eigenschaften der Embedding-Vektoren . . . . .	176
6.2.1	Nächste Nachbarn von Embeddings haben ähnliche Bedeutungen . . . . .	176

6.2.2	Differenzen zwischen Embeddings lassen sich als Relationen interpretieren . . . . .	178
6.2.3	FastText nutzt n-Gramme von Buchstaben . . . . .	180
6.2.4	StarSpace erzeugen Embeddings für andere Objekte . . . . .	182
6.3	Rekurrente neuronale Netze zur Erfassung von Sequenzen . . . . .	183
6.3.1	Rekurrente neuronale Netze als Sprachmodelle . . . . .	184
6.3.2	Training der rekurrenten neuronalen Netze . . . . .	186
6.3.3	Die Eigenschaften der Gradienten beim RNN . . . . .	187
6.4	Das Long-Short Term Memory (LSTM) ist ein Langzeitspeicher . . . . .	189
6.4.1	Gatter steuern die Speichervorgänge . . . . .	189
6.4.2	LSTMs mit mehreren Schichten . . . . .	192
6.4.3	Anwendungen des LSTM . . . . .	193
6.4.4	Bidirektionale LSTM-Netze zur Prognose von Worteigenschaften . . . . .	195
6.4.5	Visualisierung von rekurrenten neuronalen Netzen . . . . .	197
6.5	Übersetzung: Transformation einer Sequenz in eine andere Sequenz . . . . .	198
6.5.1	Sequence-to-Sequence Netze zur Übersetzung . . . . .	199
6.5.2	Attention: Verbesserung der Übersetzung durch Rückgriff auf die Eingabeworte . . . . .	203
6.5.3	Übersetzungsergebnisse mit Attention . . . . .	205
6.6	Transformer-Übersetzungsmodelle . . . . .	208
6.6.1	Selbstattention analysiert die Abhängigkeiten der Worte eines Satzes . . . . .	209
6.6.2	Kreuzattention analysiert die Abhängigkeiten zwischen Ein- und Ausgabe . . . . .	210
6.6.3	Transformer-Architektur nutzt Selbst- und Kreuzattention . . . . .	212
6.6.4	Training des Transformers für die Sprachübersetzung . . . . .	214
6.6.5	Byte-Pair Kodierung zur Reduktion des Vokabulars und zur Repräsentation unbekannter Wörter . . . . .	216
6.6.6	Ergebnisse für das Transformer-modell . . . . .	216
6.6.7	Simultane Übersetzung erfordert Wartezeiten . . . . .	219
6.7	BERT: Sprachmodelle für die Repräsentation von Bedeutungen . . . . .	221
6.7.1	BERT-Architektur . . . . .	221
6.7.2	BERT-Prognoseaufgaben zum unüberwachten Vortraining . . . . .	222
6.8	Transferlernen mit umfangreichen BERT-Modellen der Sprache . . . . .	224
6.8.1	Semantische Klassifikationsaufgaben . . . . .	224
6.8.2	Die Beantwortung von Fragen . . . . .	226
6.8.3	Extraktion von Weltwissen . . . . .	228
6.8.4	Transferlernen für Übersetzungsmodelle . . . . .	231
6.8.5	Anwendung von BERT in der Web-Suche . . . . .	233
6.9	Die Beschreibung von Bildern durch Text . . . . .	234
6.10	Die Erklärung der Prognosen von TNN . . . . .	237
6.10.1	Erklärungen sind notwendig . . . . .	237

6.10.2 Globale Erklärungsmodelle . . . . .	238
6.10.3 Lokale Erklärungsmodelle . . . . .	239
6.11 Zuverlässigkeit der Texterkennung . . . . .	241
6.11.1 Robustheit bei Textfehlern und Änderung der Domäne . . . . .	241
6.11.2 Anfälligkeit für böswillige Änderung von Eingaben . . . . .	242
6.12 Zusammenfassung und Trends . . . . .	243
Literatur . . . . .	245
<b>7 Gesprochene Sprache verstehen . . . . .</b>	<b>249</b>
7.1 Spracherkennung . . . . .	249
7.1.1 Warum ist Spracherkennung schwierig? . . . . .	249
7.1.2 Wie kann man Sprachsignale im Computer darstellen? . . . . .	250
7.1.3 Messung der Genauigkeit der Spracherkennung . . . . .	253
7.1.4 Die Geschichte der Spracherkennung . . . . .	254
7.2 Tiefe Sequence-to-Sequence-Modelle . . . . .	255
7.2.1 Listen-Attend-Spell erzeugt eine Folge von Buchstaben . . . . .	256
7.2.2 Sequence-To-Sequence Modell für Worte und Silben . . . . .	259
7.3 Convolutional Neural Network zur Spracherkennung . . . . .	259
7.3.1 CNN Modelle . . . . .	260
7.3.2 Kombinierte Modelle . . . . .	263
7.4 Lippenlesen . . . . .	264
7.5 Erzeugung von gesprochener Sprache aus Text . . . . .	265
7.5.1 WaveNet mit gedehnten Convolution-Schichten für lange Abhängigkeiten . . . . .	266
7.5.2 Das Tacotron erzeugt ein Spektrogramm . . . . .	268
7.6 Dialoge und Sprachassistenten . . . . .	270
7.7 Gunrock: Ein erweiterter Alexa-Sprachassistent . . . . .	272
7.7.1 Sprachverstehen . . . . .	273
7.7.2 Dialogmanagement . . . . .	274
7.7.3 Antworterzeugung . . . . .	275
7.7.4 Erprobung des Sprachassistenten . . . . .	275
7.8 Analyse der Inhalte von Videos . . . . .	276
7.8.1 Aufgaben der Videoinhaltsanalyse . . . . .	277
7.8.2 Trainingsdaten zur Klassifikation von Videos nach Aktivitäten . . . . .	277
7.8.3 Convolution-Schichten zur Erkennung von Videoinhalten . . . . .	278
7.8.4 Genauigkeit der Videoklassifikation . . . . .	281
7.8.5 Die Erzeugung von Untertiteln für Videos . . . . .	282
7.9 Zuverlässigkeit der Verarbeitung gesprochener Sprache . . . . .	285
7.9.1 Der Effekt von Rauschen und anderen Verzerrungen auf die Spracherkennung . . . . .	286
7.9.2 Adversariale Attacken auf die automatische Spracherkennung . . . . .	286
7.10 Zusammenfassung . . . . .	287
Literatur . . . . .	289

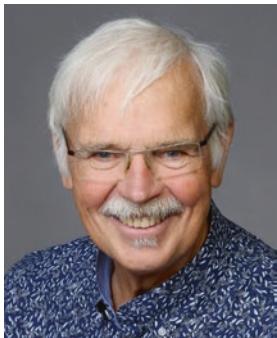
<b>8 Lernen von optimalen Strategien .....</b>	293
8.1 Einige Grundbegriffe .....	295
8.2 Tiefes Q-Netz.....	298
8.2.1 Strategie zur Maximierung der Summe der Belohnungen.....	298
8.2.2 Eine kleine Navigationsaufgabe.....	298
8.2.3 Diskontierter zukünftiger Gewinn belohnt schnelle Lösungen .....	299
8.2.4 Die Q-Funktion bewertet Zustands-Aktionspaare .....	299
8.2.5 Die Bellmangleichung stellt eine Beziehung zwischen Q-Werten her .....	300
8.2.6 Approximation der Q-Funktion durch ein tiefes neuronales Netz ..	301
8.2.7 Q-Lernen: Training eines tiefen Q-Netzes.....	301
8.3 Anwendung von Q-Lernen auf Atari Videospiele .....	305
8.3.1 Definition des Spielzustands bei Atari-Spielen .....	305
8.3.2 Architektur des Atari-Netzes .....	306
8.3.3 Training .....	306
8.3.4 Auswertung der tiefen neuronalen Netze der Atari-Videospiele ..	307
8.4 Strategiegradienten zum Erlernen von stochastischen Strategien .....	309
8.4.1 Notwendigkeit von Strategien mit Zufallselementen.....	309
8.4.2 Direkte Optimierung einer Strategie durch Strategiegradienten ..	310
8.4.3 Erweiterungen des Strategiegradienten: Actor-Critic und Proximal Policy Optimization .....	312
8.4.4 Anwendung auf Robotik und Go.....	314
8.4.5 Anwendung in Dota2.....	316
8.5 Selbstfahrende Autos .....	317
8.5.1 Sensoren selbstfahrender Autos.....	318
8.5.2 Funktionalität eines Agenten fürs autonome Fahren .....	319
8.5.3 Feintuning durch Simulation .....	322
8.6 Zuverlässigkeit des Bestärkungslernens .....	325
8.6.1 Training in Simulationsumgebungen oft schwierig übertragbar ..	325
8.6.2 Adversariale Attacken auf Modelle zum Bestärkungslernen .....	326
8.7 Zusammenfassung und Trends.....	327
Literatur.....	328
<b>9 Kreative Künstliche Intelligenz und Emotionen .....</b>	331
9.1 Bilder erzeugen mit generativen adversarialen Netzen (GAN) .....	331
9.1.1 Fälscher und Kunstexperte.....	331
9.1.2 Generator und Diskriminat...r.....	332
9.1.3 Optimierungskriterium für Generator und Diskriminat...r .....	332
9.1.4 Die Ergebnisse generativer adversarialer Netze .....	333
9.1.5 Interpolation zwischen Bildern .....	337
9.1.6 Transformation von Bildern.....	338
9.1.7 Transformation von Bildern ohne Trainingspaare .....	339

9.1.8	Creative Adversarial Network . . . . .	342
9.1.9	Erzeugung von Bildern aus Text . . . . .	345
9.1.10	GAN-generierte Modelle von Personen in drei Dimensionen . . . . .	346
9.2	Texte verfassen. . . . .	347
9.2.1	Automatischer Reporter: Daten in Zeitungsmeldungen darstellen . . . . .	347
9.2.2	Generierung von längeren Geschichten. . . . .	347
9.2.3	GPT2 erfindet komplexe Geschichten . . . . .	348
9.3	Musik automatisch komponieren. . . . .	355
9.3.1	MuseNet komponiert Mischungen von Klassik und Pop. . . . .	356
9.3.2	Der Music Transformer erfindet Klavierstücke . . . . .	358
9.4	Emotionen und Persönlichkeit. . . . .	359
9.4.1	Ein XiaoIce Dialog . . . . .	359
9.4.2	Das Ziel: Personen zum Weiterreden animieren . . . . .	361
9.4.3	Architektur von XiaoIce . . . . .	362
9.4.4	Anzahl der Benutzerantworten als Optimierungskriterium . . . . .	364
9.4.5	Emotionale Empathie und Unterstützung . . . . .	366
9.5	Zusammenfassung und Trends. . . . .	369
Literatur.	372	
<b>10</b>	<b>KI und ihre Chancen, Herausforderungen und Risiken</b> . . . . .	375
10.1	Chancen für Wirtschaft und Gesellschaft. . . . .	378
10.1.1	Smart Home, mein Haus kümmert sich um mich . . . . .	378
10.1.2	Diagnose, Therapie, Pflege und Verwaltung in der Medizin . . . . .	380
10.1.3	Maschine Learning in der industriellen Anwendung. . . . .	386
10.1.4	Weitere Einsatzgebiete der KI . . . . .	389
10.2	Wirtschaftliche Auswirkungen und Zusammenhänge . . . . .	394
10.2.1	Die Monetarisierung von Daten . . . . .	394
10.2.2	Die neue digitale Servicewelt – KI as a service. . . . .	398
10.2.3	Große Unternehmen als Treiber der KI . . . . .	401
10.2.4	Die Auswirkungen auf den Arbeitsmarkt. . . . .	404
10.3	Gesellschaftliche Herausforderungen . . . . .	410
10.3.1	Herausforderungen der KI in der Medizin. . . . .	412
10.3.2	1984 2.0: KI als Instrument der Überwachung . . . . .	413
10.3.3	Krieg der Maschinen . . . . .	416
10.3.4	Starke Künstliche Intelligenz. . . . .	418
10.4	Methodische Herausforderungen . . . . .	420
10.4.1	Kombination von Daten und unscharfem Schließen . . . . .	421
10.4.2	Schnelles und Langsames Denken. . . . .	422
10.5	Vertrauen schaffen in die KI . . . . .	426
10.5.1	Wie baut man vertrauenswürdige KI-Systeme? . . . . .	429
10.5.2	Wie kann man tiefe neuronale Netze testen? . . . . .	430

10.5.3 Ist eine selbstbestimmte, effektive Nutzung eines KI-Systems möglich? . . . . .	431
10.5.4 Behandelt das KI-System alle Betroffenen fair? . . . . .	433
10.5.5 Sind Funktionsweise und Entscheidungen der KI nachvollziehbar? . . . . .	434
10.5.6 Ist das KI-System sicher gegenüber Angriffen, Unfällen und Fehlern? . . . . .	435
10.5.7 Funktioniert die KI-Komponente zuverlässig und ist sie robust? . . . . .	436
10.5.8 Schützt die KI die Privatsphäre und sonstige sensible Informationen? . . . . .	437
10.5.9 Die Herausforderungen für ein KI-Gütesiegel. . . . .	438
10.6 Zusammenfassung . . . . .	439
Literatur . . . . .	441
<b>A Anhang</b> . . . . .	445
<b>A.1 Glossar</b> . . . . .	447
<b>A.2 Verzeichnis der Abbildungen und deren Quellen</b> . . . . .	467
<b>A.3 Literatur zum Anhang</b> . . . . .	483
<b>A.4 Stichwortverzeichnis</b> . . . . .	489

---

## Über die Autoren



**Dr. Gerhard Paaß** wurde am 10.08.1949 in Langenfeld bei Köln geboren und hat seit 1976 als Mathematiker und Informatiker hochrangige wissenschaftliche Positionen im Themenfeld der Künstlichen Intelligenz bekleidet. Nach dem Abschluss seiner Promotion im Jahre 1985 zum Thema „Prognose und Asymptotik Bayesscher Modelle“ an der Universität Bonn arbeitete Dr. Paaß im Rahmen zahlreicher Forschungsaufenthalte an Universitäten im Ausland (China, USA, Australien, Japan). Darunter waren renommierte Einrichtungen wie die UC Berkeley in Kalifornien und die University of Technology in Brisbane. Als Mitarbeiter der Gesellschaft für Mathematik und Datenverarbeitung (GMD), dem heutigen Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS, war und ist Dr. Paaß bis heute ein gefragter Gutachter und Konferenzleiter auf internationalen Konferenzen, z. B. als Mitglied des Editorial Board der Zeitschrift „International Journal of Uncertainty, Fuzziness and Knowledge-based Systems“ oder als Workshopleiter auf der „Conference on Knowledge Discovery and Data Mining“ 2010 in Washington zum Thema Analyse von Texten für Sicherheitsanwendungen. Dr. Paaß ist Autor zahlreicher Publikationen und hat mehrere Best Paper Awards im Bereich der KI erhalten. Zudem ist er seit vielen Jahren als Dozent tätig und hat im Rahmen der Fraunhofer-Allianz Big Data und Künstliche Intelligenz ganz maßgeblich daran mitgearbeitet, das neue Berufsbild des Data Scientist zu definieren und auch in Deutschland erfolgreich zu etablieren. Als Lead Scientist des Fraunhofer IAIS hat Dr. Paaß an der Entwicklung zahlreicher Curricula in diesem Themenfeld mitgewirkt.



**Dr. Dirk Hecker** wurde am 29.08.1976 in Köln geboren. Er ist stellvertretender Institutsleiter des Fraunhofer-Instituts für Intelligente Analyse- und Informationssysteme IAIS und Geschäftsführer der Fraunhofer-Allianz Big Data und Künstliche Intelligenz, der größten Allianz der Fraunhofer-Gesellschaft mit mehr als 30 Mitgliedsinstituten. Zudem ist Dr. Hecker Mitglied des Direktoriums der Fraunhofer Academy, die sich als Weiterbildungseinrichtung der Fraunhofer-Gesellschaft an Fach- und Führungskräfte technologiegetriebener Unternehmen richtet. Dr. Hecker hat langjährige Erfahrung in der Leitung von Forschungs- und Industrieprojekten im Bereich Data Mining und Machine Learning. Seine aktuellen Arbeitsschwerpunkte liegen in den Bereichen Big Data Analytics, Predictive Analytics, Deep Learning und Explainable AI. Er studierte Geo-Informatik an den Universitäten Köln und Bonn und promovierte an der Universität zu Köln. Dr. Hecker ist Autor zahlreicher Publikationen zu den oben genannten Fachgebieten und als Fachexperte sowie Auditor zum Thema Künstliche Intelligenz in unterschiedlichsten Gremien aktiv.



# Was ist intelligent an Künstlicher Intelligenz?

1

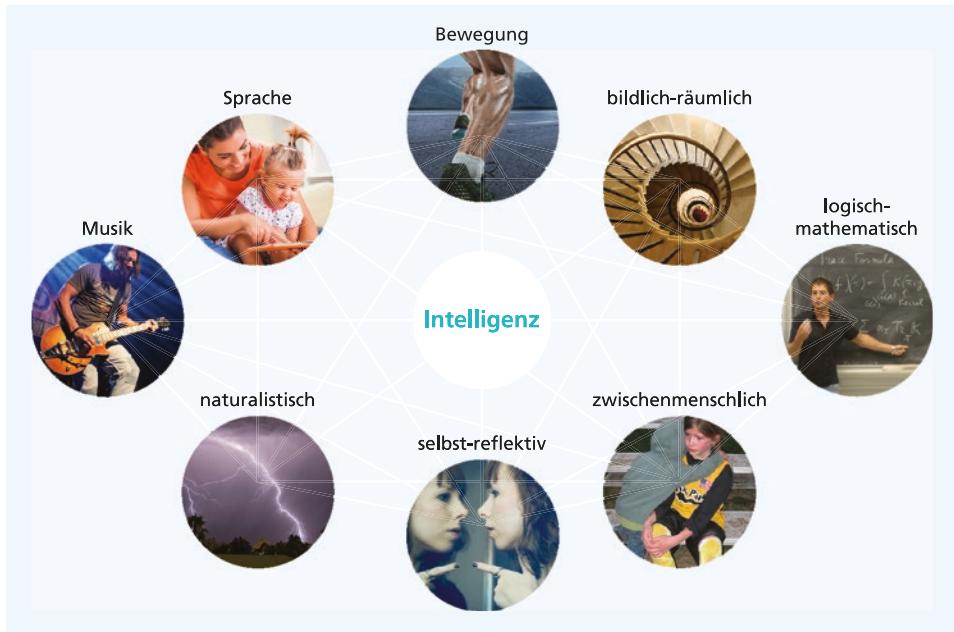
In der letzten Zeit ist der Begriff Künstliche Intelligenz (KI) in aller Munde. Presse, Parlemente und Regierungen betrachten KI als entscheidenden Motor für die weitere wirtschaftliche Entwicklung des Landes. Die Bundesregierung hat deswegen ein massives Programm zur Förderung der KI beschlossen (Álvarez 2018). Experten des Beratungsunternehmens McKinsey schätzen, dass KI weltweit ein Umsatzvolumen von ca. 12 Billionen Euro bis zum Jahre 2030 generieren wird (Tung 2018).

„Künstliche Intelligenz“ ist die Fähigkeit eines Computers oder computergesteuerten Roboters, Aufgaben zu lösen, die normalerweise von intelligenten Wesen erledigt werden“ (Copeland 2019). Das System soll in der Lage sein, sich ähnlich wie ein Mensch intelligent zu verhalten und selbstständig zu lernen. Allerdings ist diese Definition ungenau, da der Begriff „Intelligenz“ schwer abzugrenzen ist.

---

## 1.1 Menschliche Intelligenz hat viele Dimensionen

Es gibt eine Reihe von unterschiedlichen Beschreibungen der menschlichen Intelligenz. Gardner (1983) hat eine Theorie der multiplen Intelligenzen entwickelt, welche acht Dimensionen der Intelligenz aufführt (Abb. 1.1). Die Bewegungsintelligenz ist die Fähigkeit, den eigenen Körper zu fühlen und kontrolliert zu bewegen. Die bildlich-räumliche Intelligenz ermöglicht die Erkennung von Bildern und das Erfassen räumlicher Zusammenhänge. Zur Sprachintelligenz gehört das Verstehen von Sprache und die angemessene sprachliche Formulierung von Sachverhalten. Die logisch-mathematische Intelligenz ermöglicht die Analyse und Lösung logischer Probleme. Musikalische Intelligenz ist erforderlich zumverständnisvollen Hören von Musik und zum Musizieren. Die naturalistische Intelligenz umfasst die Fähigkeit, Natur zu beobachten, zu unterscheiden, zu erkennen, sowie eine Sensibilität für Naturphänomene zu entwickeln. Die zwischenmenschliche



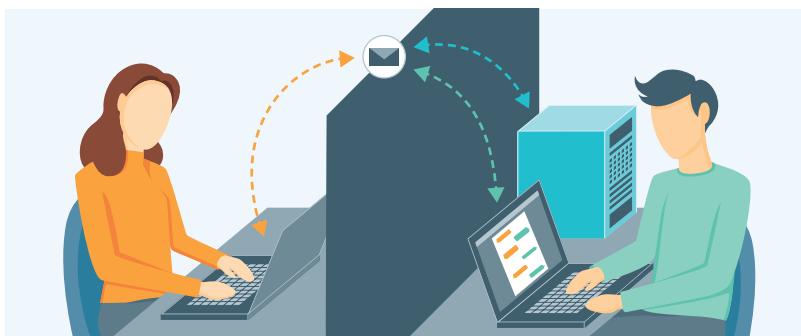
**Abb. 1.1** Die Dimensionen der menschlichen Intelligenz nach Gardner (1983). Bildnachweis im Anhang

oder emotionale Intelligenz ist das Vermögen, die Absichten, Gefühle und Motive anderer Menschen zu verstehen und vorherzusagen. Die selbstreflektive Intelligenz umfasst die Begabung, die eigenen Stimmungen, Antriebe, Motive und Gefühle zu erkennen. Hierzu gehört auch ein Bewusstsein der eigenen Person und die Fähigkeit, das eigene Verhalten in neuen Situationen vorherzusagen und sich zu Handlungen zu motivieren. Wir werden sehen, dass die KI mittlerweile auf viele – aber nicht alle – dieser Dimensionen anwendbar ist.

## 1.2 Woran erkennt man Künstliche Intelligenz

Um zu beurteilen, ob ein Computersystem intelligent ist, hat der britische Mathematiker Alan Turing ein Testverfahren – den Turing-Test – vorgeschlagen (Turing 1950). Im Rahmen des Tests kann ein menschlicher Schiedsrichter mit zwei Partnern elektronisch kommunizieren und beliebige Fragen stellen: der eine Partner ist ein Mensch, der andere ein Computer (Abb. 1.2). Wenn der Schiedsrichter nach vielen Fragen anhand der Antworten nicht entscheiden kann, welcher der Partner der Computer ist, so gilt der Computer als intelligent.

In Hinblick auf die oben dargestellten Dimensionen der Intelligenz muss der Turing-Test allerdings erweitert werden, damit auch andere Dimensionen (Bild, Bewegung, Sprache) erfasst werden können.



**Abb. 1.2** Beim Turing-Test stellt der Schiedsrichter (links) Fragen an Partner, die er nicht sehen kann: einen Menschen und einen Computer (rechts). Von beiden erhält er Antworten. Wenn der Schiedsrichter den Computer durch seine Antworten nicht vom menschlichen Partner unterscheiden kann, so muss der Computer ebenfalls intelligent sein.

Viele Forscher favorisieren mittlerweile neue Testkriterien, die stärker prüfen, wie tief das Verständnis einer Situation reicht. Zum Beispiel könnte der Schiedsrichter mit den Partnern über ein gezeigtes Netflix-Video sprechen. „Warum ist diese Szene mit Bill Murray lustig?“ etwa wäre eine Frage, die für einen Computer schwerer zu beantworten ist als „Erzähl mir von deiner Mutter!“.

Schon früh wurde versucht, Computer manuell so zu programmieren, dass sie intelligentes Verhalten zeigen. Leider führten diese Ansätze nur mit Einschränkungen zu dem gewünschten Erfolg. Als Alternative setzte sich das Vorgehen durch, ein lernfähiges Computerprogramm zu entwickeln. Dieses Lernverfahren trainiert dann mit Hilfe von Beispieldaten die gewünschte Funktionalität. Hierdurch ist es heute möglich, Teilaufgaben der KI befriedigend zu lösen. Beispiele sind die Diagnose von Krankheiten anhand von Symptomen oder Röntgenaufnahmen, die Transkription von gesprochener Sprache in Text oder die Erkennung von Objekten in Bildern.

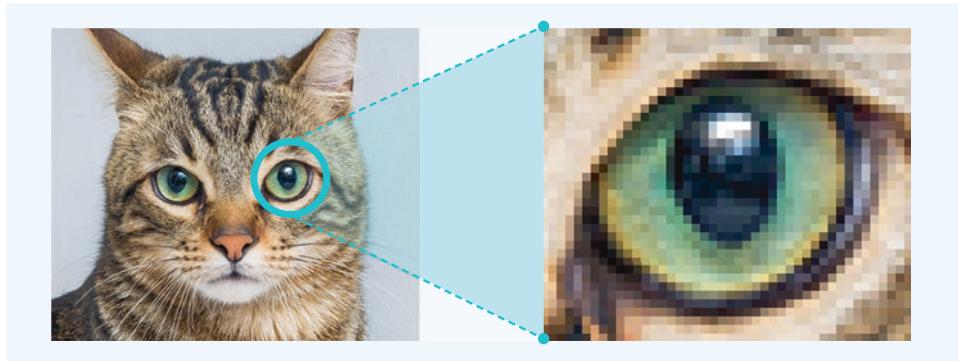
---

### 1.3 Computer lernen

Aber was heißt „Lernen“ bei einem Computersystem? Nehmen wir als Beispiel die Erkennung von Objekten in Bildern, z. B. einer Katze.

Als Eingabe erhält der Computer hier das Bild einer Katze (Abb. 1.3). Im rechten Teil von Abb. 1.3 sieht man einen vergrößerten Bildausschnitt, aus dem deutlich wird, dass das Bild der Katze ein Rechteck ist, das aus lauter kleinen quadratischen Farbflächen (Pixeln) besteht. Jedes dieser Pixel hat eine Farbe, welche man durch die Anteile der drei Grundfarben Rot, Grün und Blau kennzeichnen kann. Ein Pixel kann also durch ein Tripel von Zahlen beschrieben werden und das ganze Bild durch ein rechteckiges Schema von Zahlentripeln.

Der Computer erhält das Bild in Form eines rechteckigen Schemas von Zahlentripeln als Eingabe. Ziel ist es nun, dass der Computer das wichtigste Objekt in dem Bild benen-



**Abb. 1.3** Bild einer Katze und Bildausschnitt mit einzelnen Pixeln. Jedes Pixel wird durch drei Zahlen, die Farbwerte für Rot, Grün und Blau, beschrieben. Bildnachweis im Anhang

nen kann, in unserem Fall „Katze“. Diese Aufgabe nennt man Objektklassifikation in Bildern, eine Teilaufgabe der Bilderkennung. Es wird dem Computer also nicht mitgeteilt, wo im Bild das Objekt ist, das er benennen soll.

Frühe Lösungsansätze für diese Aufgabe versuchten, zunächst vorgegebene Teile der Bildobjekte zu erkennen, z. B. Ecken, Kanten, Linien und Flächen. Die größeren Objekte (z. B. Auge) wurden dann als Verbindungen der kleineren Teile rekonstruiert. Allerdings brachte dieses Vorgehen keine guten Resultate.

In letzter Zeit wurden Verfahren erprobt, bei denen der Computer nicht mehr die vom Menschen definierten Eigenschaften (Ecken, Kanten, Linien und Flächen) verwendet. Vielmehr wählt er selbsttätig wichtige Merkmale aus, erkennt diese im Bild und nutzt sie dann zur Objektklassifikation. Dazu benötigt er allerdings eine große Menge von Beispielbildern, in denen das gesuchte Bildobjekt (z. B. Katze) vorkommt und auch Beispielbilder, in denen es nicht vorkommt. Nur so kann der Computer die Gemeinsamkeiten und Unterschiede der Objekte erkennen.

Grundlage der Objektklassifikation ist also eine große Menge von Beispielen, welche aus der Eingabe (Bild) und der zugehörigen Ausgabe, der Objektklasse (z. B. Affe, Katze, ...) bestehen (Abb. 1.4). Die Menge der Beispiele nennt man „Trainingsmenge“ oder auch „Trainingsdaten“. Die Elemente der Trainingsmenge nennt man auch Trainingsbeispiele.

Aufgabe des Computers ist nun, die Menge der Beispiele und der zugehörigen Objektklasse zu analysieren. Anschließend soll er selbsttätig eine Rechenvorschrift zu entwickeln, mit der die Objektklassen von neuen Objekten möglichst gut vorhergesagt werden können. Die Bestimmung einer solchen Rechenvorschrift nennt man „Lernen“. Die Situation ist vergleichbar mit der eines Kleinkindes, dem die Mutter, wie in Abb. 1.5, die Namen von Gegenständen im Bilderbuch sagt. Dabei lernt das Kind, wie es die verschiedenen Objekte voneinander unterscheiden und benennen kann.

Lernen wird als der Vorgang definiert, mit dem neue oder modifizierte Fähigkeiten, Wissensinhalte oder Verhaltensmuster erworben werden (Wikipedia-Learning 2019).

	Instanz 1	Instanz 2	Instanz 3
Eingabebild			
Objektklasse	Affe	Katze	Mensch

**Abb. 1.4** Trainingsdaten aus unterschiedlichen Klassen für ein Bilderkennungssystem. Jedes Trainingsbeispiel besteht aus einem Eingabebild und der zugehörigen Objektklasse. Es sind pro Klasse eine hohe Anzahl von Trainingsbeispielen erforderlich. Bildnachweis im Anhang



**Abb. 1.5** Eine Mutter zeigt ihrem Kind Objekte im Bilderbuch. Bildnachweis im Anhang

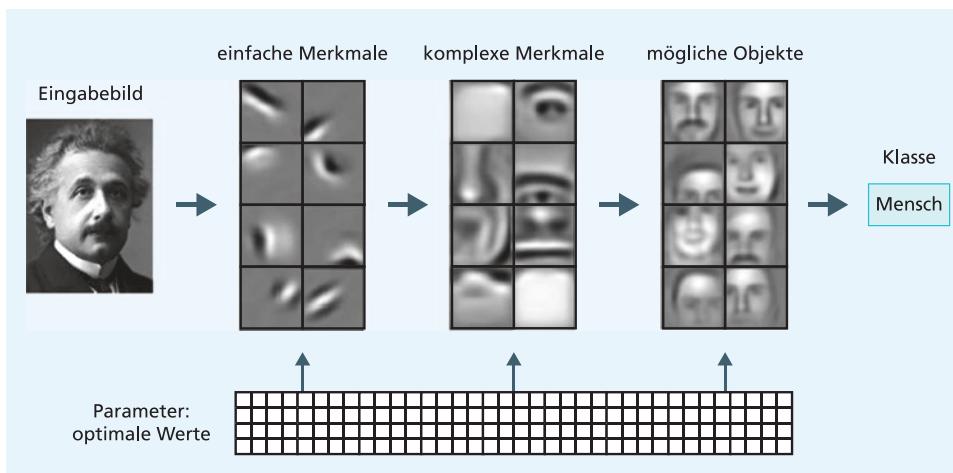
Gemeinhin wird Lernen als eine zutiefst menschliche Fähigkeit verstanden. Daher sind viele Leute nicht gewillt, einem Computerprogramm eine Lernfähigkeit zuzugestehen. Allerdings können auch Tiere lernen, wie viele Experimente aus der Biologie beweisen. Anders als bei lebenden Organismen, ist Lernen im Bereich der KI hingegen eher mit dem Begriff „trainieren“ gleichzusetzen: Hierbei kann das System die Fähigkeit erlangen, zu gegebenen Eingaben die passenden Ausgaben (z. B. Objektklasse) zu bestimmen. Dies bedeutet nicht, dass das System die Objekte der Trainingsmenge „auswendig“ lernt, son-

dern es kann auch neuen, noch nicht bearbeiteten Objekten die korrekte Klasse zuordnen. In diesem Sinne wird der Begriff „lernen“ in diesem Buch verwendet.

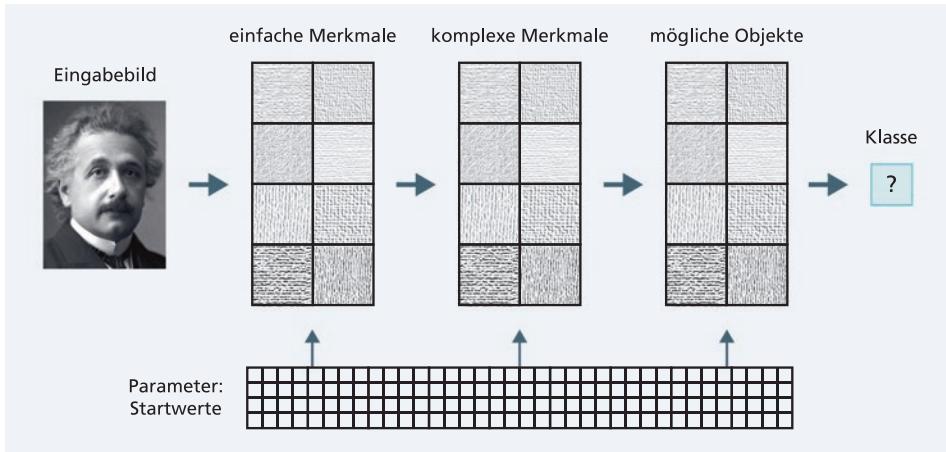
Es gibt eine Reihe anderer Verben, die normalerweise im Zusammenhang mit Menschen verwendet werden, aber auch im Bereich der KI auftauchen. Dazu gehören „erkennen“, „wissen“ usw. Übt der Mensch diese Tätigkeiten aus, so ist das immer verbunden mit menschlichem Bewusstsein und Emotionen. Im Bereich der KI werden diese Aspekte vollständig ausgeklammert. Dies muss man bei der Lektüre dieses Buch immer berücksichtigen.

## 1.4 Tiefe neuronale Netze können Objekte erkennen

Lernaufgaben, wie etwa die Objektklassifikation in Bildern, können heute von tiefen neuronalen Netzen durchgeführt werden. Wie in Abschn. 5.1.2 dargestellt wird, haben *tiefe neuronale Netze* (TNN) strukturelle Ähnlichkeiten mit der Informationsverarbeitung im Gehirn. Sie verarbeiten die Eingaben in einer Anzahl von aufeinanderfolgenden *Schichten* und transformieren die Eingabedaten in abstraktere Merkmale, die durch Pakete von Zahlen repräsentiert werden. In jeder Schicht werden ausgewählte Merkmale der Szene verarbeitet – je höher die Schicht, desto komplexer die Merkmale. Diese Merkmale wählt das System selbst aus. Abb. 1.6 zeigt die auf diese Weise von Lee et al. (2011) extrahierten Merkmale für die Klassifikation eines Objektes als Mensch. Schließlich lassen sich aus



**Abb. 1.6** Ein tiefes neuronales Netz (TNN) erhält eine Eingabe, z. B. das Bild einer Person. Aus diesem Bild werden in der unteren Schicht einfache Merkmale extrahiert, in den nachfolgenden Schichten komplexere Merkmale. Die Zuordnung zu einer Objektklasse findet in der letzten Schicht statt. Bildnachweis im Anhang



**Abb. 1.7** Berechnete Zwischenmerkmale des TNN und ausgegebene Klassifikation zu Beginn des Trainings mit zufällig ausgewählten Startwerten für das Parameter-Zahlenpaket. Das TNN hat noch nichts gelernt. Bildnachweis im Anhang

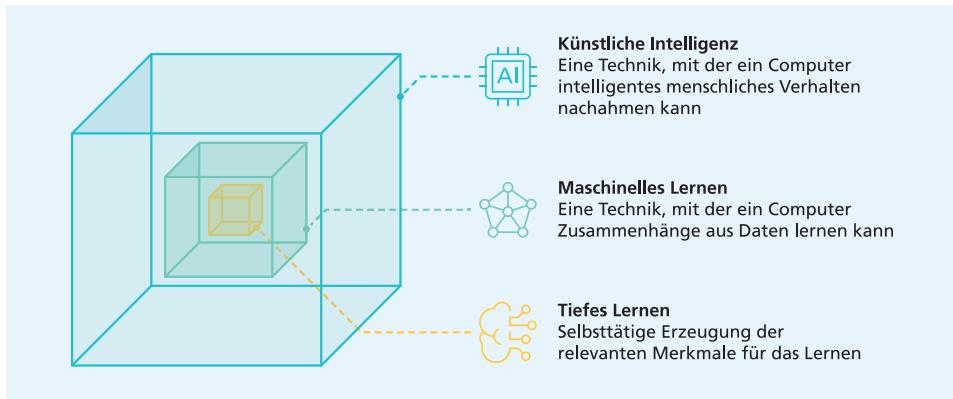
den Merkmalen der letzten Schicht in einfacher Weise die gewünschten Ergebnisse, z. B. die Namen der Objekte, bestimmen.

Allerdings kann das tiefe neuronale Netz Bilder nur erkennen, wenn seine Parameter angepasst wurden. Die Parameter sind ebenfalls eine Menge von Zahlen – ein Zahlenpaket – welches die Eigenschaften des TNN steuert. Der Aufbau des TNN und die Anzahl der Zahlen in dem Parameter-Zahlenpaket werden dabei von dem Konstrukteur des Netzes festgelegt. Das Parameter-Zahlenpaket wird zu Beginn mit zufälligen Zahlenwerten gefüllt. Wie in Abb. 1.7 gezeigt, kann das TNN zu in diesem Zustand weder sinnvolle Zwischenmerkmale erkennen noch das gezeigte Objekt identifizieren.

Wie oben dargestellt, werden die Werte des Parameter-Zahlenpaketes an eine große Menge von Trainingsbeispielen angepasst. Diese bestehen in der Regel aus der Eingabe (Bild) und der zugehörigen Ausgabe, also der Klasse des Bildobjektes (z. B. Affe, Katze, ...). Siehe Abb. 1.4). Meist sind für jede Klasse Hunderte solcher Trainingsbeispiele erforderlich. Der Computer passt nun die Werte des Parameter-Zahlenpaketes schrittweise so an, dass das TNN möglichst zu jedem Eingabebild die korrekte Klasse ausgibt. In den letzten Jahren ist es gelungen, auch Millionen von verschiedenen Parameterwerten durch sukzessive kleine Änderungen gleichzeitig so zu modifizieren, dass in einem hohen Prozentsatz der Fälle die korrekte Ausgabe erzeugt wird.

Dieses Vorgehen hat in letzter Zeit zu überraschend guten Ergebnissen in einer Vielzahl von intelligenten Erkennungsaufgaben geführt. Diesen Vorgang nennt man auch „*tiefes Lernen*“. Die Details dieses Lernvorgangs werden in den späteren Kapiteln dargelegt.

Tiefes Lernen ist eine spezielle Technik des *maschinellen Lernens*. Dieses umfasst alle Verfahren zur Suche von Mustern und Zusammenhängen in Daten (Abb. 1.8). Ein solches



**Abb. 1.8** Künstliche Intelligenz ist ein Oberbegriff von maschinellem Lernen, welches seinerseits das tiefe Lernen umfasst.

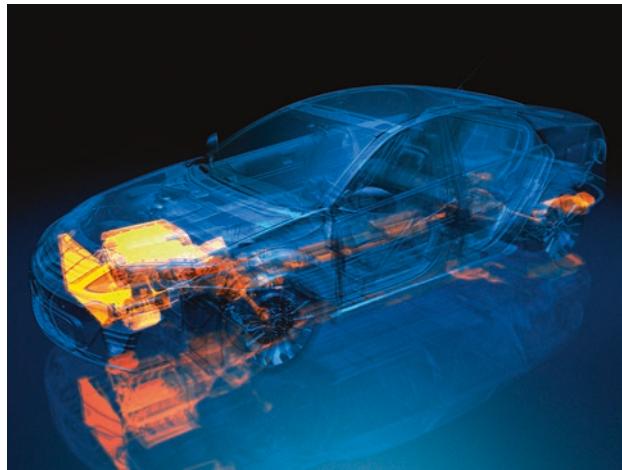
System kann beispielsweise die Niederschlagsmenge von morgen aus den Messwerten von Luftdruck, Temperatur und Windrichtung von heute vorhersagen.

Obwohl Künstliche Intelligenz und tiefe neuronale Netze in vielen Zeitschriftenbeiträgen und Talkshows diskutiert werden, liegt für die meisten Menschen die Funktionsweise dieser Computerprogramme im Dunkeln. Dieses Buch möchte daher für eine interessierte Öffentlichkeit klären, was Künstliche Intelligenz und tiefe neuronale Netze sind und wie sie funktionieren. Dabei sollen nicht nur die internen Mechanismen dargestellt, sondern auch die derzeitigen Möglichkeiten und Grenzen verdeutlicht werden.

## 1.5 Wie kann man Künstliche Intelligenz verstehen?

Die meisten Menschen werden von sich glauben, dass sie die Funktionsweise eines Autos verstehen. Abb. 1.9 zeigt das Funktionsschaubild eines Autos. Im Motor fangen Kolben den bei der Verbrennung entstehenden Druck auf und setzen ihn über die Kurbelwelle in eine Drehbewegung um. Das Getriebe in Zusammenspiel mit der Kupplung bestimmt die Geschwindigkeit der Drehbewegung, welche über das Differential auf die Räder übertragen wird. Dieser grobe Ablauf reicht den meisten Menschen für das Verständnis der Reaktionen des Autos bei der Steuerung durch den Fahrer aus. Dabei sind Details der elektronischen Motorsteuerung, des Getriebes mit seinen verwundenen Zahnradern, der Servolenkung, des Bremskraftverstärkers usw. extrem kompliziert und können ohne eine Ingenieurausbildung nicht verstanden werden.

Künstliche Intelligenz kann man auf einem ähnlichen Abstraktionsniveau verstehen. Hier werden nicht Kräfte durch mechanische Bauteile übertragen, sondern Zahlenpakete werden durch Operatoren geschickt, die nach einem einfachen Schema ein



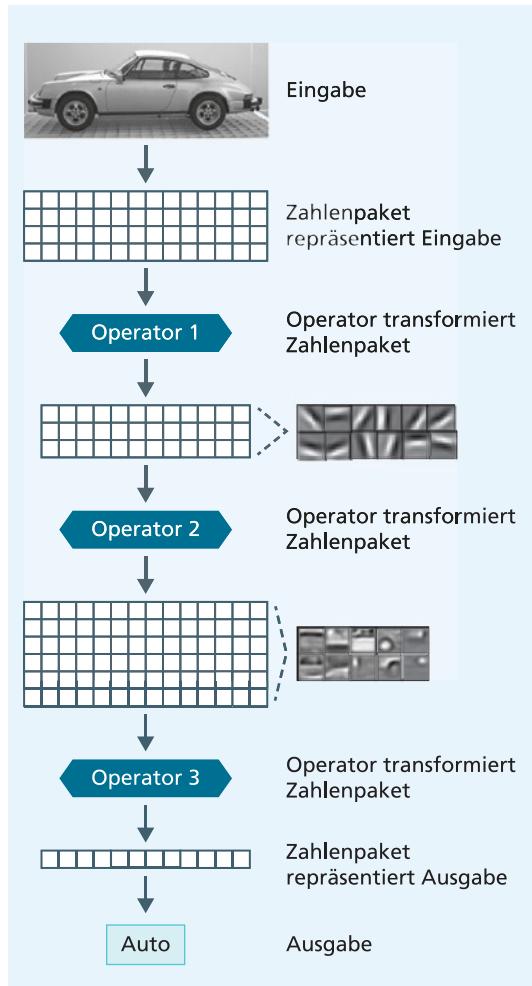
**Abb. 1.9** Funktionsschaubild eines Autos mit Motor, Getriebe, Antriebswelle, Differential und Rädern. Bildnachweis im Anhang

Eingabezahlenpaket in ein Ausgabezahlenpaket transformieren (Abb. 1.10). Die Eingabezahlenpakete repräsentieren die Eingaben der Anwendung, z. B. Bilder, Tonaufnahmen, Texte, Videos. Jeder Operator erzeugt ein neues Zahlenpaket, welches meist als Eingabezahlenpaket des nächsten Operators verwendet wird. Die Gesamtheit der verbundenen Operatoren wird ein Modell genannt. Das letzte Ausgabe-Zahlenpaket des Modells repräsentiert die gewünschte Antwort, z. B. eine Bildkategorie, eine Übersetzung, oder ein neues Bild, welche von dem Modell erzeugt wird.

Das Verständnis der Künstlichen Intelligenz in diesem Buch wird auf diesem hohen Abstraktionsniveau vermittelt. Es wird die grobe Funktion der einzelnen Operatoren erklärt, ähnlich wie die Erklärung von Motor, Getriebe und Differential beim Auto. Es wird der Fluss der Zahlenpakete durch das Modell dargestellt, analog zur Kraftübertragung beim Auto. Und es wird sehr grob die Wirkungsweise der Optimierungsverfahren erklärt, welche das Modell an die Trainingsbeispiele anpassen. Diese Verfahren werden in der Regel von den vorhandenen Programmierwerkzeugen bereitgestellt.

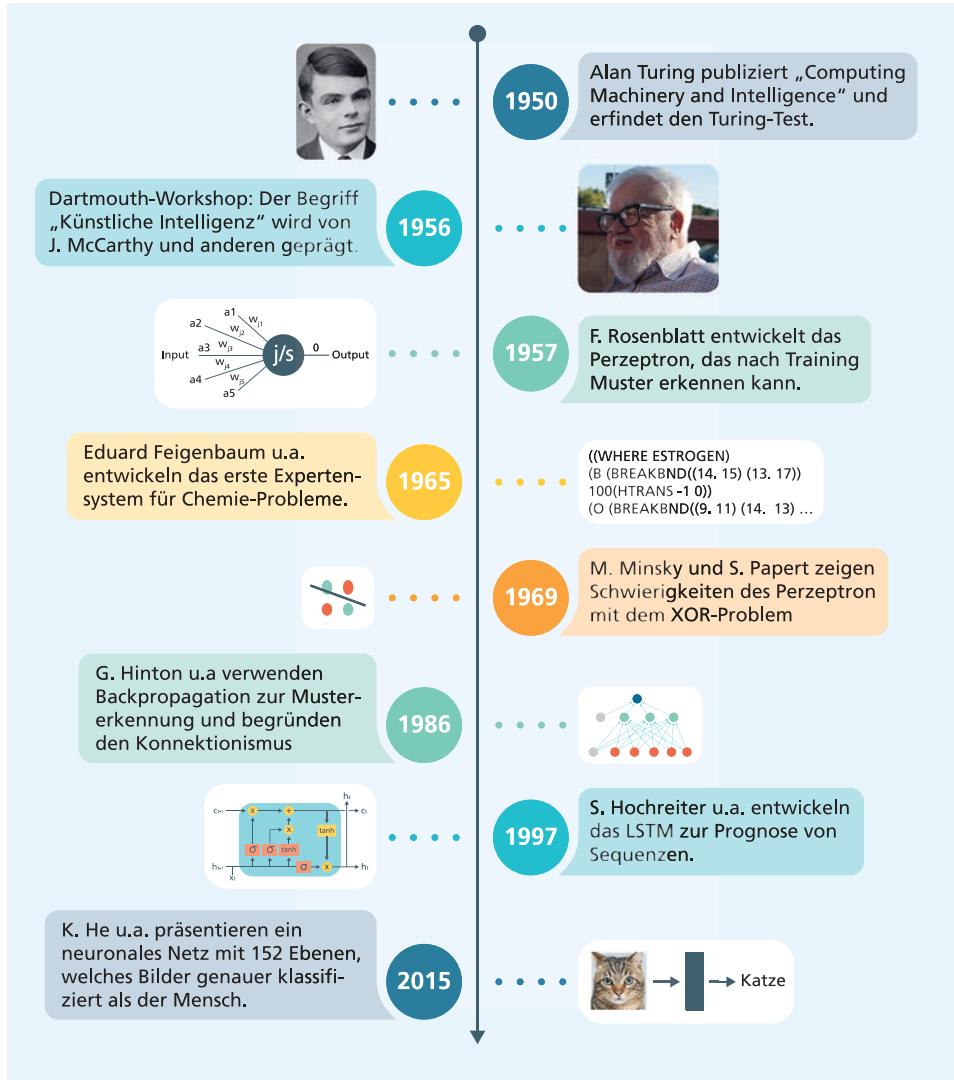
Das in diesem Buch vermittelte Bild von Künstlicher Intelligenz bleibt auf dieser relativ abstrakten Ebene. Viele Details sind sehr komplex, aber auch nicht für ein grundlegendes Verständnis erforderlich.

**Abb. 1.10** Funktions-  
schaubild eines Modells der  
künstlichen Intelligenz mit  
Eingabe, Zahlenpaketen,  
Operatoren und Ausgabe. Die  
Zahlenpakete repräsentieren je  
nach Schicht unterschiedliche  
Inhalte. Bildnachweis  
im Anhang



## 1.6 Die Geschichte der Künstlichen Intelligenz

Es ist aufschlussreich, sich die wechselvolle Geschichte der Künstlichen Intelligenz zu vergegenwärtigen (Abb. 1.11). Als Mitte des letzten Jahrhunderts die ersten programmierbaren Computer entwickelt wurden, dachten die Forscher schon bald darüber nach, ob diese Geräte auch intelligentes Verhalten an den Tag legen könnten. Zum Test eines Systems auf intelligentes Verhalten schlug Alan Turing 1950 den „Turing-Test“ vor. 1956 fand der von John McCarthy und Marvin Minsky veranstaltete Dartmouth Workshop statt, welcher den Begriff „Künstliche Intelligenz“ prägte. Ein Jahr später entwickelte Frank Rosenblatt ein neuronales Netz, das Perceptron, welches trainiert werden konnte, einfache Muster zu unterscheiden. Etwa zur gleichen Zeit wurden erste Programme zum logischen



**Abb. 1.11** Die wichtigsten Meilensteine in der Geschichte der Künstlichen Intelligenz. Bildnachweis im Anhang

Schließen vorgestellt. Eine Anwendung hiervon war das 1965 von Edward Feigenbaum u. a. vorgestellte Expertensystem DENDRAL, welches mithilfe von Regeln Probleme der organischen Chemie lösen konnte.

1969 zeigten Marvin Minsky und Seymour Papert, dass einlagige Perzeptrons keine komplizierten Probleme lösen können. Dies brachte die Forschung an neuronalen Netzen fast zum Erliegen. In dieser Zeit wird parallel die symbolische Künstliche Intelligenz entwickelt, welche intelligente Systeme durch logisches Schließen aus Fakten und Regeln

schaffen will. Erst Jahre später konnte gezeigt werden, dass mehrschichtige neuronale Netze mit nichtlinearen Elementen auch komplexe Zusammenhänge repräsentieren können. 1986 propagierten David Rumelhart, Geoffrey Hinton und Ronald Williams die Verwendung des Backpropagation-Algorithmus zum Training solcher Netze. Sie begründeten damit den Konnektionismus, der mentale Phänomene durch Netze einfacher Einheiten beschreiben will. Die Künstliche Intelligenz besteht in dieser Zeit aus zwei Lagern: der symbolischen KI sowie dem Konnektionismus. In den 90er-Jahren konnten weder die symbolischen Expertensysteme größere Probleme lösen, noch die neuronalen Netze mit den damals verfügbaren Computern komplexe Erkennungsprobleme bewältigen.

1997 schlagen Sepp Hochreiter und Jürgen Schmidhuber das LSTM vor, welches wesentlich bessere Ergebnisse bei der Modellierung von Sequenzen (Text, Spracherkennung) verspricht. Bis diese Vorteile realisiert werden können, vergeht aber noch eine Dekade, bis Grafikkarten mit einer hohen Rechenleistung verfügbar werden. Im Jahre 2015 wird von Kaiming He u. a. ein neuronales Netz mit 152 Schichten vorgestellt, welches Bilder besser erkennen kann als der Mensch. Ähnliche Erfolge werden in den darauffolgenden Jahren auch für die Übersetzung in andere Sprachen, die Spracherkennung und weitere Aufgaben gemeldet.

---

## 1.7 Zusammenfassung

Wenn Maschinen oder Computer kognitive oder geistige Fähigkeiten zeigen, die denen von Menschen ähneln, so nennt man das Künstliche Intelligenz. Bei diesen Fähigkeiten kann es sich z. B. um das Lernen aus Erfahrungen handeln oder um die Lösung von Problemen. Um zu prüfen, ob ein System intelligent ist, wird der Turing-Test eingesetzt, bei dem ein Prüfer mit einem Computer und einem intelligenten menschlichen Experten kommunizieren kann. Wenn der Prüfer nach ausgiebiger Kommunikation mit beiden Kommunikationspartnern nicht entscheiden kann, wer Computer und wer Mensch ist, dann – so ist die Schlussfolgerung – kann man den Computer als intelligent bezeichnen.

Die Beurteilung, ob eine Aufgabe Intelligenz erfordert oder nicht, hat sich in den letzten Dekaden stark verschoben. Zunächst galt das Schachspiel als eine der höchsten Intelligenzleistungen des Menschen. Dann wurden Computerprogramme entwickelt, die durch logische Bewertung der möglichen Schachzüge in der Lage waren, selbst den Schachweltmeister zu schlagen. Danach wurde Schachspielen als „mechanistisches“ Schließen abgewertet und zählte nicht mehr zum Kern der menschlichen Intelligenz (Abb. 1.12). Wenn ein Problem von einer Maschine gelöst werden kann, so wird oft anschließend festgestellt, dass die Problemlösung keine Intelligenz erfordert (McCorduck 2004, S. 204). Die Definition der „wahren“ menschlichen Intelligenz verändert sich also im Laufe der Zeit.

Der Intelligenzbegriff deckt Leistungen in sehr unterschiedlichen Anwendungsfeldern ab, von der bildlich-räumlichen Intelligenz über die Sprachintelligenz zu der zwischenmenschlichen, sozialen Intelligenz. Ziel des Forschungsgebietes der Künstlichen Intelligenz ist, einerseits Systeme zu entwickeln, die in all diesen Gebieten intelligente



**Abb. 1.12** Leistungen von Computern, die vom Publikum meist nicht mehr als „intelligent“ angesehen werden. Bildnachweis im Anhang

Leistungen erbringen können. Zum anderen ist auch der Wunsch vorhanden, mithilfe dieser Systeme zu verstehen, wie der Mensch in seinem Gehirn diese Intelligenzleistungen zustande bringt. Leider funktionieren die heutigen „intelligenten“ Computersysteme nach weitgehend anderen Prinzipien als das Gehirn des Menschen. Daher liegen die Mechanismen der menschlichen Intelligenz immer noch zu weiten Teilen im Dunkeln.

Computerprogramme der Künstlichen Intelligenz erhalten Informationen von außen in Form von Bildern, Texten, Tonsequenzen usw. Alle diese Informationen werden in Zahlenpakete verwandelt. Das Programm selbst besteht aus vielen „Schichten“ oder „Operatoren“, die Zahlenpakete als Eingabe erhalten und diese durch einfache mathematische Operationen (Addition, Multiplikation, Anwendung einfacher Funktionen) in neue Zahlenpakete transformieren. Die so erzeugten Ausgabezahlenpakete werden von anderen einfach strukturierten Operatoren weiterverarbeitet. Dabei wird die Eingabe in immer abstraktere Darstellungen verwandelt, die die wesentlichen Merkmale der Eingaben für die gesuchte Problemlösung immer besser abbilden. Schließlich kann der letzte Operator die gewünschte Ausgabe in einfacher Weise aus der letzten Repräsentation berechnen.

Das so definierte Programm wird tiefes neuronales Netz (TNN) genannt. Es enthält Parameter, welche selbst ein Zahlenpaket bilden, mit Millionen bis hin zu Milliarden Zahlenwerten. Diese Zahlenwerte werden durch Optimierungsverfahren so angepasst, dass die beobachteten Daten möglichst gut reproduziert werden können. Die Funktionsweise der einzelnen Operatoren kann gut verstanden werden und die Inhalte der intermediären Zahlenpakete können meist gut visualisiert werden. In diesem Sinne kann ein solches tiefes neuronales Netz verstanden werden.

---

## Literatur

- Álvarez, Sonja.** 2018. Deutschland will bei Künstlicher Intelligenz führend sein. [Online] 15. 11 2018. [Zitat vom: 22. 03. 2019.] <https://www.tagesspiegel.de/wirtschaft/strategie-der-bundesregierung-deutschland-will-bei-kuenstlicher-intelligenz-fuehrend-sein/23628138.html>.
- Copeland, B.J.** 2019. Artificial Intelligence. [Online] 2019. [Zitat vom: 22. 03. 2019.] <https://www.britannica.com/technology/artificial-intelligence>.
- Gardner, Howard.** 1983. *Frames of Mind, the theory of multiple intelligences*. New York : s.n., 1983.

- Lee, H., et al.** 2011. Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks. *Communications of the ACM*. 2011.
- McCorduck, Pamela.** 2004. *Machines Who Think* (2nd ed.). s.l. : Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1., 2004.
- Tung, Liam.** 2018. AI will create \$13 trillion in value by 2030 – But get ready to change your occupation. *ZDNet*. [Online] 10. 09 2018. [Zitat vom: 09. 07 2019.] <https://www.zdnet.com/article/mckinsey-ai-will-create-13-trillion-in-value-by-2013/>.
- Turing, Alan M.** 1950. Computing Machinery and Intelligence. *Mind. Band LIX, Nr. 236, 1950, ISSN 0026-4423, S. 433–460.* 1950.
- Wikipedia-Learning.** 2019. Learning. [Online] 2019. [Zitat vom: 06. 08. 2019.] <https://en.wikipedia.org/wiki/Learning>.



# Was kann Künstliche Intelligenz?

2

Schon in der Antike gab es Mythen und Sagen über künstliche Wesen, welche von ihrem Meister mit Intelligenz und Bewusstsein ausgestattet wurden (History 2019). Die Vorstellung von der Schaffung künstlicher Wesen ist also ein alter Menschheitstraum. Der Begriff der Künstlichen Intelligenz wurde auf einem Workshop in Dartmouth College im Jahre 1956 von John McCarthy geprägt (Moor 2006), kurz nachdem die ersten Computer entwickelt worden waren. Nach einer ersten Euphorie wurde klar, dass man die Schwierigkeiten bei der Erreichung dieses Ziels in grober Weise unterschätzt hatte. Erst in den letzten Jahren haben Fortschritte in der Rechenleistung der Computer und die Verfügbarkeit geeigneter Programmierumgebungen und Algorithmen dazu geführt, dass man einige Teilaufgaben der Künstlichen Intelligenz in befriedigender Weise lösen kann.

In diesem Kapitel werden einige der erreichten Lösungen dargestellt, um dem Leser einen Überblick über das Spektrum der Anwendungen zu geben. Die verwendeten Techniken werden in späteren Kapiteln erläutert. Einen aktuellen Überblick über den State of the Art bietet die Webseite Paperswithcode (2019).

---

## 2.1 Objekterkennung in Bildern

Wie schon auf Seite 2 beschrieben, hat die Objektklassifikation in Bildern das Ziel, unterschiedliche Klassen von Objekten (Hund, Katze, Auto, ...) in Bildern automatisch zu identifizieren. Die Identifikation der Objekte wird anhand von umfangreichen Trainingsdaten gelernt, welche zu jedem Bild die Klassennamen der abgebildeten Objekte beinhalten.

ImageNet (Deng et al. 2009) ist eine große Bilddatenbank für die Erprobung von Bilderkennungs-Software. Sie enthält mehr als 14 Millionen Bilder, die von Menschen manuell annotiert wurden. Dies geschah mithilfe von sogenannten Crowdworkern im Internet,

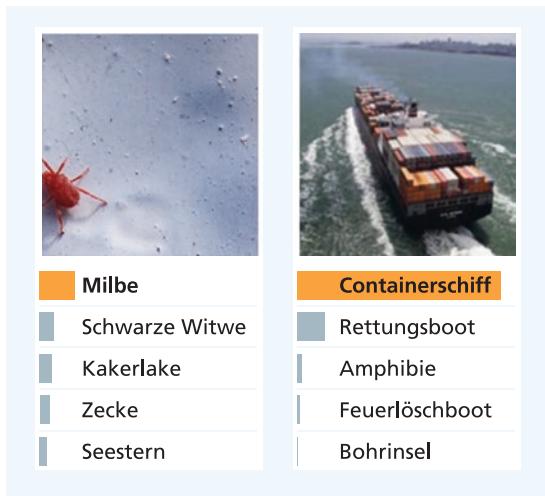
die für die Annotation jedes Bildes einen kleinen Geldbetrag, z. B. 20 Cent, erhielten. Die Annotation eines Bildes enthält die Objektklassen, die in einem Bild auftauchen. Es gibt bei ImageNet insgesamt mehr als 20.000 Objektklassen. Für die meisten Objektklassen wie etwa „Ballon“ oder „Erdbeere“ enthält ImageNet mehrere hundert Bilder.

Seit 2010 wurde jährlich ein Wettbewerb veranstaltet, in dem unterschiedliche Forschungsgruppen ihre Bilderkennungssoftware auf ein reduziertes Problem mit 1000 Objektklassen von ImageNet anwenden konnten. Die Qualität der Software wurde mit der Top5-Fehlerrate bewertet, welche misst, ob die wahre Objektkategorie eine der ersten fünf prognostizierten Kategorien ist. Die Prognosen sollen dabei nach ihrer Plausibilität angeordnet werden. Abb. 2.1 zeigt Beispielbilder aus ImageNet zusammen mit der tatsächlichen Objektkategorie und den fünf plausibelsten prognostizierten Kategorien.

Im Jahr 2011 wurde mit herkömmlichen Verfahren eine Top5-Fehlerrate von 25 % erreicht. Im Jahre 2012 gab es einen Durchbruch mit einer Top5-Fehlerrate von 16 %. Es handelte sich um AlexNet, ein tiefes neuronales Netz (TNN). Im Jahre 2015 erreichte ein anderes TNN eine Top5-Fehlerrate von 4,9 %. Dies ist besser als die Leistung menschlicher Annotatoren, welche in umfassenden Experimenten mit den gleichen Daten eine Top5-Fehlerrate von 5,1 % erzielten. Im Jahre 2017 erzielte ein neues neuronales Netz eine Top5-Fehlerrate von 3,8 % (Zoph et al. 2017). Die plausibelste Prognose war dabei in 17,3 % der Fälle nicht korrekt.

Damit sind die modernsten Systeme zur automatisierten Bilderkennung inzwischen praktisch genauso gut wie der Mensch: Wenn sie auf einer beliebigen Fotografie einen Gegenstand identifizieren sollen, machen sie im Schnitt genauso wenige Fehler wie ein geübter Betrachter. Die Technik der Bilderkennung wird in Kap. 5 vorgestellt.

**Abb. 2.1** Zwei Bilder aus der ImageNet Bilderkollektion. Darunter sind die vom System vorhergesagten Kategorien, geordnet nach abnehmender Plausibilität (= Balkenlänge). Die orangene Färbung markiert die korrekte Bildklasse. Beide Bilder wurden nicht zum Training verwendet.  
Quelle: (Krizhevsky et al. 2012) mit eigener Grafik



### 2.1.1 Medizinische Diagnose

Bilderkennungsverfahren lassen sich auch auf die Interpretation medizinischer Bilddaten anwenden. Die Stanford Universität hat z. B. ein Computersystem entwickelt, welches Hautkrebs mit hoher Präzision erkennen kann (Esteva et al. 2017). Dabei fütterte ein Team von Hautärzten und Informatikern ein Bilderkennungssystem mit etwa 130.000 Bildern unterschiedlicher Hautkrebsarten, aber auch mit Bildern normaler Haut und gutartigen Hautveränderungen. Insgesamt verwendeten sie 2000 verschiedene Objektklassen.

Abb. 2.2 enthält Beispielbilder von Hautkrebs und anderen Hautveränderungen. Sie zeigen, wie schwierig es ist, bösartige und gutartige Hautpartien voneinander zu unterscheiden. Um die Qualität der Diagnosen zu bewerten, wurden die Ergebnisse des Bilderkennungssystems mit den Diagnosen von 21 Dermatologen verglichen, die die gleichen Muttermale, Hautverletzungen beziehungsweise Melanome begutachtet hatten. Im Ergebnis zeigte sich, dass die automatische Diagnose genauso gut war wie die der Dermatologen. Das TNN für Hautkrebserkennung wird in Abb. 5.27 dargestellt.

Der Hautkrebs ist natürlich ein ideales Anwendungsfeld für die Computerdiagnostik. Er ist von außen sichtbar, er tritt häufig auf und eine frühe Entdeckung ist besonders wichtig, da dann die Heilungschancen am höchsten sind.

2018 wurde ein neues Ergebnis publiziert (Mar und Soyer 2018), welches einen anderen Datenbestand mit mehr als 100.000 Bilder von Hautpartien zum Training verwendete. Diesmal waren die Hautärzte in der Lage, 88,9 % der bösartigen Tumore zu entdecken, wohingegen das tiefe neuronale Netz in 95 % der Fälle korrekt war.

Insgesamt erkannte das tiefe neuronale Netz mehr Tumore als die Dermatologen und diagnostizierte weniger gutartige Hautpartien als bösartig. Das galt sogar dann noch, als die Dermatologen zusätzliche klinische Information über die Patienten erhielten. Dies zeigt,



**Abb. 2.2** Beispieldbilder zu Hautkrebs (oben) und anderen Hautveränderungen (unten). Insgesamt gibt es mehr als 2000 verschiedene Klassen von Hautveränderungen, deren Unterscheidung extrem schwierig ist. Bildnachweis im Anhang

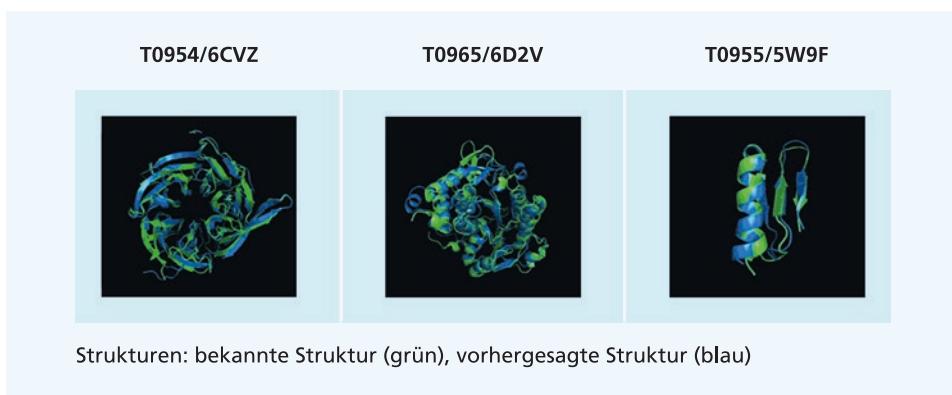
dass tiefe neuronale Netze mittlerweile bessere Ergebnisse liefern als erfahrene Hautärzte. Es ist geplant, in Zukunft eine Handy-App zur Verfügung zu stellen, mit der Patienten selbstständig verdächtige Hautstellen identifizieren können. Ähnliche Systeme wurden in vielen anderen Bereichen der Medizin publiziert, unter anderem für die Diagnose von Augenkrankheiten (De Fauw et al. 2018), Herzrhythmusstörungen (Hannun et al. 2019), die Interpretation von Röntgenaufnahmen des Brustkorbs (Irvin et al. 2019) und die Diagnose von 50 verschiedenen Arten von Tumoren aus der DNA im Blutstrom (Davis 2020).

## 2.1.2 Vorhersage der 3D-Struktur von Proteinen

Proteine sind große, komplexe Moleküle, aus denen unser Körper besteht und die Grundbausteine des Lebens sind. Nahezu alle Körperfunktionen – Verdauung, Atmung, Sehen etc. nutzen Proteine. Die Baupläne für Proteine sind in unserer DNA kodiert und heißen Gene.

Die Funktion eines Proteins wird von seiner 3D-Struktur bestimmt. Beispielsweise haben Antikörper zur Immunabwehr eine Y-Form und Kollagenproteine sind Fäden und können Kräfte übertragen. Es ist allerdings sehr schwierig, die räumliche Struktur der Proteine aus der Sequenz seiner Bestandteile vorherzusagen. Dabei sind nämlich Interaktionen zwischen den verschiedenen Aminosäuren zu berücksichtigen, die letztlich die 3D-Struktur bestimmen. Da viele Krankheiten auf Abweichungen in der räumlichen Anordnung der Proteine beruhen, ist die Prognose der 3D-Struktur extrem wichtig.

Der Google-Tochter DeepMind ist es gelungen (Evans et al. 2018), mithilfe von TNN die 3D-Struktur vieler Proteine korrekt vorherzusagen. Der Ansatz prognostizierte die Distanzen zwischen Paaren von Aminosäuren und die Winkel zwischen den chemischen Bindungen, die die Aminosäuren verbinden. Hiermit waren die Forscher in der Lage, den Raum der möglichen Konfigurationen zu durchsuchen, um passende 3D-Strukturen zu finden. Abb. 2.3 zeigt die vorhergesagten Strukturen (blau) und die tatsächlichen Strukturen



**Abb. 2.3** Die von tiefen neuronalen Netzen vorhergesagten (blau) und die zugehörigen korrekten (grün) 3d-Strukturen einiger Proteine. Die Bilder sind Screenshots einer 3D-Animation. Bildnachweis im Anhang

(grün) für einige Proteine. Hierbei wird einerseits die komplexe räumliche Struktur der bearbeiteten Proteine deutlich, zum anderen wird der Grad der Übereinstimmung zwischen tatsächlicher und prognostizierter Form sichtbar.

In der Chemie gibt es eine Reihe weitere Anwendungen der Künstlichen Intelligenz, z. B. die Planung der Synthese von organischen Molekülen. Segler et al. (2017) trainieren ein TNN mit 12 Millionen bekannten Reaktionen und können für 40 % der Test-Moleküle einen Syntheseplan erstellen.

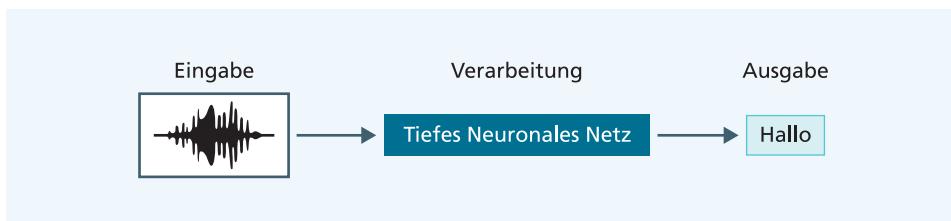
## 2.2 Spracherkennung

Die Spracherkennung hat das Ziel, gesprochene Sprache in Text zu transformieren. Sprache ist die natürlichste Art der Kommunikation für Menschen und angesichts von mehr als 5 Milliarden Handybesitzern auf der Welt wird die Spracherkennung eine sehr wichtige Rolle spielen. Schon in den 1980er-Jahren gab es dazu erste Computersysteme, welche aber noch eine hohe Fehlerrate aufwiesen. Mit Hilfe von TNN konnte man jetzt bessere Ergebnisse erzielen (Abb. 2.4).

Dieses System verwendete als Trainingsdaten den Switchboard-Korpus (Godfrey et al. 1992), der eine große Zahl von gesprochenen englischen Sätzen und den zugehörigen Transkriptionen in die Schriftform enthält. Es handelt sich um Telefongespräche zwischen wechselnden Personen, die über Themen wie Sport und Politik reden. In aufwändigen Versuchen wurde die Wortfehlerrate, d. h. die Anzahl der falsch erkannten Worte, professioneller menschlicher Zuhörer gemessen (Xiong et al. 2018). Sie betrug für diesen Datenbestand 5,9 %. Wurden die Konversationen mehrfach von einer Gruppe von Zuhörern transkribiert, so sank die Fehlerrate auf 5,1 %.

Schließlich wurde mithilfe der Daten ein TNN trainiert, welches eine Fehlerrate von 5,1 % erreichte (Xiong et al. 2018). Damit ist das System besser als einzelne menschliche Zuhörer und hat den gleichen Qualitätslevel wie ein Expertenteam.

Mittlerweile wird Spracherkennung schon von sehr vielen Nutzern verwendet, insbesondere bei sprachgesteuerten persönlichen Assistenten, wie Amazon Alexa und Google



**Abb. 2.4** Schema eines Spracherkennungssystems. Eingabe ist die numerische Repräsentation des Sprachsignals und Ausgabe ist der erkannte Text.

**Abb. 2.5** Spracherkennung ist eine zentrale Komponente von persönlichen Assistanten, mit der die Nutzer mündlich unterschiedliche Informationen und Dienste anfordern können. Bildnachweis im Anhang



Home (Abb. 2.5), mit denen die Nutzer unterschiedliche Dienste anfordern können. Die Spracherkennung ist meist internetbasiert, wird also auf Servern der Firmen mithilfe von TNN durchgeführt, kann aber auch bei aktuellen Smartphones offline erfolgen. Aktuelle TNN zur Spracherkennung werden in Kap. 7 dargestellt.

### 2.3 Maschinelle Übersetzung

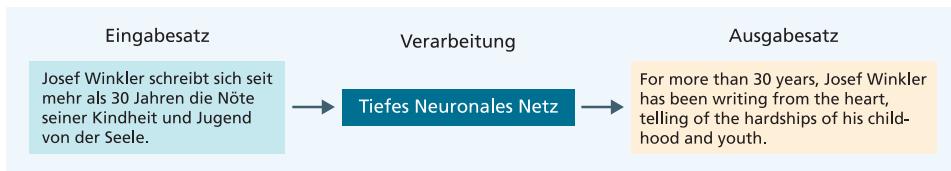
Als maschinelle Übersetzung bezeichnet man die computergestützte Übersetzung eines Textes aus einer Sprache in eine andere. Wie die Geschichte vom Turmbau zu Babel zeigt, ist der Wunsch, unbekannte Sprachen zu verstehen, schon sehr alt. Mit dem Aufkommen leistungsfähiger Computer wurden die Arbeiten an automatischen Übersetzungsprogrammen aufgenommen. Sie basierten insbesondere auf Wörterbüchern und grammatischen Analysen und zunehmend auf statistischen Verfahren.

Seit dem Jahre 2016 wurden tiefe neuronale Netze für die Übersetzung eingesetzt. Das tiefe neuronale Netz erhält die Trainingsdaten als eine Liste von Satzpaaren, welche aus einem Satz in der Quellsprache und dessen Übersetzung in der Zielsprache bestehen. Charakteristisch ist hierbei, dass die Übersetzung nicht Wort für Wort erfolgt, sondern Sätze als Ganzes übersetzt werden. Abb. 2.6 zeigt den Ablauf für einen Beispielsatz.

Maschinelle Übersetzungssysteme haben in den letzten Jahren rasante Fortschritte erlebt. Millionen Menschen nutzen täglich Online-Übersetzungssysteme und Handy-Apps, um über Sprachgrenzen hinweg zu kommunizieren. Microsoft hat 2017 eine neue Version seines Übersetzungssystems präsentiert (Hassan et al. 2018), das aus mehreren TNN besteht, aber als Ganzes trainiert wird. Für die Übersetzung vom Chinesischen ins Englische wurde eine Trainingsmenge von etwa 26 Millionen Satzpaaren verwendet. Bei einer Menge von Satzpaaren, die nicht zum Training verwendet wurden, konnte dieses System die gleiche Übersetzungsqualität wie menschliche Übersetzer erreichen. Dies wurde in umfangreichen Vergleichsexperimenten ermittelt. Überraschend dabei ist, dass das Programm weder Wörterbücher noch Grammatiken verwendet, sondern die Übersetzungen alleine aus den Satzpaaren lernt.

In jedem Jahr gibt es einen Wettbewerb auf der Conference for Machine Translation (WMT), auf der die besten Forschungsteams der Welt die Leistungsfähigkeit ihrer Übersetzungssysteme für vorgegebene Testdaten demonstrieren. Dies hat im Jahre 2018 auch für weitere Sprachpaare ergeben, dass die Genauigkeit der maschinellen Übersetzungs- systeme „sehr nahe an die Performanz menschlicher Übersetzer“ heranreicht (Bojar et al. 2018). TNN-Systeme zur maschinellen Übersetzung werden in Abschn. 6.5 im Detail beschrieben (Adams 1979).

In dem Buch „Per Anhalter durch die Galaxis“ (Adams 1979) ist der Babelfisch eine kleine Kreatur, die man ins Ohr stecken kann, und die dem Besitzer das sofortige Verständnis aller gesprochenen Sprachen ermöglicht. Für menschliche Übersetzer ist eine derartige Simultanübersetzung sehr schwierig, da er gleichzeitig zuhören und übersetzen muss (Abb. 2.7). Übersetzt man z. B. den Satz „Ich bin mit dem Zug nach Bonn gefahren“ ins Englische als „I traveled by train to Bonn“ kann das Wort „traveled“ erst ausgesprochen werden, wenn das Verb „gefahren“ bekannt ist. Diese lange Verzögerung ist für den



**Abb. 2.6** Tiefes neuronales Netz zur Übersetzung. Es wird deutlich, dass die Übersetzung eines Satzes in eine andere Sprache zu völlig neuen Satzstrukturen führen kann.



**Abb. 2.7** Simultanübersetzer bei einer Veranstaltung der World Trade Organization 2017. Bildnachweis im Anhang

Hörer sehr ärgerlich. Die Qualität der Übersetzung ist schlecht, weil die Fehlerraten der Übersetzer nach einigen Minuten stark anwachsen. Zudem sind sie nach 15–30 Minuten so erschöpft, dass sie abgelöst werden müssen.

Ma et al. (2019) haben einen Durchbruch im Bereich der Simultanübersetzung erzielt. Ihre Übersetzungsmethode kommt mit nur einer kurzen Verzögerung aus, hat aber nahezu die Qualität der Übersetzung des komplett bekannten Satzes, d. h. keine relevanten Inhalte werden weggelassen. Derartige Verfahren haben das Potential, für Politiker, Geschäftsreisende und Touristen echte Gespräche mit anderssprachigen Menschen überhaupt erst zu ermöglichen. Aktuelle TNN zur Audio-nach-Audio-Übersetzung werden in Abschn. 6.6.7 beschrieben.

---

## 2.4 Die Beantwortung natürlichsprachiger Fragen

Automatisches Beantworten von Fragen ist eine Teilaufgabe der Künstlichen Intelligenz. Hierbei wird ein Computersystem erstellt, welches automatisch Fragen beantworten kann, die in natürlicher Sprache gestellt werden. Zur Beantwortung kann das System entweder eine Datenbank mit Informationen nutzen oder kann die relevanten Fakten aus einer unstrukturierten Menge von Textdokumenten extrahieren.

Das Programmsystem Watson wurde 2011 berühmt, als es in der Frageshow Jeopardy (Abb. 2.8) die beiden besten bisherigen Teilnehmer schlagen konnte (Ferrucci 2012). Watson ist eine semantische Suchmaschine, die eine 100 GB große Datenbank von Fakten und Zusammenhängen durchsuchen kann. Dabei werden nicht nur exakte Treffer gefunden, sondern auch Umschreibungen eines Sachverhalts entdeckt. Es besteht aus vielen verknüpften Funktionseinheiten, welche die Fragen verstehen und die Antworten suchen, bewerten und auswählen.



**Abb. 2.8** Jeopardy Wettbewerb zwischen dem Watson Computersystem (Mitte) und seinen menschlichen Gegnern. Bildnachweis im Anhang

Dabei wurden auch viele Techniken der symbolischen Künstlichen Intelligenz verwendet. Diese repräsentiert die Merkmale eines Problems durch logische Fakten und Regeln (z. B. authorVon[Dracula, Bram\_Stoker] und bornIn [Bram\_Stoker, Ireland]). Die Lösung eines Problems wird dann durch logische Schlussfolgerungen gefunden.

Leider enttäuschte das Watson-System bei weiteren Anwendungen, z. B. medizinischen Diagnosen, die hochgesteckten Erwartungen (Krempl 2018). Die Mediziner identifizierten „mehrere Beispiele von unsicheren und falschen Behandlungsempfehlungen“. Es zeigte sich, dass vorprogrammierte Suchstrategien nicht flexibel genug sind. Offenbar muss ein System selbst einen Weg entdecken, wie es Antworten finden kann. Dieser Einwand trifft auf viele Ansätze der symbolischen Künstlichen Intelligenz zu.

Um die Leistungsfähigkeit lernender Systeme zu demonstrieren, wurden eine Reihe von Standardaufgaben definiert, z. B. SQuAD (Stanford Question Answering Dataset) (Rajpurkar et al. 2016). Zu seiner Erstellung wurden zunächst Abschnitte von Wikipedia-Artikeln zufällig aus der gesamten Wikipedia ausgesucht. Zu diesen Abschnitten wurden mehr als 100.000 Fragen von Crowdworkern formuliert. Die Antwort zu diesen Fragen ist eine Folge von Worten aus dem jeweiligen Abschnitt. Fragen und Antworten sollten dabei möglichst unterschiedliche Formulierungen enthalten. Allerdings sollten zur Beantwortung keine komplexen Schlussfolgerungen aus mehreren Sätzen erforderlich sein. Abb. 2.9 zeigt eine typische Frage, den zugehörigen Wikipedia-Abschnitt, sowie die Antwort (Squad 2019).

Im letzten Jahr hat sich die Performanz der Frage-Antwort-Systeme bei der Lösung der SQuAD1.1-Aufgabe dramatisch erhöht. Insbesondere wurden tiefe neuronale Netze ver-

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).



Frage	Antwort
Who killed Harold II?	Duke William II

**Abb. 2.9** Frage aus der SQuAD1.1 Fragensammlung. Oben links ist der Abschnitt des Wikipedia-Artikels, auf den sich die Frage bezieht. Die Frage steht darunter und die Antwort unten rechts entspricht der orange umrandeten Passage im Text. Die Abbildung zeigt eine Darstellung der Schlacht von Hastings im Teppich von Bayeux. Bildnachweis im Anhang

wendet, welche ausschließlich mit den 100.000 Fragen, den Antworten und den Abschnitten der zugehörigen Wikipedia-Artikel trainiert wurden.

Devlin et al. (2018) präsentierten ein TNN zur Beantwortung von SQuAD-Fragen. Es ist in der Lage, Fragen mit weniger Fehlern zu beantworten als menschliche Leser. Dabei sind nur 6,8 % der Antworten falsch, im Vergleich zu 8,8 % beim Menschen. Hierzu muss das Netz offensichtlich ein Verständnis der Bedeutungen von Worten und alternativer Formulierungen entwickeln und den Kontext der Worte erfassen können. Zudem können die Verfahren direkt auf andere Sprachen übertragen werden: es werden keine Wörterbücher, Grammatiken oder weitere Lexika verwendet. Mittlerweile konzentriert sich die Forschung auf die Lösung komplexer Frageprobleme, bei denen die Antwort in großen Textmengen gefunden werden muss und bei denen Weltwissen und logisches Schließen erforderlich ist. Die Beantwortung von Fragen ist ein Teilproblem der Erfassung der Bedeutung von geschriebenem Text und wird in Abschn. 6.8.2 beschrieben.

Zum Besuch der Highschool müssen Schüler der 8. Klasse in New York einen Aufnahmetest bestehen. Das Programmsystem Aristo hat es geschafft, mehr als 90 % der Fragen des Tests korrekt zu beantworten (Metz 2019) und hätte damit den Zugang zur Highschool geschafft. Dabei wurden Fragen zu Bildern und Grafiken ausgeschlossen und lediglich Multiple-Choice-Fragen beantwortet. Zwei der Fragen sind in Abb. 2.10 wiedergegeben. Während die erste Frage Wissen verlangt, müssen bei der zweiten Frage logische Schlussfolgerungen gezogen werden. Bei den entsprechenden Tests nach der 12. Klasse konnte Aristo 83 % der Fragen korrekt beantworten.

Die Autoren von Aristo betrachten diese Aufgaben als verbesserte Varianten des Turing-Tests (Clark et al. 2019). Im Idealfall sollte ein Benchmark eine Vielzahl von Fähigkeiten auf natürliche und uneingeschränkte Weise testen und gleichzeitig klar messbar, verständlich und motivierend sein. Standardisierte Tests, insbesondere wissenschaftliche Prüfungen, sind ein seltenes Beispiel für Aufgaben, die diese Anforderungen erfüllen. Sie sind zwar keine umfassenden Prüfungen der maschinellen Intelligenz, erfordern jedoch einige Fähigkeiten, die eng mit der Intelligenz verbunden sind, darunter das Sprachverständnis, das logische Schließen und die Verwendung von gesundem Menschenverstand.

**Abb. 2.10** Zwei Beispielfragen aus dem Highschool-Test (Metz 2019)

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

## 2.5 Dialoge und persönliche Assistenten

In den letzten Jahren wurden persönliche intelligente Assistenten (Assistant 2019) wie Siri, Google Home oder Alexa populär, die mit dem Nutzer in gesprochener Sprache kommunizieren und Aufträge für ihn ausführen (Abb. 2.11). Virtuelle Assistenten verwenden Spracherkennungsmodule und können damit Anweisungen und Fragen des Nutzers in Text übertragen. Anschließend interpretiert ein weiterer Systemteil die Äußerungen des Nutzers, führt Aufträge aus oder stellt den Text einer Antwort zusammen. Die Antwort wird schließlich durch ein Teilsystem zur Generierung gesprochener Sprache erzeugt. Oft führt der Nutzer den Dialog weiter, beispielsweise um eine Anweisung zu präzisieren oder Nachfragen des Assistenten zu beantworten. Daher muss der virtuelle Assistent den Inhalt der vorherigen Äußerungen als Kontext nutzen, um den Sinn des Gesagten zu verstehen, und adäquat antworten oder handeln zu können.

Die Nutzer können dem System Fragen stellen (beispielsweise zu den Wetteraussichten), die aus dem Internet beantwortet werden können. Es können Kalendereinträge, E-Mails oder Aufgaben mit verbalen Anweisungen erstellt werden. Darüber hinaus können Musik und andere Medien abgespielt, Restaurants reserviert oder Einkäufe im Internet getätigkt werden. Schließlich kann sich der Nutzer auch einfach mit dem virtuellen Assistenten unterhalten (Smalltalk).

In einem von Amazon veranstalteten Wettbewerb wurde die Qualität der Antworten virtueller Assistenten bewertet (Khatri et al. 2018, S. 19). Es zeigte sich, dass die Antworten in 74 % der Fälle als passabel, gut oder exzellent bewertet wurden. Dies ist weit entfernt von idealen Bewertungen, hat sich aber in den letzten Jahren stetig verbessert. Die derzeitig von Sprachassistenten verwendeten TNN werden in den Abschn. 7.6, 7.7 und 9.4 diskutiert.

IBM hat ein Dialogsystem Project Debater entwickelt, welches eine längere Diskussion mit einem menschlichen Opponenten durchführen kann (Debater 2018). Die Diskutanten erhalten ein ihnen unbekanntes Thema. Beide machen dann ein jeweils vierminütiges

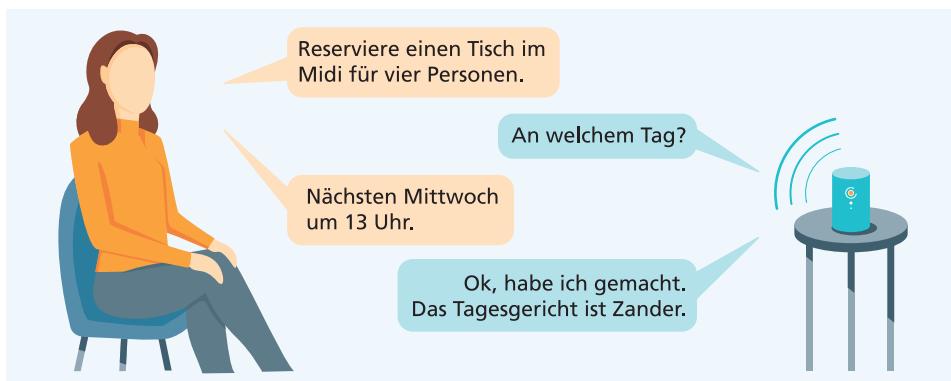
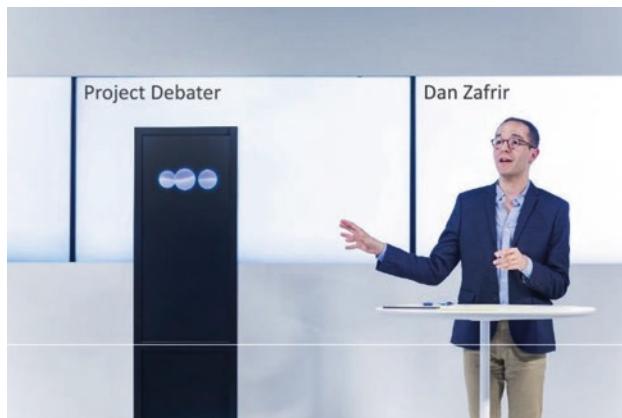


Abb. 2.11 Dialog mit einem persönlichen Assistenten.

Statement zum Thema und anschließend eine vierminütige Widerlegung der Argumente des Gegners. Das Computersystem hat dabei Zugang zu sehr großen Datenbeständen mit Millionen von Zeitungsartikeln. Das System muss dabei Fähigkeiten im Redenverfassen, Zuhören und Verstehen erwerben und eine Kette von Argumenten darlegen können. Im Juni 2018 wurde eine Schauveranstaltung durchgeführt mit zwei Debatten zu den Themen „Sollten wir die Erkundung des Weltraums finanzieren?“ und „Sollten wir die Telemedizin verstärkt nutzen?“ (Abb. 2.12). Die Maschine sprach mit einer selbstbewussten weiblichen Stimme mit natürlichem Sprachrhythmus und angemessenen Übergängen zwischen den Sätzen. Bei der anschließenden Befragung befanden die Zuhörer, dass die Beiträge des Dialogsystems sprachlich schlechter waren als die Beiträge der menschlichen Gegner, aber angemessenere Informationen und überzeugendere Argumente enthielten (Solon 2018).

Wie oben schon diskutiert ist die Erkennung gesprochener Sprache durch Computersysteme heute genauso gut wie bei menschlichen Zuhörern. Die Erzeugung von Sprache aus Text (Sprachsynthese) hat in der letzten Zeit ebenfalls einen Qualitätssprung durch den Einsatz tiefer neuronaler Netze erlebt (WaveNet 2016). Bei einer Vorführung des Google Duplex Systems konnten die Zuhörer kaum unterscheiden, ob der Mensch oder die Maschine sprach (Kremp 2018). Die Sprachmelodie der Äußerungen des Systems hörte sich sehr überzeugend an, und das System fügte kleine Unregelmäßigkeiten („äh“, „Mhmm“) in die Sätze ein und erzeugte auch Sprechpausen, als ob es nachdenken würde. Modelle zur Erzeugung gesprochener Sprache und die Struktur von Sprachassistenten werden ab Abschn. 7.5 dargestellt.



**Abb. 2.12** Diskutant vom Projekt Debater mit einem seiner menschlichen Opponenten. Bildnachweis im Anhang

## 2.6 Brettspiele

Bisher haben wir hauptsächlich Anwendungen der Künstlichen Intelligenz diskutiert, in denen es darum ging, ein Bild, ein Sprachsignal oder einen Text zu interpretieren und dazu eine adäquate Antwort zu erzeugen. Viele reale Probleme sind aber komplexer und haben einen zeitlichen Ablauf.

Ein Beispiel sind Spiele gegen einen Gegner. Hierbei empfängt das System sukzessive eine Reihe von Informationen, z. B. die nächste Stellung auf dem Schachbrett, und muss eine Aktion durchführen, z. B. den nächsten Schachzug wählen. Erst nach einer größeren Zahl solcher Aktionen wird deutlich, wie erfolgreich das System war: z. B. Gewinn, Verlust oder Remis.

Probleme dieser Art wurden in der mathematischen Spieltheorie und im Bereich des Operation Research schon lange diskutiert. Hier wurden Lösungen für einfache Probleme abgeleitet, die aber nur auf kleine Anzahlen möglicher Zustände (z. B. Stellungen beim Damespiel) anwendbar waren. Da aber z. B. auf dem Schachbrett mehr als  $10^{43}$  Stellungen möglich sind (mehr als das Gewicht der kompletten Milchstraße in kg), müssen Näherungslösungen für das theoretisch beste Verfahren gefunden werden.

Ein erster Erfolg gelang mit dem Schachprogramm Deep Blue (DeepBlue 1997), das 1997 den damaligen Schachweltmeister Garry Kasparov in einem Wettkampf in sechs Partien mit 3,5 : 2,5 schlagen konnte (Abb. 2.13). Deep Blue suchte heuristisch relevante Stellungen aus und konnte in einer Sekunde 200 Millionen Stellungen berechnen und mit einer Bewertungsfunktion deren Potential beurteilen. Heute ist die Rechnerleistung so stark gestiegen, dass Schachprogramme auf Smartphones auf Großmeister-Level spielen können.



**Abb. 2.13** Garry Kasparov spielt 2009 Simultanschach gegen andere Großmeister. Bildnachweis im Anhang

### 2.6.1 Das Strategiespiel Go

In China, Japan und Korea ist Go ein sehr populäres Spiel mit recht einfachen Regeln: Ziel des Spiels ist es, das größtmögliche Gebiet auf dem Spielfeld zu erlangen, indem man schwarze bzw. weiße Steine auf dem  $19 \times 19$  Felder großen Spielbrett platziert und gegnerische Steine schlägt. Ein Spiel dauert im Schnitt 150 Züge. Möglich sind mehr als  $10^{170}$  Stellungen, also mehr als es Atome im Universum gibt.

Die massive Suchtechnik von Deep Blue half hier nicht mehr weiter. Daher entwickelte Google DeepMind das Programm AlphaGo unter Verwendung von TNN. Hierbei wurde insbesondere Bestärkungslernen (engl. Reinforcement Learning) verwendet, was die besten Aktionen in einer Folge von Spielzügen liefert. Abb. 2.14 zeigt ein Turnier von AlphaGo gegen den südkoreanischen Großmeister Lee Sedol. In einem weiteren Turnier im Mai 2017 konnte AlphaGo den Weltranglistenersten Ke Jie dreimal schlagen (Bögeholz 2017).

Im Unterschied zu Deep Blue wurde AlphaGo nicht speziell für das Gewinnen des Go-Spiels programmiert, sondern verwendete ein TNN, welches seine Strategien aus einer großen Anzahl von gespeicherten Go-Partien lernte. Danach war es in der Lage, automatisch die vielversprechendsten Züge auszuwählen.

Das Nachfolgeprogramm AlphaZero ging einen Schritt weiter. Es bekommt nur die Regeln eines Spieles einprogrammiert (z. B. Go, Schach oder Shogi). Anschließend spielt das System gegen ein weiteres AlphaZero-System und erzeugt so die Daten, mit dessen Hilfe das tiefe neuronale Netz trainiert wird. Damit können immer bessere Versionen des TNN trainiert werden. Zum Schluss konnte es die jeweils besten verfügbaren Programme von Go, Schach und Shogi schlagen (Silver et al. 2018).

Das Erstaunliche ist, dass AlphaZero neue Züge und Strategien erfand, die bisher in der Spielercommunity nicht bekannt waren. Das Schachportal chess.com erklärte AlphaZero zum „gefährlichsten Gegner aller Zeiten“ (Pete 2018). AlphaZero opferte in bisher nicht bekannter Art Figuren, um schließlich doch zu gewinnen. „Sein Verständnis von Schach



**Abb. 2.14** Kommentierung des 3.Spiels von AlphaGo gegen den südkoreanischen Großmeister Lee Sedol. Bildnachweis im Anhang



**Abb. 2.15** Beim Pokern gegen fünf Gegner kommt es darauf an, Strategieänderungen jedes Spielers zu erkennen. Bildnachweis im Anhang

und rücksichtslosem, intuitivem Angriffsstil liegt so weit über dem anderer Gegner, dass diese nicht einmal merken, wenn sie verlieren.“

## 2.6.2 Künstliche Intelligenz gewinnt gegen fünf Poker-Professionals

Bei Pokern geht es darum, den Gegner im Unklaren zu lassen über die eigene Strategie. Allgemein unterscheidet man zwischen vier Spielertypen: konservativ-aggressiv, locker-aggressiv, konservativ-passiv und locker-passiv. Brown und Sandholm (2019) haben 2019 das Programmsystem „Pluribus“ entwickelt, welches gleichzeitig „No Limit Texas Hold’em“ gegen fünf unterschiedliche Gegner spielen kann. Beim Pokern mit mehreren Spielern ergeben sich wesentlich höhere Schwierigkeiten als bei zwei Gegenspielern (Abb. 2.15) und es kommt es darauf an, eine gute Strategie, taktisches Geschick und psychologisches Einfühlvermögen zu haben. Pluribus berücksichtigte nun, dass die Spieler jederzeit im Verlauf der Partien zu einer anderen Strategie wechseln können. Das Programm war in der Lage, bei 10.000 gespielten Händen mit sechs Spielern häufiger zu gewinnen als echte Poker-Experten, darunter Darren Elias, Rekord-Titelträger der World Poker Tour, und Chris Ferguson, sechsfacher Turniersieger der World Series of Poker.

---

## 2.7 Videospiele

Die Beherrschung von Videospielen durch den Computer ist weitaus schwieriger als die von Brettspielen. Dies liegt hauptsächlich daran, dass der Spieler den Inhalt und Ablauf der Videobilder interpretieren muss, um den nächsten Zug auszuwählen. Während eine

Brettposition mit wenigen Angaben beschrieben werden kann, muss für die Erkennung der aktuellen Spielsituation die Abfolge der Bilder des Videos analysiert werden.

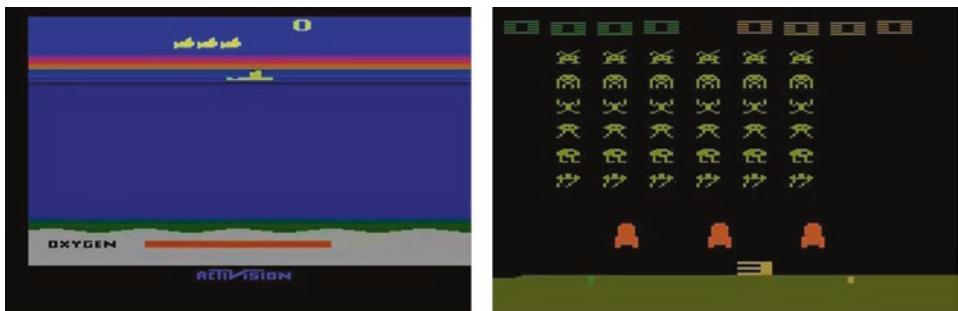
### 2.7.1 Atari 2600 Spielekonsole

Atari 2600 (Atari2600 1977) war eine weitverbreitete Spielkonsole von 1977 mit einem großen Angebot an Videospielen (Abb. 2.16). Diese hatten einen wesentlich einfacheren Aufbau als die heutigen fotorealistischen Videospiele und verfügten über eine Bildgröße von nur  $210 \times 160$  Pixeln.

Videospiele sind ideal dafür geeignet, dass ein oder mehrere Spieler durch Softwareagenten, auch Bots genannt, ersetzt werden. Im Jahre 2015 präsentierte Mnih et al. (2015) ein TNN mit Teilnetzen zur Bilderkennung und zum Bestärkungslernen, welches die Atari Spielkonsole steuern konnte. Diese Netze verwendeten lediglich Videobilder und den Spielstand als Input. Die Bots wurden mithilfe von Beispieldaten für die einzelnen Spiele trainiert, wobei bei jedem Spiel die gleiche Netzarchitektur verwendet wurde. Die Bots waren in der Lage, bei 29 von 49 Spielen bessere Ergebnisse als professionelle Spieletester zu erzielen. Mittlerweile wurde das Verfahren verbessert, indem der Planungshorizont erweitert und die Explorationsstrategien getunnt wurden. Die Bots konnten nun bei 40 von 42 Atari-Spielen durchschnittliche menschliche Spieler schlagen (Pohlen et al. 2018). Die Anwendung von Lerntechniken mit tiefen neuronale Netzen auf die Atari Videospiele wird in Abschn. 8.3 dargestellt.

### 2.7.2 Capture the Flag

Quake III Arena Capture the Flag ist ein populäres 3D-Videospiel, in dem zwei Mannschaften von Spielern gegeneinander spielen. Die Teams agieren in einer vorgegebenen Landschaft und haben die Aufgabe, die Fahne des anderen Teams zu erbeuten, dabei aber



**Abb. 2.16** Die Spiele Seaquest und Space Invaders für die Atari 2600 Spielekonsole. Bildnachweis im Anhang



**Abb. 2.17** Perspektive eines Bots von einer Spielszene aus dem Videospiel Quake III Capture the Flag. Bildnachweis im Anhang

die eigene Fahne zu beschützen (Abb. 2.17). Um sich einen taktischen Vorteil zu verschaffen, können sie die gegnerischen Teammitglieder markieren, um sie an ihre Ausgangspunkte zurückzusenden. Das Team, das nach fünf Minuten die meisten Flaggen erbeutet hat, gewinnt. Um das Spiel interessanter zu machen, wird die Landschaft nach jedem Spiel geändert.

Diese Art von Videospielen fasziniert Millionen von Spielern wegen des realistischen Eintauchens in eine Spielumgebung und wegen der Anforderungen an Strategie, Taktik, Hand-Augen-Koordination und der Kooperation mit anderen Spielern.

Google DeepMind hat eine Gruppe von Bots für das Capture-the-Flag Spiel trainiert (Jaderberg et al. 2018), wobei die Softwareagenten lediglich die Pixel des Videos ihrer Perspektive als Eingabe erhielten und die abschließende Information über Gewinn, Unentschieden oder Verlust. Jeder Agent wurde unabhängig von den anderen trainiert. Er entdeckte dabei die Spielregeln, unterschiedliche, komplementäre Strategien, und entwickelte sich zu einem Spezialisten für bestimmte Aufgaben. Als Nebenprodukt entwickelte sich auch die Fähigkeit zu Kooperation. Beispielsweise wurden die Strategien „Verteidigung der Heimatbasis“, „Besetzen der generischen Basis“, und „Einem Teamkollegen folgen“ gelernt (Capture 2018).

Um die Fähigkeiten der KI-Agenten zu testen, veranstaltete DeepMind ein Turnier mit Mannschaften aus jeweils zwei Spielern. Bestand ein Team nur aus Bots, so war es mit einer Gewinnwahrscheinlichkeit von 74 % am erfolgreichsten. Im Vergleich dazu gewannen durchschnittliche menschliche Spieler in 43 % der Fälle und starke menschliche Spieler in 52 % der Fälle. In einem separaten Wettbewerb spielten zwei professionelle Spieletester mit voller Kommunikation gegen ein festes Paar von Bots, die ja nicht kommunizieren konnten. Auch nach zwölf Stunden Übung konnten die menschlichen Spieletester nur in 25 % der Spiele (6,3 % Unentschieden) gegen das Team der Bots

gewinnen (Jaderberg et al. 2018, S. 5). Die KI-Agenten waren also eindeutig die besseren Spieler.

### 2.7.3 Das Echtzeit-Strategiespiel Dota2

Dota2 ist ein Echtzeit-Strategiespiel, das im Jahre 2013 veröffentlicht wurde und in dem zwei Teams gegeneinander antreten. Die Grafik und die Spielabläufe sind wesentlich komplexer als bei Quake Capture the Flag. Bei Dota2 steuert der Spieler eine Spielfigur aus der Vogelperspektive. Jeder Spieler gehört zu einer Gruppe von fünf Spielern (Dire oder Radiant). Es ist das Ziel des Spiels, den „Ancient“, das Hauptgebäude der jeweiligen Mannschaft, zu verteidigen und den gegnerischen zu zerstören (Abb. 2.18). Die Hauptgebäude der Teams liegen in einer Landschaft. Sie sind durch Türme gesichert und durch Wege verbunden. Eine Spielfigur kann an Stärke gewinnen, u. a. durch Sammeln von Gold und durch das Töten von generischen Spielfiguren. Jede der mehr als 100 Spielfiguren hat unterschiedliche Fertigkeiten und Talente.

OpenAI hat Bots für Dota2 trainiert, die dieses komplexe Spiel beherrschen (OpenAI-Five 2018). Als Input erhalten sie den Zustand des Spiels aus der Bot-Perspektive als einen Vektor von 20.000 Zahlen. Allerdings ist die Situation für einen Bot nur teilweise beobachtbar, viele Details bleiben den Bots verborgen. Jeder Bot hat Dutzende von Aktionsmöglichkeiten mit unterschiedlichen Intensitäten, die in 170.000 Einzelaktionen aufgeteilt wurden. Zum Training spielen die Bots täglich 180 Jahre realen Spielablauf in einer beschleunigten Spielgeschwindigkeit gegeneinander. Dabei wurde ein TNN für Bestärkungslernen verwendet. Im April 2019 hat sich auf einer Veranstaltung in San Francisco die Software OpenAI gegen den amtierenden Weltmeister, das E-Sports-Team OG, durchgesetzt (Steinlechner 2019). Dabei ist erstaunlich, dass der Softwareagent in der Lage war, eine Planung für ein Spiel von durchschnittlich 45 Minuten Dauer durchzuführen. Mit ähnlichen Ansätzen hat ein TNN von DeepMind Profispielern in dem komplexen Realzeit-

**Abb. 2.18** Spielszene von Dota2 aus der Sicht von „Death Prophet“, einem Mitglied des Teams von OpenAI, bei einer Attacke auf die Gegner. Jeder Agent hat unterschiedliche Fähigkeiten. Die Ressourcen von „Death Prophet“ sind unten zu sehen. Für jeden Teamkollegen ist dessen Stärke eingeblendet. Bildnachweis im Anhang



Strategiespiel StarCraft 2 geschlagen (Vinyals et al. 2019). Details zu den verwendeten Modellen finden sich in Abschn. 8.4.5.

---

## 2.8 Selbstfahrende Autos

Selbstfahrende Autos sind ein langgehegter Traum der Ingenieure. Schon 1977 wurde in Japan ein Wagen entwickelt, der Straßen mit weißen Markierungen folgen konnte (Self-Driving 2019). Im Jahre 2005 veranstaltete die DARPA einen Wettbewerb, der über 212 km durch wüstenähnliches Gelände in Nevada führte (Abb. 2.19). Die Streckeninformation bestand aus GPS-Koordinaten für Streckenpunkte im Abstand von 72 m, die zwei Stunden vor dem Start an alle Teams verteilt wurden. Von den dreiundzwanzig Startern erreichten fünf Wagen das Ziel.

Der Sieger war ein VW Touareg der Universität von Stanford mit einer Durchschnittsgeschwindigkeit von 30,7 km/h. Das Siegerteam unter der Leitung von Sebastian Thrun heimste ein Preisgeld von 2 Millionen Dollar ein. Der Touareg (Abb. 2.20) verwendete eine große Anzahl von Sensoren (Videokamera, Laser-Entfernungsmesser, Radar, GPS, Lagesensoren). Ein Softwaresystem mit etwa 30 Komponenten zur Positionsverfolgung, Pfadplanung und Steuerung verarbeitete die Sensorinformationen und steuerte das Fahrzeug. Lernende Komponenten wurden nur wenige eingesetzt. Thrun leitete später die Entwicklung von autonomen Autos bei Google (Abb. 8.27).



**Abb. 2.19** Der Beer Bottle Pass, Teil der DARPA Rennstrecke. Er liegt etwa 12 km vor der Ziellinie und hat über 20 Kehren. Bildnachweis im Anhang



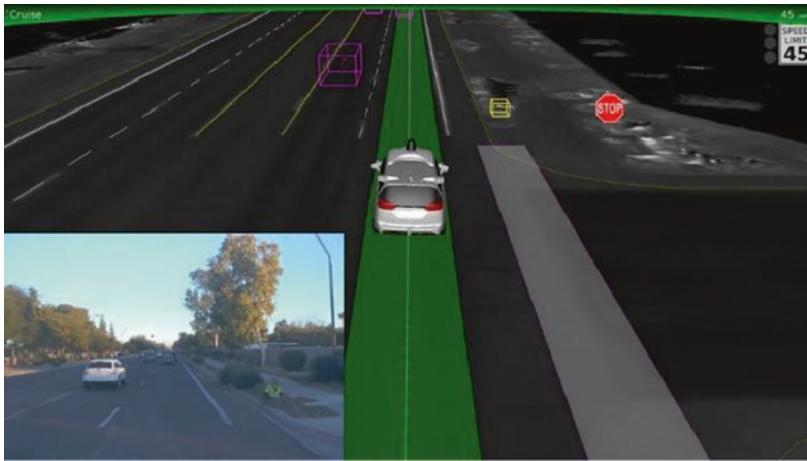
**Abb. 2.20** Mit diesem VW-Touareg mit diversen Sensoren auf dem Dach gewann das Team von Sebastian Thrun die DARPA Grand Challenge 2005. Bildnachweis im Anhang



**Abb. 2.21** Autonomes Auto von Waymo. Es handelt sich um einen umgerüsteten Minivan des Modells Chrysler Pacifica. Bildnachweis im Anhang

### 2.8.1 Weiterentwicklung der selbstfahrenden Autos

Seither entwickeln eine Reihe von Herstellern selbstfahrende Autos (Daimler, Honda, Audi, VW, Toyota, Tesla, Uber, GM etc.). Google bündelte seine Aktivitäten in der Firma Waymo, welche unterschiedliche Modelle mit der nötigen Sensorik ausstattete (Abb. 2.21). In den letzten Jahren hat Waymo verstärkt tiefe neuronale Netze eingesetzt, um die Leistungen der Sensoren zu verbessern und beispielsweise Fahrbahnmarkierungen, Hindernisse und Fußgänger auf einer Straße zu erkennen (Hawkins 2018). Abb. 2.22 zeigt eine von der Steuerungssoftware analysierte alltägliche Verkehrssituation, in der das autonome Fahrzeug die richtigen Steuerungsaktionen durchführen muss.



**Abb. 2.22** Erkennung der Verkehrssituation mit einem Bauarbeiter an einer Baustelle. Die Fahrzeuge auf den Fahrspuren sowie Personen sind durch Kuben repräsentiert. Verkehrsschilder werden gesondert gezeigt. Bildnachweis im Anhang

Entwickler von Roboterwagen versprechen, dass sie den Straßenverkehr viel sicherer machen werden – schließlich gingen über 90 Prozent der Unfälle auf Fehler von Menschen zurück. Im Oktober 2018 gab Waymo bekannt, dass seine Fahrzeuge 10 Millionen Meilen auf öffentlichen Straßen zurückgelegt haben. Während Tesla und andere Hersteller eine Reihe von Unfällen zu verzeichnen hatten, hat das Computersystem von Waymo in letzter Zeit nur einen Unfall mit Blechschaden verursacht (Vardhman 2019). Waymo-Sicherheitsfahrer mussten im Jahr 2018 im Schnitt nur alle 18.000 km eingreifen.

Ende Oktober 2018 erhielt Waymo als erste Firma von der zuständigen Behörde in Kalifornien die Erlaubnis, vollständig fahrerlose Autos (d. h. ohne menschliche Sicherheitsfahrer) zu betreiben (Waymo 2018). Waymo kündigte an, dass es in einem „early rider program“ mit Hunderten von ausgewählten Testkunden die selbstfahrenden Fahrzeuge optimieren will (Marshall 2018). Die Sensoren, die Modellkomponenten und die Lernstrategie selbstfahrender Autos werden ab Abschn. 8.5 diskutiert.

---

## 2.9 Der Computer als kreatives Medium

Computer sind bekannt für die Logik und Präzision, mit der sie wieder und wieder sehr komplexe Berechnungen durchführen können. Daher stößt die Idee, dass Computer kreativ sein können, auf eine weitverbreitete Ablehnung mit dem Argument: „Der Computer führt ja nur das aus, was einprogrammiert wurde.“ In diesem Abschnitt wird gezeigt, dass Computerprogramme in bestimmten Anwendungsbereichen durchaus kreativ sein können. Allerdings ist Kreativität ein schlecht definierter Begriff und bezeichnet eher das Neue, Unerwartete. Dies hat auch zur Folge, dass man Kreativität schlecht messen und oft

nicht sagen kann, ob ein Programm kreativer ist als ein Mensch. Im Folgenden werden einige Beispiele von Kreativität durch Künstliche Intelligenz zusammengestellt.

### 2.9.1 Neue Bilder komponieren

Im Oktober 2018 wurde beim Auktionshaus Christie's ein computergeneriertes Porträt einer Person „Portrait of Edmond Belamy“ für 432.500 \$ (Quackenbush 2018) verkauft. Das Portrait ist in Abb. 2.23 dargestellt und wurde mit einer statistischen Formel signiert. Es wurde von einem Algorithmus erstellt, welcher Malstile mithilfe einer Sammlung von 15.000 Porträts aus der Zeit von 14. bis zum 20. Jahrhundert gelernt hatte (Abschn. 9.1).

„Wir haben einige Arbeiten mit Akten und Landschaftsbildern durchgeführt und auch versucht, die Algorithmen mit dem Werk berühmter Maler zu füttern. Wir haben jedoch festgestellt, dass Porträts die beste Art und Weise sind, unseren Standpunkt zu veranschaulichen: Algorithmen sind in der Lage, Kreativität zu emulieren“, sagte Hugo Caselles-Dupré, einer der Schöpfer des Programms.

Ein ähnliches TNN ist in der Lage, fotorealistische Portraits von Personen zu generieren. Es sind sehr unterschiedliche Personen zu sehen: Weiße und Farbige, dicke und dünne Personen, Männer und Frauen. Sie haben Sommersprossen, Falten und Bartstoppeln. Abb. 2.24 zeigt ein Beispiel. Bei jedem Aufruf des Verfahrens wird ein anderes Portrait



**Abb. 2.23** Das „Portrait of Edmond Belamy“, ein von einem KI-Programm erzeugtes Gemälde, wurde bei Christie's für 432.500 \$ verkauft. Bildnachweis im Anhang



**Abb. 2.24** Ein hochauflösendes Portrait einer nichtexistierenden Person erzeugt mit einem TNN. Auf der Webseite <https://thispersondoesnotexist.com/> kann man sehr viele solche Bilder sehen (Karras 2019). Bildnachweis im Anhang

produziert. Die Portraits sind täuschend echt, aber die gezeigten Personen existieren nicht, sondern die Bilder wurden von einem Algorithmus erzeugt (Abschn. 9.1.4). Dieses TNN setzt nicht Teile von Bildern aus der Trainingsmenge zusammen, sondern hat die Struktur von Portraitfotos erfasst. Bei jedem Aufruf erzeugt es ein neues Bild, bei dem alle Bildteile aufeinander abgestimmt sind.

Derartige Algorithmen können in Zukunft eingesetzt werden, um realistische Personenbilder für Computerspiele und Werbeplakate zu entwickeln. Allerdings können computerproduzierte Bilder auch für Fake News oder die Verfälschung von Fakten verwendet werden. Mittlerweile kann man auch schon in Videos die Köpfe der Akteure austauschen und ihnen beliebige Äußerungen in den Mund legen (Agarwal et al. 2019). Daher muss man heute sehr genau aufpassen, wenn man ein realistisches Foto oder Video einer Szene sieht, mit der eine gewünschte Botschaft transportiert werden soll. Es ist sehr schwierig, solche künstlichen Portraits und Videos zu enttarnen. Anhaltspunkte finden sich manchmal in den Haaren oder bei den Übergängen zwischen verschiedenen Bildregionen. Tiefe neuronale Modelle zur kreativen Erzeugung von Bildern werden in Abschn. 9.1 dargestellt. Die synthetische Erzeugung von Videos wird in Abschn. 9.1.10 diskutiert.

## 2.9.2 Geschichten erfinden

Im Februar 2019 veröffentlichte die Forschungsinstitution OpenAI Ergebnisse eines TNN, welches darauf trainiert wurde, das nächste Wort eines Textes zu prognostizieren. Dabei berücksichtigt es den Satzbau, aber auch inhaltliche Bezüge zwischen den Wörtern. Das TNN wird mit 8 Millionen Webseiten mit insgesamt einem Umfang von 40 GB trainiert und kann aus diesen Webseiten sowohl die Grammatik einer Sprache als auch inhaltliche Zusammenhänge zwischen Begriffen lernen. Insgesamt hat das Modell 1,5 Milliarden freie Parameter und kann daher sehr viele Zusammenhänge speichern. Zum Training des Modells ist eine enorme Rechenleistung erforderlich.

Wie Abb. 2.25 zeigt, weisen die erzeugten Fortsetzungen einen flüssigen Stil und eine bisher ungesehene logische und argumentative Konsistenz auf. Auf den ersten Blick könnten sie von lebenden Menschen geschrieben sein, welche den Stil von „Herr der Ringe“ imitieren. Interessant ist, dass das Modell sehr viele der in den 8 Millionen Webseiten angesprochenen Themengebiete beherrscht, unter anderem Brexit, Genetik und atomare Technologie.

Die Autoren haben untersucht (Radford et al. 2018), ob das Modell einfach Versatzstücke aus verschiedenen Webtexten zusammenstückelt. Dies ist aber nicht der Fall. Die Inhalte werden neu formuliert und in einen logischen Handlungs- und Argumentationszusammenhang gebracht. Die Erzeugung der Texte ist stark zufallsabhängig: Jede Wiederholung ergibt eine komplett andere Fortsetzung. Die Autoren berichten, dass bei gut im

### Starttext:

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

### Fortsetzung:

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."

...

**Abb. 2.25** Automatisch generierte Fortsetzung des Starttextes durch ein tiefes neuronales Netzwerk (Radford et al. 2019)

Trainingsmaterial abgedeckten Themen jede zweite Fortsetzung brauchbar ist. Details dieses und anderer TNN zur kreativen Erzeugung von Text finden sich ab Abschn. 9.2.

## 2.10 Allgemeine Künstliche Intelligenz

Wir haben in den vorherigen Abschnitten eine Reihe von Anwendungsbeispielen gesehen, in denen KI-Systeme die Leistung von Menschen erreichen oder übertreffen. Die meisten Anwendungsbeispiele sind einzelne sensorische Fähigkeiten, z. B. Erkennung von Objekten in Bildern, Übertragung von Audiosignalen in geschriebenen Text oder die Übersetzung eines Textes in eine andere Sprache. Eine solche KI wird auch schwache KI genannt, weil es sich um einen Teilespekt der menschlichen Intelligenz handelt (Abb. 2.26 links).

Als Fernziel gibt es die allgemeine Künstliche Intelligenz (AKI), welche im Englischen Artificial General Intelligence (AGI) genannt wird. Dies ist ein Programmsystem, welches alle intellektuellen Fähigkeiten eines Menschen in sich vereint (Pennachin und Goertzel 2007). Ein solches KI-System löst nicht nur einen Teilespekt eines Problems, sondern verarbeitet gleichzeitig sensorische Signale unterschiedlicher Art, agiert in einer Umgebung und kann Probleme lösen (Abb. 2.26 rechts). Weitere Anforderungen sind ein „Bewusstsein“, mit dem das System seine Rolle reflektieren kann. Insgesamt soll eine AKI sämtliche acht Aspekte der menschlichen Intelligenz aus Abb. 1.1 abdecken.

Dieses Ziel ist bei Weitem noch nicht erreicht, wie im Verlauf der weiteren Kapitel klar werden wird. Allerdings gibt es schon KI-Systeme, die mehrere Aspekte der menschlichen Intelligenz abdecken. Ein Beispiel sind die KI-Systeme für Videospiele. Wie oben beschrieben, können diese unterschiedliche sensorische Eingaben verarbeiten (wenn auch in simulierten Umgebungen), eine optimale Strategie verfolgen und sogar mit anderen Agenten kooperieren. Die Frage der allgemeinen Künstlichen Intelligenz wird im letzten Kapitel des Buches (Abschn. 10.3.4) noch einmal aufgenommen.



Abb. 2.26 Schwache Künstliche Intelligenz und allgemeine Künstliche Intelligenz.

## 2.11 Zusammenfassung

Die aufgeführten Beispielanwendungen zeigen, dass die großen Fortschritte in der Künstlichen Intelligenz in den letzten Jahren hauptsächlich auf tiefe neuronale Netze zurückgehen, während die symbolische Künstliche Intelligenz eher eine Nebenrolle hat. Zu den Anwendungsbereichen der TNN gehören die Bildverarbeitung, Spracherkennung, Übersetzung in eine andere Sprache, Beantwortung von Fragen, natürlichsprachige Dialoge mit Personen, Brett- und Videospiele sowie selbstfahrende Autos.

Im Vordergrund steht dabei die Interpretation von Mediendaten, etwa von Bildern, gesprochener Sprache oder auch Texten. Es gibt aber auch integrierte Anwendungen, etwa im Spielebereich oder bei selbstfahrenden Autos, welche die Interpretation von Sensordaten, die Ableitung einer optimalen Handlungsstrategie und das Agieren in einer Welt umfassen. Ausschlaggebend für den Erfolg waren mehrere Faktoren, z. B. die Verfügbarkeit umfangreicher Trainingsdaten, sehr schnelle Parallelprozessoren, geeignete Modellarchitekturen und neue Toolboxen, mit denen die Modelle einfach formuliert und trainiert werden können.

Die vorherige Darstellung sollte allerdings nicht den Eindruck erwecken, dass TNN auch in der Praxis in jedem Fall gut funktionieren. In den oben aufgeführten Anwendungsbereichen haben oft große Teams von Experten in langwierigen Versuchsreihen die TNN so optimiert, dass die Performanz auf den Testdaten erreicht wurde. Dies kann zu einer sehr starken Ausrichtung des Modells auf den jeweiligen Datenbestand führen. In einer realen Anwendung sind Abweichungen und Datenfehler zu erwarten, sodass die Genauigkeit oft geringer ist. Wir werden diese Aspekte in den weiteren Kapiteln ausführlich diskutieren.

---

## Literatur

- Adams, Douglas. 1979.** *The Hitchhiker's Guide to the Galaxy*. s.l. : Del Rey, 1979. 0345391802.
- Agarwal, S., et al. 2019.** Protecting World Leaders Against Deep Fakes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 38–45). 2019.
- Assistant, Virtual. 2019.** Wikipedia Englisch. [Online] 2019. [https://en.wikipedia.org/wiki/Virtual\\_assistant](https://en.wikipedia.org/wiki/Virtual_assistant).
- Atari2600. 1977.** Atari 2600. [Online] 1977. [https://de.wikipedia.org/wiki/Atari\\_2600](https://de.wikipedia.org/wiki/Atari_2600).
- Bögeholz, H. 2017.** Künstliche Intelligenz: AlphaGo besiegt Ke Jie zum dritten Mal. [Online] 27. 05. 2017. [Zitat vom: 08. 05. 2017.] <https://www.heise.de/newsticker/meldung/Kuenstliche-Intelligenz-AlphaGo-besiegt-Ke-Jie-zum-dritten-Mal-3726711.html>.
- Bojar, O., et al. 2018.** Findings of the 2018 Conference on Machine Translation (WMT18). [https://www.research.ed.ac.uk/portal/files/76181549/Findings\\_of\\_the\\_2018\\_conference\\_on\\_machine\\_translation\\_WMT18.pdf](https://www.research.ed.ac.uk/portal/files/76181549/Findings_of_the_2018_conference_on_machine_translation_WMT18.pdf), 2018.
- Brown, N. und Sandholm, T. 2019.** Superhuman AI for multiplayer poker. *Science*, vol. 365, pp. 885–890. 2019.
- Capture, D. 2018.** Capture the Flag: the emergence of complex cooperative agents. [Online] 2018. [Zitat vom: 05. 03. 2019.] <https://deepmind.com/blog/capture-the-flag/>.

- Clark, P., et al. 2019.** From ‘F’ to ‘A’ on the N.Y. Regents Science Exams: An Overview of the Aristo Project. [Online] 2019. [Zitat vom: 09. 09. 2019.] arXiv preprint arXiv:1909.01958.
- Davis, Nicola. 2020.** New blood test can detect 50 types of cancer. *The Guardian*. 2020, 31.03.2020.
- De Fauw, J., et al. 2018.** Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9), 1342. 2018.
- Debater. 2018.** Project Debater. [Online] 2018. <https://www.research.ibm.com/artificial-intelligence/project-debater/>.
- DeepBlue. 1997.** Deep Blue. *Wikipedia*. [Online] 1997. [https://de.wikipedia.org/wiki/Deep\\_Blue](https://de.wikipedia.org/wiki/Deep_Blue).
- Deng, J., et al. 2009.** ImageNet: A large-scale hierarchical image database. *CVPR 2009 – IEEE Conference on Computer Vision and Pattern Recognition*. S. 248–255. 2009.
- Devlin, J., et al. 2018.** Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*. [Online] 2018. arXiv:1810.04805.
- 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. 2017.
- Evans, R., et al. 2018.** De novo structure prediction with deep-learning based scoring. [Online] 2018. [Zitat vom: 02. 09. 2019.] <https://deepmind.com/blog/article/alphafold>.
- Ferrucci, D. A. 2012.** Introduction to „this is watson“. *IBM Journal of Research and Development*, 56(3.4), 1–I. 2012.
- Godfrey, J. J., Holliman, E. C. und McDaniel, J. 1992.** SWITCHBOARD: Telephone speech corpus for research and development. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1992. ICASSP-92., (Vol. 1, pp. 517–520). I. 1992.
- Hannun, A. Y., et al. 2019.** Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature medicine*, 25(1), 65. 2019.
- Hassan, H., et al. 2018.** Achieving human parity on automatic chinese to english news translation. *arXiv preprint*. [Online] 2018. arXiv:1803.05567.
- Hawkins, Andrew. 2018.** waymo. *Inside Waymo’s strategy to grow the best brains for self-driving cars*. [Online] 2018. <https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview>.
- History. 2019.** History of Artificial Intelligence. [Online] 2019. [Zitat vom: 22. 03. 2019.] [https://en.wikipedia.org/wiki/History\\_of\\_artificial\\_intelligence](https://en.wikipedia.org/wiki/History_of_artificial_intelligence).
- Irvin, J., et al. 2019.** Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. [Online] 2019. [Zitat vom: 02. 09. 2019.] arXiv preprint arXiv:1901.07031.
- Jaderberg, M., et al. 2018.** Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint*. [Online] 2018. arXiv:1807.01281.
- Karras. 2019.** This Person does not exist. [Online] 2019. [Zitat vom: 19. 05. 2019.] <https://thispersondoesnotexist.com/>.
- Khatri, C., et al. 2018.** Advancing the state of the art in Open Domain Dialog Systems through the Alexa Prize. *arxiv*. [Online] 2018. <https://arxiv.org/ftp/arxiv/papers/1812/1812.10757.pdf>.
- Kremp, Matthias. 2018.** Google Duplex ist gruselig gut. *Spiegel Online*. [Online] 2018. <http://www.spiegel.de/netzwelt/web/google-duplex-auf-der-i-o-gruselig-gute-kuenstliche-intelligenz-a-1206938.html>.
- Kremp, Stefan. 2018.** Kampf gegen Krebs: Dr. Watson enttäuscht Erwartungen. *heise online*. [Online] 15. 08. 2018. <https://www.heise.de/newsticker/meldung/Kampf-gegen-Krebs-Dr-Watson-enttaeuscht-Erwartungen-4137203.html>.
- Krizhevsky, A., Sutskever, I. und Hinton, G. E. 2012.** Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105). 2012.
- Ma, C. Y., et al. 2019.** Self-Monitoring Navigation Agent via Auxiliary Progress Estimation. [Online] 2019. [Zitat vom: 21. 05. 2019.] arXiv preprint arXiv:1901.03035.
- Mar, V.J. und Soyer, H. 2018.** Artificial intelligence for melanoma diagnosis: How can we deliver on the promise? *Annals of Oncology*. 2018.

- Marshall, Adam.** 2018. Waymo Can Finally Bring Truly Driverless Cars to California. [Online] 30. 10. 2018. <https://www.wired.com/story/waymo-self-driving-cars-california/>.
- Metz, Cade.** 2019. A Breakthrough for A.I. Technology: Passing an 8th-Grade Science Test. *New York Times*. 2019, 04.09.2019.
- Mnih, V., et al.** 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529. 2015.
- Moor, J.** 2006. The Dartmouth College artificial intelligence conference: The next fifty years. *Ai Magazine*, 27(4), 87. 2006.
- OpenAIFive.** 2018. OpenAIFive. [Online] 2018. [Zitat vom: 05. 03. 2019.] <https://blog.openai.com/openai-five/>.
- Paperswithcode.** 2019. Browse state-of-the-art. [Online] 2019. [Zitat vom: 01. 04. 2019.] <https://paperswithcode.com/sota>.
- Pennachin, Cassio und Goertzel, Ben.** 2007. Contemporary Approaches to Artificial General Intelligence. [Buchverf.] Ben Goertzel und Cassio Pennachin. *Artificial general Intelligence*. Heidelberg : Springer , 2007.
- Pete.** 2018. Die 5 gefährlichsten Gegner aller Zeiten. [Online] 2018. [Zitat vom: 22. 03. 2019.] <https://www.chess.com/de/article/view/die-5-gefaehrlichsten-gegner-aller-zeiten>.
- Pohlen, T., Piot, B., Hester, T., Azar, M. G., Horgan, D., Budden, D., ... & Hessel, M.** 2018. Observe and Look Further: Achieving Consistent Performance on Atari. *arXiv preprint*. [Online] 2018. arXiv:1805.11593.
- Quackenbush, C.** 2018. A Painting Made by Artificial Intelligence Has Been Sold at Auction for \$432,500. [Online] 26. 10. 2018. [Zitat vom: 19. 05. 2019.] <http://time.com/5435683/artificial-intelligence-painting-christies/>.
- Radford, A., et al.** 2019. Better Language Models and their Implications. *OpenAI*. [Online] 2019. [Zitat vom: 14. 05. 2019.] <https://openai.com/blog/better-language-models/>.
- Radford, A., et al.** 2018. Improving language understanding by generative pre-training. [Online] 2018. [Zitat vom: 16. 05. 2019.] [https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language_understanding_paper.pdf).
- Rajpurkar, P., et al.** 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint*. [Online] 2016. arXiv:1606.05250.
- Segler, M. H., Preuss, M., & Waller, M. P.** 2017. Learning to plan chemical syntheses. [Online] 2017. [Zitat vom: 02. 09. 2019.] arXiv preprint arXiv:1708.04202.
- Self-Driving.** 2019. Self-driving car. [Online] 03. 01. 2019. [https://en.wikipedia.org/wiki/Self-driving\\_car](https://en.wikipedia.org/wiki/Self-driving_car).
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... & Lillicrap, T.** 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419), 1140–1144. 2018.
- Solon, Olivia.** 2018. Man 1, machine 1: landmark debate between AI and humans ends in draw. *Guardian*. 19. 06. 2018.
- Squad.** 2019. Squad Explorer. [Online] 2019. [Zitat vom: 22. 03. 2019.] <https://rajpurkar.github.io/SQuAD-explorer/>.
- Steinlechner, P.** 2019. Open AI besiegt die Weltmeister. *Golem*. [Online] 19. 04. 2019. [Zitat vom: 08. 05. 2019.] <https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister-1904-140710.html>.
- Vardhman, Raj.** 2019. 29 Self-Driving Car Statistics & Facts. [Online] 25. 09. 2019. [Zitat vom: 05. 10. 2019.] [https://carsurance.net/blog/self-driving-car-statistics/#Accident\\_Statistics\\_on\\_Googles\\_Self-Driving\\_Cars](https://carsurance.net/blog/self-driving-car-statistics/#Accident_Statistics_on_Googles_Self-Driving_Cars).

- Vinyals, Oriol, et al. 2019.** AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. [Online] 24. 01 2019. [Zitat vom: 02. 09. 2019.] <https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii>.
- WaveNet. 2016.** WaveNet: A Generative Model for Raw Audio. [Online] 2016. <https://deepmind.com/blog/wavenet-generative-model-raw-audio/>.
- Waymo. 2018.** Waymo. [Online] 2018. [Zitat vom: 05. 03. 2019.] <https://en.wikipedia.org/wiki/Waymo>.
- Xiong, W., et al. 2018.** The Microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5934–5938). IEEE. 2018.
- Zoph, B., et al. 2017.** Learning transferable architectures for scalable image recognition. [Online] 2017. arXiv preprint arXiv:1707.07012, 2(6).



# Einige Grundbegriffe des maschinellen Lernens

3

## 3.1 Die wichtigsten Arten des maschinellen Lernens

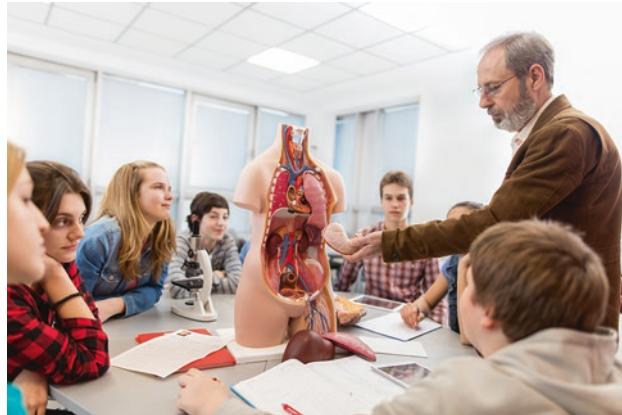
Der Mensch löst eine Vielzahl von realen Problemen, ohne dass er angeben kann, wie er das im Einzelnen macht. Ein Beispiel ist die Unterscheidung eines Hundes von einer Katze. Nahezu niemand kann sagen, auf welche Merkmale man denn genau achten muss, um einen Hund von einer Katze zu unterscheiden. Ähnliches gilt für komplexe Bewegungsabläufe wie etwa das Radfahren. Fast keiner kann im Einzelnen beschreiben, mit welchen Bewegungen er es schafft, das Gleichgewicht auf einem Fahrrad zu halten. Dies sind Beispiele von implizitem Wissen, welche wir nicht in Regeln oder Anweisungen fassen können.

Wenn man keine expliziten Vorschriften hat, mit denen man einen Computer für eine Aufgabe programmieren kann, so ist das Lernen aus Erfahrungen oder Daten eine Alternative. Wie schon oben diskutiert, lernt auch der Mensch aus Beispielen. Daher kam schon früh die Idee auf, dies auch bei Computern zu versuchen.

Das Lernen bei Computern ist ein Vorgang, bei dem sich der Computer Wissen aneignet durch die Auswertung von Erfahrungen. Bei diesem „maschinellen Lernen“ analysiert ein System die verfügbaren Daten und modifiziert sich schrittweise, sodass es seine Aufgabe besser erfüllen kann. Dabei kann man drei Arten von Lernaufgaben unterscheiden:

### 3.1.1 Überwachtes Lernen

Beim überwachten Lernen bekommt das System vorgegeben, was es lernen soll. Beispielsweise soll es Autos und Häuser in Bildern unterscheiden. Hierzu werden ihm viele Bilder mit Autos und Häusern vorgelegt, die manuell schon als Auto bzw. Haus markiert (annotiert) sind. Nachdem es eine große Anzahl solcher Beispiele verarbeitet hat, kann das



**Abb. 3.1** Überwachtes Lernen: Der Lehrer vermittelt Kindern die Namen von Objekten. Bildnachweis im Anhang

System Muster finden, durch welche sich die Objekte unterscheiden lassen. Anschließend ist das System in der Lage, diese Muster auf neue Bilder anzuwenden, um Autobilder von Hausbildern zu unterscheiden. Das System wird durch das Zeigen und Benennen von Beispielen ähnlich instruiert wie die Kinder der Schulkasse in Abb. 3.1.

Beim überwachten Lernen unterscheidet man noch zwei wichtige Anwendungsfälle. Bei der Klassifikation muss das System die Antwort aus einer meist kleinen Anzahl von Alternativen oder Klassen aussuchen. Ein Beispiel ist die Kategorisierung eines Produkts als „gut“ oder „defekt“. Bei dem anderen Anwendungsfall, der Regression, muss das System eine oder mehrere kontinuierliche Größen vorhersagen, z. B. die Höchsttemperatur und Windstärke für den nächsten Tag in Bonn.

### 3.1.2 Unüberwachtes Lernen

Beim unüberwachten Lernen muss das System ohne solche manuellen Vorgaben auskommen. Es liest die verfügbaren Daten und versucht selbsttätig, darin Muster und Regelmäßigkeiten zu finden (Abb. 3.2). Es kann dabei z. B. Daten anhand ihrer Ähnlichkeit gruppieren.

Besonders vielversprechend ist dieses Vorgehen, wenn die Daten unterschiedliche Komponenten aufweisen. Ein Beispiel sind die unterschiedlichen Wörter eines geschriebenen Satzes. Dann kann man versuchen, die eine Komponente (Wort) aus den anderen vorherzusagen. Diese Art von Lernen nennt man auch selbstüberwachtes Lernen. Es handelt sich hierbei um überwachtes Lernen, aber es sind keine Menschen notwendig um Annotationen zu liefern. Diese Form des Lernens ist in der Künstlichen Intelligenz weit verbreitet.

Für Yann LeCun, einen führenden KI-Forscher, wird selbstüberwachtes Lernen die Künstliche Intelligenz weit voranbringen. Erhält ein System beispielsweise die Aufgabe, jeweils die nächsten Bilder eines Videos zu prognostizieren, so muss es eine Repräsentation



**Abb. 3.2** Beim unüberwachten Lernen werden Muster und Regelmäßigkeiten in den Daten selbsttätig gefunden. Hier findet ein Kind die Zusammenhänge zwischen Wasser und Schwerkraft. Bildnachweis im Anhang

tion der Szenen entwickeln und die möglichen Bewegungen und Aktionen der Objekte vorhersagen. Bei genügend vielen Trainingsvideos wird es ein grundlegendes „Verständnis“ der Abläufe in Videos erlangen und kann dies bei der Prognose neuer Videoszenen anwenden (Smith 2020).

### 3.1.3 Bestärkungslernen

Beim Bestärkungslernen muss das System erst eine Reihe von Aktionen durchführen, bevor es das endgültige Ergebnis erfährt. Dies trifft z. B. auf Brettspiele (Schach, Abb. 3.3)



**Abb. 3.3** Beim Bestärkungslernen muss der Agent auf unterschiedliche Zustände mit einer Aktion reagieren. Dabei soll er eine Strategie lernen, mit der er in vielen Schritten eine möglichst hohe Gewinnsumme erzielen kann. Bildnachweis im Anhang

oder auch auf die Steuerung von Robotern zu. Nach jeder Aktion reagiert die Umwelt (z. B. der Gegner beim Schach) und das System erhält neue Informationen über den Zustand (z. B. die Brettpositionen), gegebenenfalls auch Belohnungen (z. B. Punkte, Sieg oder Niederlage quantifiziert durch einen Zahlenwert). Ziel des Systems ist es, eine Aktions-Strategie zu entwickeln, mit der es auf beliebige Zustände so reagieren kann, dass es insgesamt eine möglichst hohe Summe von Belohnungen erhält. Die Technik des Bestärkungslebens mithilfe von Belohnungen und Bestrafungen ist aus der Psychologie entlehnt und wird z. B. bei der Dressur von Hunden verwendet.

## 3.2 Programmieren und Lernen

In diesem Abschnitt werden die grundlegenden Abläufe des maschinellen Lernens für den einfachsten Fall, das überwachte Lernen, dargestellt.

### 3.2.1 Modelle transformieren eine Eingabe in eine Ausgabe

Wir haben im vorherigen Kapitel eine Reihe von erfolgreichen Anwendungen der tiefen neuronalen Netze (TNN) kennengelernt: Objekterkennung in Bildern, Spracherkennung zur Übertragung gesprochener Sprache in Text oder Übersetzung eines Satzes von einer Sprache in eine andere. Tiefe neuronale Netze sind spezielle Modelle maschinellen Lernens. Diese Modelle arbeiten beim überwachten Lernen nach folgendem Schema (Abb. 3.4):



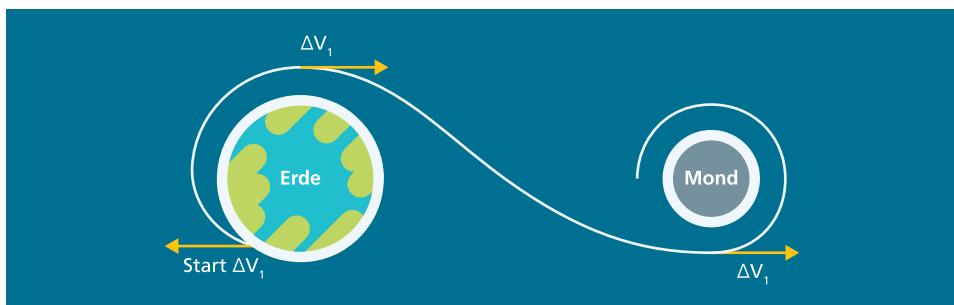
**Abb. 3.4** Im einfachsten Fall erhalten tiefe neuronale Netze eine Eingabe, führen eine Berechnung durch und produzieren eine Ausgabe. Bildnachweis im Anhang

1. Dem Modell wird eine Eingabe präsentiert, z. B. ein Bild, ein Audiosignal oder ein geschriebener Text.
2. Das Modell verarbeitet diese Eingabe und erzeugt eine Ausgabe.
3. Je nach dem Ziel der Analyse werden unterschiedliche Ausgaben erzeugt: z. B. die Benennung der Objekte in dem Bild, der Text aus dem Audiosignal oder die Übersetzung des Textes in die Zielsprache.

Mathematisch gesehen ist ein Modell daher eine Funktion  $f: x \rightarrow y$ , die aus einer Eingabe  $x$  eine Ausgabe  $y$  berechnet. Es ist daher auch ein Operator (Abschn. 1.5) und kann aus mehreren Teiloperatoren bestehen. Für jede Aufgabe gibt es ein unterschiedliches Modell, welches diese Aufgabe erledigen kann. Dieses wird trainiert anhand von Trainingsdaten, welche eine große Menge von Eingaben zusammen mit den zugehörigen korrekten Ausgaben enthalten. Um zu verstehen, wie dieses Training abläuft, muss man zunächst verstehen, wie das Modell aus den Eingaben die Ausgaben erstellen kann.

In seiner ursprünglichen Bedeutung ist ein Modell ein vereinfachtes Abbild der Wirklichkeit. Ein System, dessen komplexe Einzelheiten man nicht vollständig versteht, wird vereinfacht dargestellt. Beispiele sind physikalische Modelle, welche etwa die Bahn einer Rakete zum Mond darstellen (Abb. 3.5), ökonomische Modelle, welche das Wirtschaftswachstum für das nächste Jahr prognostizieren, oder meteorologische Modelle, welche Temperatur und Niederschlag für den nächsten Tag vorhersagen.

Im Bereich der Datenanalyse wurde dieses Konzept verallgemeinert. Dort ist ein Modell eine Beschreibung des Zusammenhangs zwischen beliebigen Eingaben und Ausgaben. Dieser Zusammenhang ist zu Beginn im Detail nicht bekannt und soll aus Daten rekonstruiert werden. Die Modellierung nimmt komplexe Probleme der realen Welt auf und packt sie in ordentlich formulierte mathematische Objekte, die Modelle. Sie können einer strengen Analyse unterzogen werden und sind für Computer besser geeignet. Die Modellbildung ist jedoch verlustbehaftet: Es kann nicht die gesamte Komplexität der realen Welt erfasst werden, und daher müssen Vereinfachungen gemacht werden. Der Modellkonstrukteur muss also entscheiden, welche Zusammenhänge und Aspekte er berücksichtigt und welche er ignoriert, indem er Modelle unterschiedlicher Komplexität erprobt.



**Abb. 3.5** Die Bahn einer Rakete von der Erde zum Mond kann mithilfe eines physikalischen Modells berechnet werden.

### 3.2.2 Algorithmen arbeiten schrittweise eine Liste von Anweisungen ab

Computer sind in der Lage, Daten zu speichern und zu verarbeiten. Die Verarbeitung findet mithilfe einer Reihe von Anweisungen statt, welche ebenfalls auf dem Computer gespeichert werden und in sehr hoher Geschwindigkeit nacheinander ausgeführt werden. Eine solche Folge von genau definierten Anweisungen nennt man Algorithmus.

Algorithmen kennt man auch aus dem täglichen Leben. In Abb. 3.6 ist der Algorithmus zur Berechnung des Preises eines Autos mit Mehrwertsteuer aufgeführt. Aber auch ein Kochrezept ist ein Algorithmus, wenn alle Zutaten genau aufgeführt sind und wenn die Ausführung aller Teilaufgaben, z. B. Braten oder Röhren, genau definiert sind.

Es ist meist nahezu unmöglich, ein Modell zu konstruieren, das auf Anhieb die gewünschten Ausgaben produziert. Vielmehr wird ein Modell anhand von Daten schrittweise so modifiziert (trainiert), dass es die gewünschten Ausgaben immer besser erzeugen kann. Im weiteren Verlauf des Kapitels haben wir die Aufgabe, das **Training eines Modells** als einen Algorithmus auf einem Computer zu realisieren. Wenn man sich die obige Einführung ansieht, so könnte man zu dem Schluss kommen, dass ein solcher Algorithmus immer die gleichen Berechnungen ausführt und somit nicht lernen kann. Wie wir aber sehen werden, können selbst einfache Algorithmen ihre Fähigkeiten durch Lernprozesse erweitern. Solche Systeme agieren zudem anders als herkömmliche Software, weil sie nicht nur starr binär arbeiten – also entweder „ja“ oder „nein“ kennen –, sondern Entscheidungen auch anhand von Wahrscheinlichkeiten treffen.

### 3.2.3 Ein Lernproblem: Die Erkennung von Ziffern

Wir möchten nun die Konstruktion und das Training eines Modells anhand einer einfachen Lernaufgabe darstellen. Hierbei geht es um die Erkennung von handgeschriebenen Ziffern. Jede Ziffer wird durch ein Grauwertbild aus  $28 \times 28$  Pixeln wiedergegeben. Wie in Abb. 3.7 dargestellt ist der Grauwert eines Pixels eine Zahl zwischen 0,0 (weiß) und 1,0 (schwarz). Die Größe der Ziffern wurde normalisiert und in der Bildmitte zentriert. Es handelt sich hier um den MNIST Datensatz mit handgeschriebenen Ziffern, der seit 1998 als

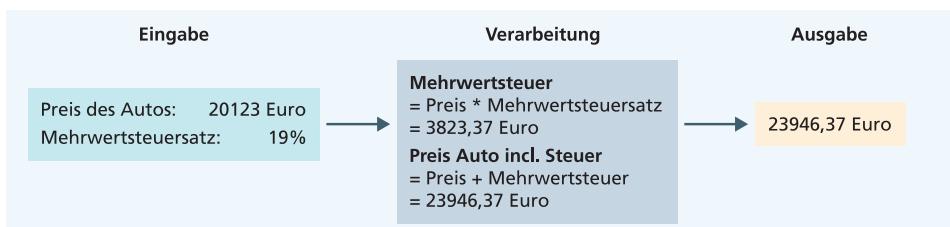
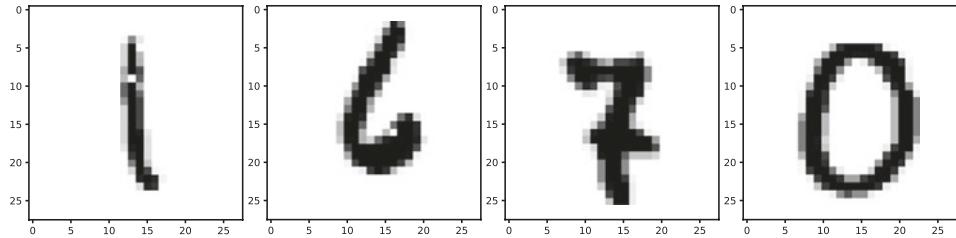


Abb. 3.6 Beispiel eines einfachen Algorithmus zur Berechnung der Mehrwertsteuer



**Abb. 3.7** Beispieldaten aus dem MNIST-Datensatz von Fotos von handgeschriebenen Ziffern. Ziel ist die Zuordnung jedes Bildes zu einer der Ziffern-Klassen.

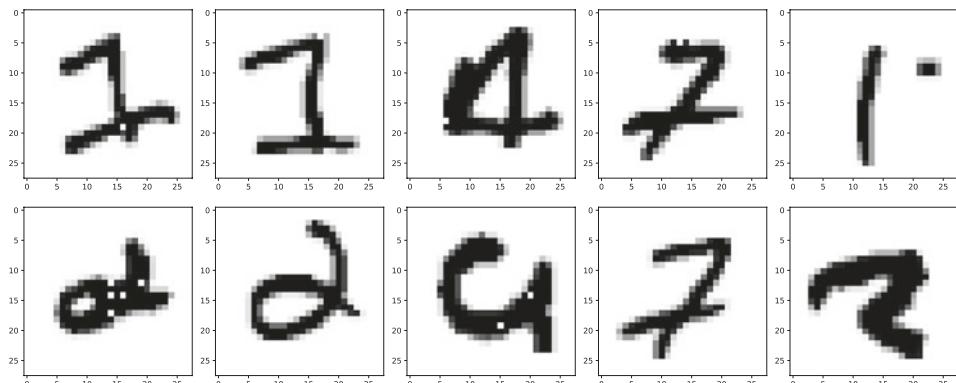
Benchmark-Datensatz für viele Lernverfahren genutzt wird (LeCun et al. 1998). Wir haben also hier ein Klassifikationsproblem mit 10 verschiedenen Klassennamen: „null“, „eins“, bis „neun“. Dies verdeutlicht, dass man Klassen durch beliebige Namen bezeichnen kann.

Abb. 3.7 enthält für vier der zehn Ziffern Beispieldaten aus dem MNIST-Datensatz. Abb. 3.8 zeigt einige besonders schwierig zu kategorisierende Ziffern. Es wird deutlich, dass eine Reihe von Verwechlungsmöglichkeiten existieren, z. B. zwischen „eins“ und „sieben“.

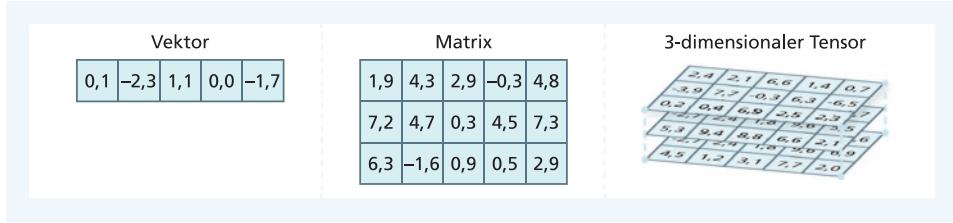
### 3.2.4 Vektoren, Matrizen und Tensoren

Abb. 3.10 zeigt eine handgeschriebene „zwei“ und die zugehörigen Grauwert-Zahlenwerte eines Pixelbildes. Ein solches rechteckiges Zahlenpaket bezeichnet man auch als *Matrix* (Mehrzahl: *Matrizen*). Eine  $28 \times 28$  Matrix hat dabei 28 Zeilen und 28 Spalten. Jede Zeile einer Matrix besteht aus einer einzelnen Reihe von Zahlen der Länge 28. Eine solche Reihe von Zahlen bezeichnet man auch als *Vektor*.

Die Zahlen in der Matrix liegen im Bereich 0.0 (weiß) bis 1.0 (schwarz), und dazwischenliegende Zahlen entsprechen einem Grauwert. Man könnte auch ein farbiges Bild



**Abb. 3.8** Beispiele im MNIST Datenbestand, bei denen die Ziffernbilder nur schwer zu klassifizieren sind.

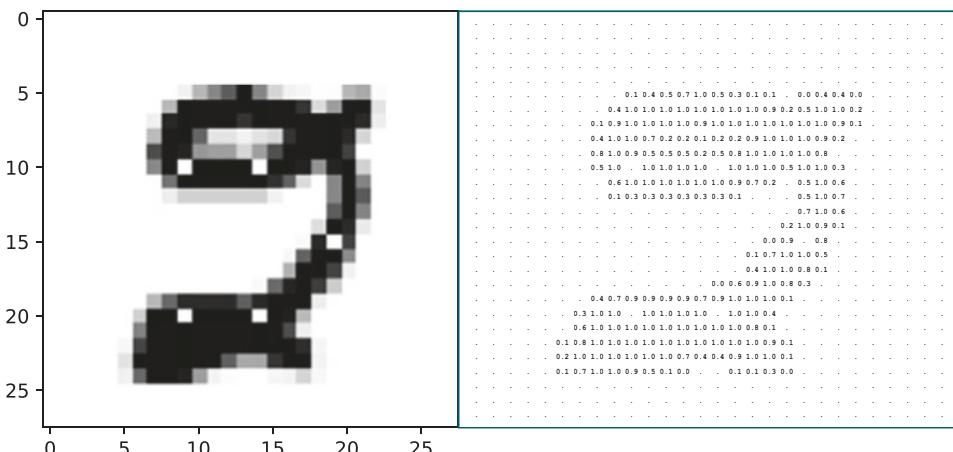


**Abb. 3.9** Vektor, Matrix und Tensor mit beliebigen reellen Zahlen. Die Eingaben für tiefe neuronale Netze werden meist in dieser Form kodiert. Man kann auch die Dimension zu der Bezeichnung hinzufügen. Im Bild haben wir einen 5-Vektor, eine  $3 \times 5$ -Matrix und einen  $3 \times 3 \times 5$ -Tensor.

einer Ziffer betrachten. Dabei kann man für jedes Pixel die Farbe durch drei Zahlen festlegen: die Anteile von rot, grün und blau. Damit lässt sich ein farbiges Bild durch drei Matrizen darstellen, die Rotwert-, Grünwert- und Blauwert-Matrix. Stapelt man diese drei Matrizen übereinander, so erhält man ein quaderförmiges dreidimensionales Zahlenpaket, einen dreidimensionalen „Tensor“.

In der ursprünglichen Bedeutung beschreibt ein Tensor spezielle Größen in der Physik (Tensor 2019). Heutzutage ist ein Tensor eine Sammelbezeichnung von Skalaren, Vektoren, Matrizen und höherdimensionalen Zahlenpaketen (Abb. 3.9). Somit ist ein Vektor ein eindimensionaler Tensor. Ein zweidimensionaler Tensor ist eine Matrix mit Zeilen und Spalten (also zwei Dimensionen). Eine einzelne Zahl kann man als einen Vektor der Länge 1 darstellen. Die Bezeichnung Tensor ist offenbar unabhängig von der Anzahl der vorliegenden Dimensionen.

Die Eingaben für ein Modell sind in den allermeisten Fällen ein oder mehrere Tensoren. Im Falle der Ziffernerkennung ist die Eingabe die  $28 \times 28$ -Matrix von Grauwerten (Abb. 3.10). Die Ausgabe ist wiederum ein beliebiger Tensor, bei einer Klassifikation eine einzige ganze Zahl. Dieser Klassenindex wird den Namen der möglichen Klassen zugeordnet.



**Abb. 3.10** Grauwertbild einer handgeschriebenen „zwei“ (links) und zugehörige Zahlenwerte in Form einer  $28 \times 28$ -Matrix. Die Zahlenwerte liegen zwischen 0.0 (weiß) und 1.0 (schwarz). Der Zahlenwert „0.0“ ist als „.“ dargestellt.

### 3.3 Lernen eines Zusammenhangs

#### 3.3.1 Schema für das Lernen: Modell, Verlustfunktion und Optimierung

Nahezu alle maschinellen Lernverfahren sollen den Zusammenhang zwischen den Werten einer Eingabe  $x$  und den zugehörigen Werten der Ausgabe  $y$  in den Trainingsdaten reproduzieren können. Bevor wir einen konkreten Lernalgorithmus kennenlernen, soll der allgemeine Ablauf des Lernens skizziert werden. Dieses Vorgehen wird bei den meisten Verfahren des maschinellen Lernens, d. h. also auch bei den TNN, verwendet. Dies wird fast immer mit den folgenden drei Schritten durchgeführt (Abb. 3.11):

- **Prognose:** Das Modell prognostiziert (berechnet) eine Ausgabe  $\hat{y}$  aus der Trainingseingabe  $x$ . Die Modelleigenschaften werden gesteuert durch einen Parameter  $w$ , einen Vektor, dessen Werte zu Beginn zufällig gewählt werden.
- **Verlust:** Der Prognosewert  $\hat{y}$  wird mit dem Ausgabewert  $y$  in den Trainingsdaten verglichen und daraus ein Verlust, eine reelle Zahl berechnet, die den Unterschied zwischen beiden Werten misst. Der Verlust sollte möglichst klein sein.
- **Optimierung:** Die Optimierung ändert den Parameter  $w$  so ab, dass der Verlust kleiner wird. Dies ist möglich, weil die Prognose eines Modells von dem Wert des Parameters  $w$  abhängt. Meist erfolgt dies in Form vieler kleiner Änderungen des Parametervektors  $w$ . Das Verfahren endet, wenn keine Verbesserung mehr erfolgt.

Die Anpassung des Parametervektors  $w$  wird im Allgemeinen sehr oft wiederholt, bis ein Modell mit kleinem Verlust gefunden wurde. Diesen Vorgang nennt man *Training*.

Dies ist das grundlegende Schema, mit dem man ein Modell bestimmen kann, welches aus den Eingaben  $x$  die Ausgabe  $y$  in den Trainingsdaten mit geringem Fehler prognostizieren kann. Es hat für TNN die gleiche Bedeutung wie der Antrieb der Räder durch Motor, Getriebe, Kupplung und Achse, welcher das Fahren bei einem Auto ermöglicht. Daher ist es wesentlich, die Logik dieses Vorgehens zu verstehen. Im nächsten Abschnitt wird im Einzelnen dargestellt, wie dieses Schema auf das Problem der Ziffernklassifikation angewendet werden kann.

#### 3.3.2 Detaillierter Ablauf des Lernens

Ausgangspunkt sind die Trainingsdaten, deren  $i$ -tes Element  $(x,y)^{(i)}$  aus einer Eingabe  $x$  (Pixelmatrix) und der zugehörigen Ausgabe  $y$  (Ziffernklaasse) besteht. Die Prognose des

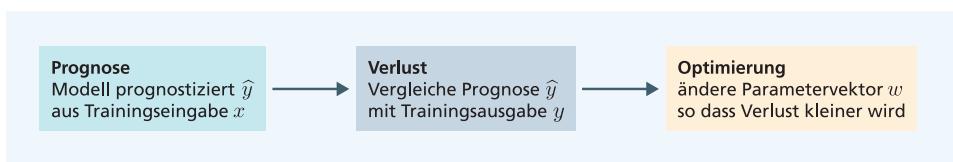
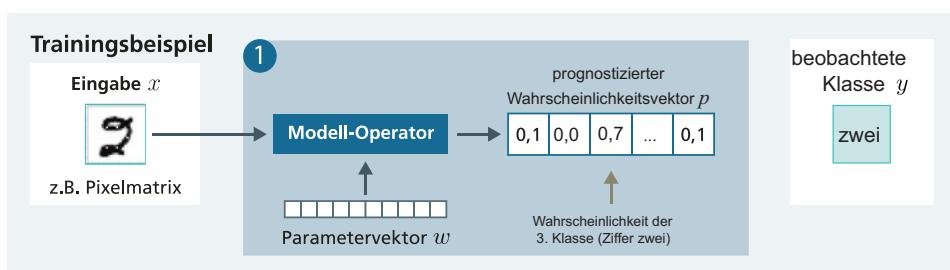


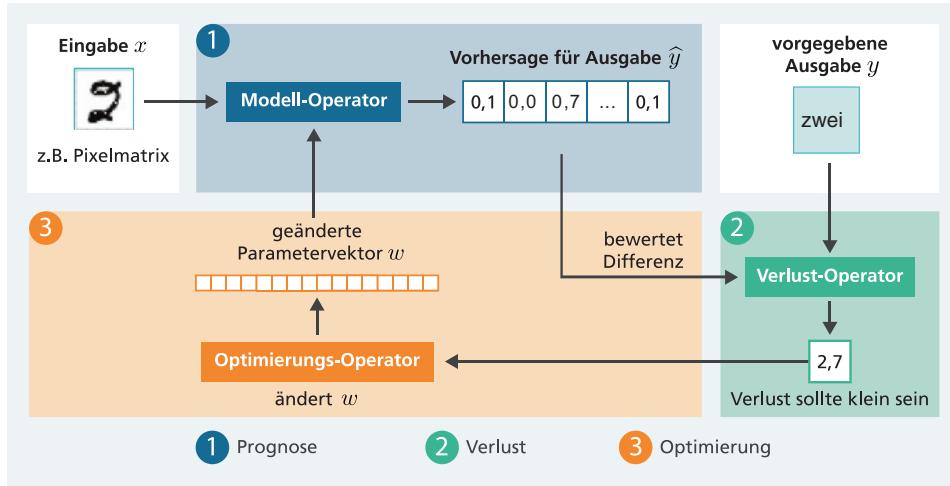
Abb. 3.11 Grundlegende Schritte für das Training von maschinellen Lernverfahren.

Modelloperators wird durch einen mit reellen Zahlen gefüllten Parametervektor  $w$  beeinflusst. Beim Lernen der Klassifikation können wir nun folgendermaßen vorgehen:

- **Prognose:** Der Modelloperator erhält als Eingabe  $x$  eine Pixelmatrix und einen mit reellen Zahlen gefüllten Parametervektor  $w$ . Er berechnet nicht für  $x$  direkt eine Vorhersage  $\hat{y}$  für die zugehörige Klasse  $y$ , sondern geht einen Umweg. Er prognostiziert für  $x$  und  $w$  zunächst einen Wahrscheinlichkeitsvektor  $p$  mit den Wahrscheinlichkeiten aller möglichen Klassen (Ziffern) (Abb. 3.12). Ein Wahrscheinlichkeitsvektor enthält positive Zahlen zwischen 0.0 und 1.0, deren Summe gleich 1.0 ist. Jeder Klasse wird eine Position (Klassenindex) in dem Wahrscheinlichkeitsvektor  $p$  zugeordnet, also der Klasse „null“ die erste Position, der Klasse „eins“ die zweite Position, usw. Der trainierte Modelloperator sollte nun für die beobachtete Klasse  $y$  an der zugehörigen Position eine hohe Wahrscheinlichkeit prognostizieren. Da sich alle Wahrscheinlichkeiten zu 1.0 aufsummieren, müssen die restlichen Wahrscheinlichkeiten klein sein. Ist also in der Pixelmatrix zu  $x$  eine „zwei“ abgebildet, so sollte der berechnete Wahrscheinlichkeitswert an der entsprechenden Position des Ausgabevektors hoch sein, also an der 3. Position.
- **Verlust:** Um die Güte der Vorhersage zu bestimmen, wird der Verlustoperator eingesetzt (2 in Abb. 3.13), der die Verlustfunktion berechnet. Sie bewertet den Unterschied zwischen dem vorhergesagten Wahrscheinlichkeitsvektor  $p$  mit den Wahrscheinlichkeiten der einzelnen Ziffern und der beobachteten Klasse  $y$  (z.B. „zwei“) mit einem Verlustwert, einer reellen Zahl. Der Verlust sollte möglichst klein sein (d. h. die Wahrscheinlichkeit der beobachteten Ziffer möglichst hoch). Dieses Vorgehen prognostiziert die Klasse also nicht direkt, sondern die Wahrscheinlichkeit der beobachteten Klasse  $y$ . Daher berechnet das Modell einen kontinuierliche Wert, der sich stufenlos steigern lässt. Dies ist entscheidend, wenn man die Prognose in kleinen Schritten verbessern will, z.B. durch Optimierungsverfahren.



**Abb. 3.12** Die Prognose berechnet mit dem Modelloperator für ein Trainingsbeispiel  $(x,y)^{(i)}$  aus der Eingabe  $x$  einen Wahrscheinlichkeitsvektor  $p$  mit den Wahrscheinlichkeiten der verschiedenen Klassen (Ziffern). Ziel ist es, für die beobachtete Klasse  $y$  (z.B. „zwei“) eine möglichst hohe Wahrscheinlichkeit zur prognostizieren. Die Prognose des Modells hängt von einem Parametervektor  $w$  ab.



**Abb. 3.13** Der Verlustoperator ② bewertet für ein Trainingsbeispiel  $(x,y)^{(i)}$  den Unterschied zwischen dem Wahrscheinlichkeitsvektor  $p$  mit den Wahrscheinlichkeiten der Klassen und der beobachteten Klasse  $y$  („zwei“). Der Optimierungsoperator ③ analysiert den Verlust und modifiziert den Parametervektor  $w$ , sodass sich der Verlust verringert (d.h. die Wahrscheinlichkeit der beobachteten Ziffer größer wird).

- **Optimierung:** Ein Optimierungsverfahren analysiert die Verlustfunktion und das Modell. Wenn man den Parametervektor  $w$  ändert, so ändert sich auch der vorhergesagte Wahrscheinlichkeitsvektor  $p$ . Daher kann man durch Änderung der Parameter die Wahrscheinlichkeit der beobachteten Klasse (Ziffer) erhöhen. Das Optimierungsverfahren bestimmt, wie man die Werte des Parametervektors  $w$  ändert muss, damit sich der Verlust des Modells verringert (③ in Abb. 3.13). Anschließend wird eine meist geringfügige Änderung von  $w$  nach diesem Schema durchgeführt.

Der Verlust wird bei einem solchen *Optimierungsschritt* meist nur unwesentlich verkleinert. Daher muss man die Schrittfolge ①②③ sehr oft wiederholen, um tatsächlich einen geringen Verlust zu erreichen. Die Optimierung stoppt, wenn sich keine Verbesserungen mehr ergeben. Die einzelnen Berechnungsschritte können durch Operatoren (Abschn. 1.5) mit passenden Ein- und Ausgabetsoren realisiert werden.

Durch den Parametervektor hat das Modell die Möglichkeit, ein reichhaltiges Repertoire an Verhaltensweisen darzustellen. Die Idee hinter dem maschinellen Lernen ist, die Detailinformationen über die Anwendung aus den Daten zu beziehen. Anstatt ein ausgefeiltes Modell manuell zu konstruieren, spezifiziert man ein Gerüst eines Modells, also ein Modell mit noch nicht festgelegten Parameterwerten. Der Parametervektor umfasst typischerweise mehrere Millionen bis hin zu Milliarden Parameterwerte. Damit ist er in der Lage, sehr komplexe Zusammenhänge zwischen den Eingabe- und Ausgabetsoren zu erfassen. Mit passenden und ausreichenden Daten kann man dann das oben beschriebene Optimierungsverfahren ausführen, um die Parameter des Modells zu adaptieren und das Verhalten des Modells an die Daten anzupassen.

Konzeptionell ermöglicht uns dieses Vorgehen, die Komplexität der realen Welt nicht allein durch die Modellstruktur, sondern durch die Parameter zu repräsentieren, welche automatisch aus den Daten bestimmt werden können. Der fast magische Hauptteil des Lernens besteht darin, dass – wenn es richtig gemacht wird – das trainierte Modell gute Vorhersagen auch für Daten ermöglicht, welche **nicht** zum Trainieren benutzt wurden. Dies wird als *Generalisierung* bezeichnet und steht explizit oder implizit im Mittelpunkt eines jeden Algorithmus für maschinelles Lernen.

Es wird nun ein sehr einfaches Modell und die zugehörige Verlustfunktion in allen Einzelheiten formuliert.

### 3.4 Ein simples Modell: Die logistische Regression

Der Einfachheit halber ignorieren wir zunächst die zweidimensionale Anordnung der Pixel im Grauwertbild der Ziffern. Wir bilden aus der Pixelmatrix einen Vektor, indem wir alle Zeilen der Pixelmatrix hintereinanderhängen. Dieser Vektor wird  $x$  genannt. Er hat die Länge  $784 = 28 \times 28$  und dient als Eingabe für unser Lernverfahren.

Wir benötigen jetzt ein Modell, das aus unserem Eingabevektor  $x$  die gewünschte Ausgabe berechnet, d. h. die zur Pixelmatrix gehörige Ziffer. Es gibt in diesem Fall genau  $k = 10$  mögliche Ausgabewerte. Ein solches Modell wird auch als Klassifikator bezeichnet und die  $k$  verschiedenen möglichen Ausgaben „null“, ..., „neun“ als Klassen.

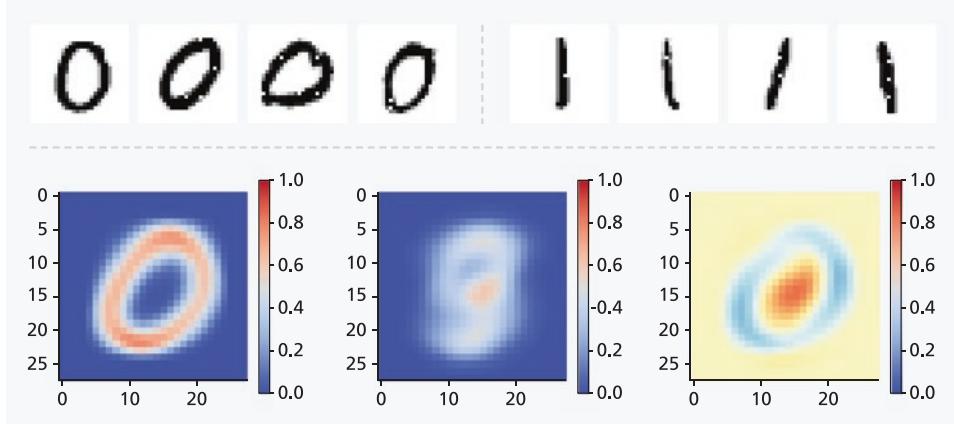
Als konkreten Algorithmus wählen wir hier eine einfache Möglichkeit: Für jede der  $k$  möglichen Klassen berechnen wir einen Punktewert  $u_i$ . Je höher dieser Punktewert  $u_i$  ist, desto plausibler ist die zugehörige  $i$ -te Klasse.

#### 3.4.1 Berechnung eines Punktwertes

In unserem ersten simplen Modell gehen wir davon aus, dass jedes Pixel einen **separaten** Einfluss auf den Punktewert  $u_i$  der  $i$ -ten Klasse hat. Um eine Intuition hierfür zu erhalten, betrachten wir Abb. 3.14. Hier sind einige Trainingsbeispiele der Klassen „null“ und „eins“ aufgeführt. Im unteren Teil sind die mittleren Grauwerte für die Bilder mit einer „null“ und für die Bilder mit einer Ziffer ungleich „null“. Man sieht deutlich, dass die schwarzen Pixel der „null“ meist in einem kreisförmigen Bereich liegen (unten links) und die schwarzen Pixel der anderen Klassen verteilt in den mittleren Bereich (unten Mitte). Jedes Pixel soll als unabhängiger Indikator für eine Klasse verwendet werden.

Man sieht, dass hohe Werte in dem kreisförmigen Bereich und niedrige Werte im Zentrum auf eine „null“ hindeuten. Die Pixel in dem äußeren Bereich haben kaum Aussagekraft. Diese Pixelgewichtungen soll das Modell automatisch finden.

Ausgehend von diesen Überlegungen möchten wir jedem Pixel  $j$  einen Gewichtswert  $w_{ij}$  für jede der Klassen  $i = 1, \dots, 10$  zuordnen. Zu Beginn wissen wir nichts über die Gewichtswerte und ordnen ihnen zufällig ausgewählte Startwerte zu. Um hieraus für ein Trainingsbeispiel  $(x,y)$  einen Punktewert z. B.  $u_1$  für die erste Klasse „null“ zu erhalten,



**Abb. 3.14** Die obersten beiden Reihen enthalten einige Trainingsbeispiele der Klassen „null“ und „eins“ aus MNIST. Die untere Zeile enthält die Mittelwerte der Pixelwerte für die Klasse „null“ (unten links) und die der anderen Klassen (unten Mitte). Rechts unten findet sich die Differenz der Mittelwerte. Hohe Pixelwerte im blauen Bereich der Differenz und niedrige Pixelwerte im roten Bereich deuten auf eine „null“ hin.

multiplizieren wir den Grauwert  $x_j$  des  $j$ -ten Pixels aus dem Bereich [0.0, 1.0] in unserem Eingabevektor  $x$  mit dem zugehörigen Gewicht  $w_{1,j}$ . Diese Berechnung führen wir für jeden der 784 Grauwerte des Eingabevektors durch und addieren das Ergebnis. Schließlich wird noch ein weiterer Wert  $b_1$  hinzugefügt, der von keinem Pixel abhängt. Dieser bewirkt unabhängig von den Eingaben eine globale Erhöhung des Punktwertes  $u_1$  für die erste Klasse und kann die Prognose des Modells verbessern. Für den Punktwert  $u_1$  der 1. Klasse „null“ ergibt sich damit:

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

Punktwert  
der 1. Klasse
Grauwert  
des 1. Pixels
Grauwert  
des 2. Pixels
  
Gewicht  
des 1. Pixels
Gewicht  
des 2. Pixels
Globale Erhöhung des  
Punktwertes der 1. Klasse

Das Ergebnis dieser Berechnung ist eine einzige reelle Zahl  $u_1$ , der Punktwert der 1. Klasse.

Ist  $y = i$  die beobachtete Klasse der Eingabe  $x$ , so sollte:

- der Punktwert  $u_i$  dieser Klasse hoch sein
- alle anderen Punktwerte möglichst klein sein.

Das Gewicht  $w_{1,j}$  ist der Einfluss des  $j$ -ten Pixels auf den Punktwert der 1. Klasse. Diesen Einfluss kennen wir zu Beginn nicht und wir möchten ihn während des Lernens bestimmen. Daher wird die Zahl  $w_{1,j}$  während des Lernens verändert, um die Klassifikation zu verbessern: sie ist ein **Parameter**. Aus der Diskussion zu Abb. 3.14 wird deutlich, dass für die Klasse „null“ ein Pixel  $x_i$  in dem ringförmigen Bereich einen hohen positiven  $w_{1,i}$ -Wert erhalten sollte, während einem Pixel  $x_j$  im Zentrum ein hoher negativer Wert  $w_{1,j}$ -Wert zugewiesen werden sollte. Insgesamt bilden die Zahlen  $(w_{1,1}, w_{1,2}, w_{1,3}, \dots, w_{1,784})$  einen Vektor, der mit  $w_1$  bezeichnet wird. Die oben aufgeführte Berechnung von  $u_1$  lässt sich dann alternativ in Vektorschreibweise notieren:

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

Skalarprodukt

Punktwert der 1. Klasse

Gewicht des 2. Pixels

Grauwert des 2. Pixels

Globale Erhöhung des Punktwertes

Diese Art der Multiplikation der beiden Vektoren  $w_1$  und  $x$  wird auch als *Skalarprodukt* bezeichnet, weil das Ergebnis eine einzelne Zahl (Skalar) ist.

### 3.4.2 Die gleichzeitige Berechnung aller Punktwerte

Diese Berechnung kann für jede der 10 möglichen Klassen („null“, ..., „neun“) durchgeführt werden. Die Gewichte  $w_{i,j}$  des  $j$ -ten Pixels für die  $i$ -te Klasse sind natürlich jeweils anders. Daher erhält man die separate Berechnung von 10 unterschiedlichen Punktwerten  $u_1, \dots, u_{10}$  für jede Klasse.

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

Punktwerte jeder Klasse

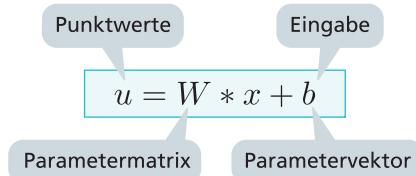
Eingabe-Pixel

Nun wird deutlich, dass die Zahlen  $w_{i,j}$  ein rechteckiges Zahlenpaket bilden, also als eine Matrix  $W$  mit 10 Zeilen und 784 Spalten behandelt werden können. Ebenfalls lassen sich die  $u_i$ ,  $b_i$  und  $x_i$  zu Vektoren der Längen 10 bzw. 784 zusammenfassen:

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

### 3.4.3 Lineare Transformation

Damit kann man die oben vorgenommene Berechnung der Punktwerte  $u_1, \dots, u_{10}$  in folgender Matrix-Notation schreiben:



Es handelt sich hier um die Multiplikation des Vektors  $x$  mit der Matrix  $W$ . Zu dem Ergebnis wird anschließend der Vektor  $b$  addiert und ergibt den Vektor  $u$ . Ist  $i$  der Klassenindex der beobachteten Klasse der Eingabe  $x$ , so sollte der so berechnete Punktwert  $u_i$  dieser Klasse hoch sein und die anderen Punktwerte niedriger.

Die Parametermatrix  $W$  enthält alle Einflussparameter der einzelnen Pixel auf den Punktwert der jeweiligen Ziffern, insgesamt  $10 * 784 = 7840$  Parameter. Der Parametervektor  $b$  wird *Bias* genannt. Er enthält die globalen Erhöhungen/Erniedrigungen für den Punktwert, also zusätzlich 10 Parameter. Damit stehen 7850 Stellwerte zur Verfügung, mit denen die Punktwerte für die einzelnen Eingaben angepasst werden können.

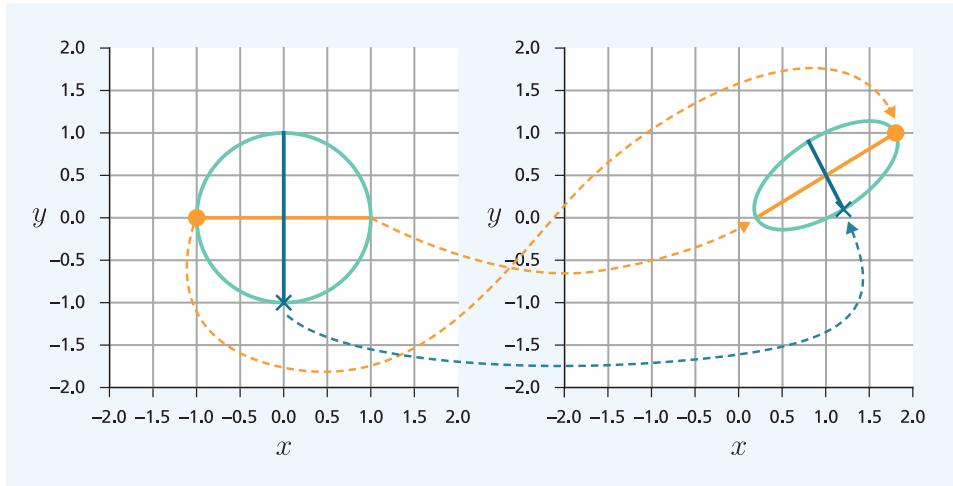
Der Einfachheit halber werden beide Parameter oft zu einem *globalen Parametervektor*  $w$  zusammengefasst, in dem die Werte von  $W$  und  $b$  hintereinander stehen

$$w = (W, b)$$

Die Multiplikation eines Vektors mit einer Matrix und die nachfolgende Addition eines weiteren Vektors ist der am häufigsten genutzte Bestandteil tiefer neuronaler Netze. Sie wird auch *lineare Transformation* genannt, aus Gründen, die wir im nächsten Absatz erklären werden. Auf den heutigen Computern gibt es optimierte Rechenbefehle, mit deren Hilfe sie sehr schnell ausgeführt werden kann.

Warum diese Transformation lineare Transformation genannt wird, lässt sich leicht verdeutlichen. In Abb. 3.15 wurden links zwei Linien und ein Kreis eingezeichnet. Diese wurden mit der linearen Transformation

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



**Abb. 3.15** Zwei Linien und ein Kreis im 2-dimensionalen Raum (links). Jeder Punkt  $x = (x_1, x_2)$  auf den Linien und dem Kreis wird durch die Abbildung  $u = W * x + b$  transformiert. Die transformierten Vektoren  $u$  wurden auf dem rechten Bild eingezeichnet. Entscheiden ist, dass eine gerade Linie immer in eine gerade Linie überführt wird. Dies gilt für beliebig hohe Dimensionen. Aus diesem Grund wird diese Abbildung auch lineare Transformation genannt. Offenbar wird ein Kreis in eine Ellipse transformiert

transformiert. Die transformierten Punkte sieht man in der rechten Grafik von Abb. 3.15. Der Vektor  $b$  sorgt für eine Verschiebung nach rechts oben und die Matrix  $W$  bewirkt eine Drehung und Streckung. Wichtig ist, dass eine lineare Transformation alle Punkte auf einer geraden Linie immer in Punkte überführt, die wieder auf einer geraden Linie liegen. Dies wird mit der roten und der blauen Linie verdeutlicht.

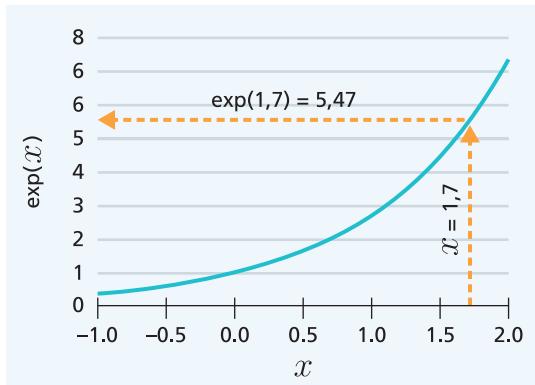
Daher röhrt die Bezeichnung „lineare Transformation“. Im Wesentlichen bewirkt die lineare Transformation eine Verschiebung, eine Drehung sowie eine Stauchung oder Dehnung. Wie man an der Überführung des Kreises in eine Ellipse sieht, ist das Ausmaß der Stauchung/Dehnung von der Richtung abhängig. Diese Aussagen gelten auch für lineare Transformationen mit beliebig großen Matrizen und entsprechen hochdimensionalen Ein- oder Ausgabevektoren. Dabei kann sich die Länge der Eingaben von der Länge der Ausgaben unterscheiden.

### 3.4.4 Die Softmaxfunktion erzeugt einen Wahrscheinlichkeitsvektor

Ist  $y$  die beobachtete Klasse der Eingabe  $x$ , so sollte der Punktwert dieser Klasse hoch und die Punktwerter der anderen Klassen sollten niedriger sein. Um einen Lernalgorithmus zu erhalten, müssen wir aber festlegen, wieviel höher der Punktwert der korrekten Klasse sein sollte. Leider sind Punktwerter inhaltlich nur schwer interpretierbar. Daher wird der Vektor  $u$  der Punktwerter in einen Vektor von Wahrscheinlichkeiten transformiert, mit dem man einen plausiblen Lernalgorithmus festlegen kann.

Ein Wahrscheinlichkeitsvektor besteht aus positiven reellen Zahlen zwischen 0,0 und 1,0. Zudem muss die Summe dieser Zahlen 1,0 sein. Man kann nun einen beliebigen Vek-

**Abb. 3.16** Die Exponentialfunktion steigt immer weiter an und hat nur positive Zahlen als Ergebnis



tor mit positiven und negativen Zahlen, z. B.  $u = (1,5, -1,2, 0,5)$ , in drei Schritten in einen solchen Wahrscheinlichkeitsvektor umwandeln. Dies wird in Abb. 3.17 demonstriert.

Im ersten Schritt wendet man auf jede Komponente von  $u$  die Exponentialfunktion an, die in Abb. 3.16 gezeigt wird, und erhält einen Vektor  $v$  aus positiven Zahlen. Im zweiten Schritt wird die Summe der Zahlen in  $v$  berechnet. In einem dritten Schritt dividiert man die Komponenten von Vektor  $v$  durch diese Summe und erhält einen Wahrscheinlichkeitsvektor  $p$ .

Dieser kleine Algorithmus wird durch die *Softmaxfunktion* ausgeführt:

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

Das Ergebnis  $p = (p_1, p_2, \dots, p_k)$  ist immer ein Wahrscheinlichkeitsvektor.

### 3.4.5 Das logistische Regressionsmodell

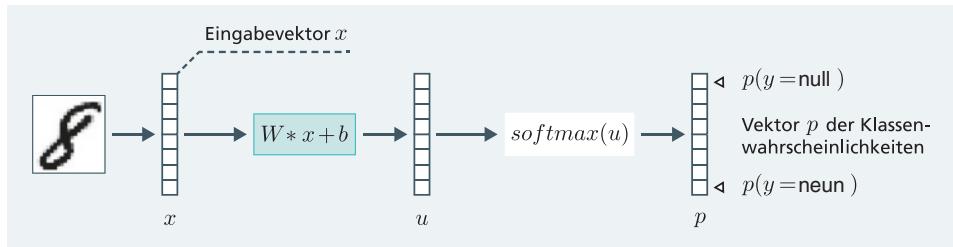
Wir sind nun in der Lage, ein einfaches Klassifikationsmodell für die Erkennung der Ziffern (Abschn. 3.2.3) anzugeben. Führt man die lineare Transformation  $u = Wx + b$  und die Softmaxfunktion  $p = \text{softmax}(u)$  nacheinander aus, so erhalten wir ein Modell, welches den Eingabevektor  $x$  in einen Vektor  $p$  von Klassenwahrscheinlichkeiten überführt. Dies

**Abb. 3.17** Ein beliebiger Vektor  $u$  wird durch die Softmaxfunktion in einen Wahrscheinlichkeitsvektor  $p$  transformiert

Ausgangsvektor  $u = (1,5, -1,2, 0,5)$ .

1. Anwendung der Exponentialfunktion (Abbildung 54) auf jedes Element von  $u$ :  
 $v = \exp(u) = (4,48, 0,30, 1,65)$ .  
 $v$  enthält nur positive Zahlen.
  2. Berechnung der Summe der Elemente in  $v$ :  
 $4,48 + 0,30 + 1,65 = 6,43$
  3. Division der Elemente von  $v$  durch die Summe.
- $$p = \frac{(4,48, 0,30, 1,65)}{6,43} = (0,70, 0,05, 0,25)$$

Das Ergebnis  $p$  ist ein Wahrscheinlichkeitsvektor.



**Abb. 3.18** Komponenten des logistischen Regressionsmodells. Aus den Eingaben  $x$  wird über die lineare Transformation und die Softmaxfunktion ein Vektor  $p$  von Klassenwahrscheinlichkeiten berechnet. Die Matrix  $W$  und der Vektor  $b$  sind Parameter, die an die Trainingsdaten angepasst werden

wird in Abb. 3.18 gezeigt. Die vorhergesagte plausibelste Klasse  $\hat{y}$  erhält man, indem man die Klasse mit der höchsten Wahrscheinlichkeit auswählt. Das Modell wird logistisches Regressionsmodell genannt. Laut einer Umfrage von Kaggle ist das logistische Regressionsmodells das am häufigsten angewandte Verfahren des maschinellen Lernens.

Damit haben wir jetzt ein **Modell** (Abschn. 3.3.1), welches für einen Eingabevektor  $x$  (kodiert das Bild, z. B. ) und die gegebenen Parameter  $w = (W, b)$  den Wahrscheinlichkeitsvektor der einzelnen Klassen (Ziffern) ausrechnet:

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

Der beobachteten Ausgabe  $y$  (z.B. „acht“) ist der Klassenindex (laufende Nummer) der Klasse zugeordnet. Der Einfachheit halber verwendet man statt des Klassennamens den Klassenindex, wenn ein numerischer Wert benötigt wird. Hat das Modell nun z. B. den Vektor  $p(x, w) = (0,02, 0,0, 0,08, 0,09, 0,22, 0,01, 0,0, 0,0, 0,57, 0,0)$  als Ergebnis berechnet, so ist  $p_{\hat{y}}(x, w) = 0,57$  die höchste Klassenwahrscheinlichkeit, wobei  $y$  hier der Index 9 der Klasse „acht“ ist. Dabei ist  $p_y(x, w)$  der  $y$ -te Wert des Wahrscheinlichkeitsvektors.

## 3.5 Die Güte des Modells

### 3.5.1 Maßstab für die Modellgüte: Die Wahrscheinlichkeit der kompletten Trainingsdaten

Die Wahrscheinlichkeit  $p_y(x, w)$  der „richtigen“ Klasse  $y$  wird auch nach der Optimierung nie exakt gleich 1.0 sein. Die Frage ist daher, wie man die Güte eines Wahrscheinlichkeitsvektors  $p(x, w)$  für einen Eingabevektor  $x$  bewerten sollte. Eine einfache Möglichkeit wäre, die Differenz zwischen  $p_y(x, w)$  und 1.0 als Maßstab zu wählen. Es hat sich aber gezeigt, dass ein solches Vorgehen die Elemente des Wahrscheinlichkeitsvektors falsch gewichtet. Solche Probleme werden bei einem alternativen Bewertungsmaßstab vermieden. Hierzu betrachten wir die Menge aller 60.000 Trainingsbeispiele in der Trainingsmenge

$$\text{Train} = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(60000)}, y^{(60000})\}$$

Dabei ist  $x^{(m)}$  der Eingabevektor (Pixelwerte) und  $y^{(m)}$  der Index (laufende Nummer) der zugehörigen Klasse für das  $m$ -te Trainingsbeispiel. Wir können mithilfe des Modells für die gegebenen Parameter  $w = (W, b)$  den Wahrscheinlichkeitsvektor  $p(x^{(m)}, w) = \text{softmax}(Wx^{(m)} + b)$  für jedes Trainingsbeispiel  $x^{(m)}$  ausrechnen. Ist  $y^{(m)}$  die zugehörige korrekte Klasse in der Trainingsmenge, so ist die Komponente  $p_{y^{(m)}}(x^{(m)}, w)$  die **Wahrscheinlichkeit der korrekten Klasse  $y^{(m)}$**  des  $m$ -ten Trainingsbeispiels  $(x^{(m)}, y^{(m)})$ .

Um die besten Parameter  $w = (W, b)$  zu finden, kann man aus der statistischen Theorie die Anforderung ableiten, dass die Wahrscheinlichkeit der gesamten Trainingsmenge  $\text{Train}$  möglichst hoch sein soll. Diese Wahrscheinlichkeit ist das Produkt der Wahrscheinlichkeiten der beobachteten Klassen der Elemente der Trainingsmenge. Das Produkt kann verwendet werden, wenn die Beobachtung der einzelnen Beispiele unabhängig voneinander stattfindet.

$$P(\text{Train}|w) = p_{y^{(1)}}(x^{(1)}, w) * \dots * p_{y^{(60000)}}(x^{(60000)}, w)$$

Die Parameter  $w = (W, b)$  kann man nun so ändern, dass die Wahrscheinlichkeit der **gesamten Trainingsmenge** möglichst groß wird. Dies macht die beobachteten Klassen der Trainingsbeispiele für die Trainingsdaten im Mittel besonders wahrscheinlich. Falls man dann die Berechnung für neue Eingabedaten wiederholt, so kann man davon ausgehen, dass die prognostizierten Wahrscheinlichkeiten für die tatsächlichen Klassen ebenfalls hoch sein werden. Die Wahrscheinlichkeit der gesamten Trainingsmenge wird auch als **Likelihood** bezeichnet. Die Suche nach dem Parameter mit der größten Likelihood ist das **Maximum-Likelihood-Prinzip**. Dies ist eine fundamentale Richtschnur zur Bestimmung von Modellparametern in der modernen Statistik (Efron und Hastie 2016, S. 38).

Die Parameter  $w = (W, b)$  sollten so geändert werden, dass die Wahrscheinlichkeit der **gesamten Trainingsmenge** möglichst groß wird.

### 3.5.2 Wie misst man den Lernerfolg: Die Verlustfunktion

Berechnet man für Wahrscheinlichkeit der gesamten Trainingsmenge das Produkt von 60.000 kleinen Zahlen (z. B. 0.1), so wird das Ergebnis sehr klein (z. B.  $0.1^{60000}$ , d. h. eine

Zahl mit 60.000 Nullen nach dem Komma). Eine solche Zahl ist auf einem Rechner nur sehr schwierig handhabbar. Daher transformiert man die Berechnungsformel mit der Logarithmusfunktion  $\log(\cdot)$ . Diese Funktion erhält die „>“-Relation, d. h. ist  $a > b$  so ist auch  $\log(a) > \log(b)$ . Daher kann man auch die transformierte Berechnungsformel optimieren und erhält das gleiche Optimum wie bei der Ausgangsformel. Die transformierte Berechnung hat folgende Form:

$$\log P(Train | w) = \log(p_{y^{(1)}}(x^{(1)}, w)) + \dots + \log(p_{y^{(60000)}}(x^{(60000)}, w))$$

Dabei wurde das Produkt der Wahrscheinlichkeiten durch eine Summe ersetzt, weil  $\log(a * b) = \log(a) + \log(b)$ . Es hat sich zudem eingebürgert, dass anstelle einer Maximierung eine Minimierung vorgenommen wird. Dies erfordert einfach noch eine Multiplikation mit  $-1$ . Damit haben wir die **Verlustfunktion**  $L(w)$ , welche die Abweichung zwischen den vorhergesagten Klassen (Klassenwahrscheinlichkeiten  $p$ ) und den beobachteten Klassen  $y$  für die komplette Trainingsmenge und die gegebene Parameter  $w = (W, b)$  bewertet (Abschn. 3.3.1):

$$L(w) = -\log P(Train | W, b) \quad (3.1)$$

Diese Verlustfunktion wird *Log-Likelihood Verlustfunktion* oder auch *Maximum-Entropie Verlustfunktion* genannt. Diese Verlustfunktion wird von der großen Mehrzahl der Klassifikationsmodelle verwendet. Abb. 3.19 beschreibt der Vollständigkeit halber die *Kleinste-Quadrat-Verlustfunktion* für Modelle mit kontinuierlichen Ausgaben, welche sich auch aus dem Maximum-Likelihood-Prinzip ergibt.

Ist die Prognose des Modells  $\hat{y} = f(x; w)$  eine reelle Zahl, so wird oft die Normalverteilung mit Erwartungswert  $f(x^{(i)}; w)$  und einer festen Varianz  $\sigma^2$  gewählt, um die Verteilung der möglichen Ausgabewerte um den Erwartungswert zu beschreiben:

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

Berechnet man die Wahrscheinlichkeit der gesamten Trainingsmenge aus  $n$  Elementen und wendet man den negativen Logarithmus an, so erhält man als Verlustfunktion das Kriterium des kleinsten quadratischen Fehlers:

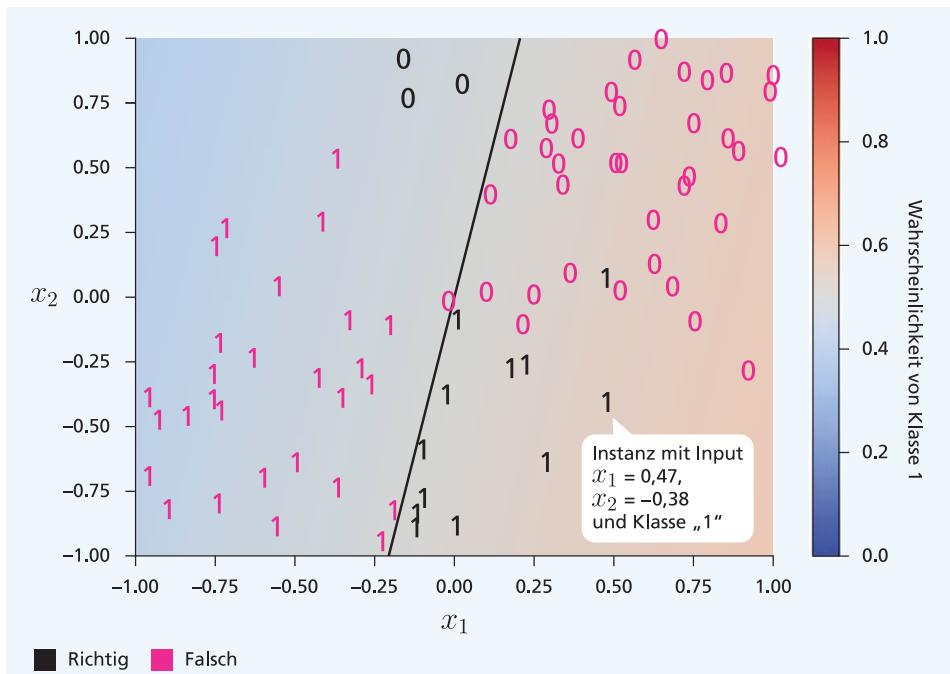
$$L_{normal}(w) = (y^{(1)} - f(x^{(1)}; w))^2 + \dots + (y^{(n)} - f(x^{(n)}; w))^2$$

**Abb. 3.19** Die Kleinste-Quadrat-Verlustfunktion für die Prognose kontinuierlicher Werte

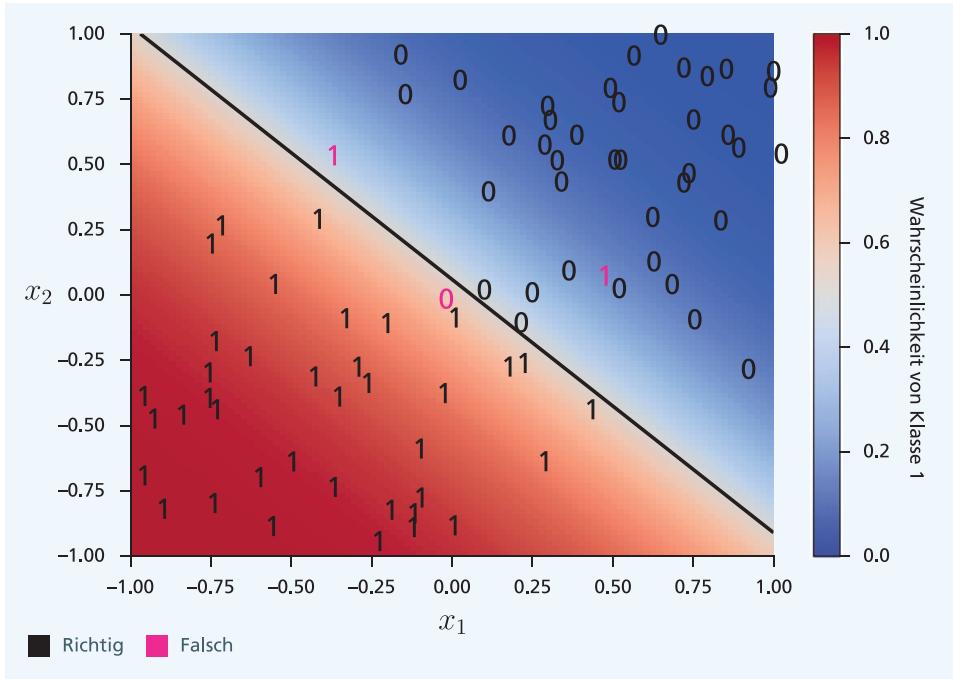
### 3.5.3 Verdeutlichung für zwei Klassen und zwei Eingabemerkmale

Man kann das Klassifikationsproblem stark vereinfachen, indem man nur noch zwei Klassen (0 und 1) betrachtet und nur noch zwei Eingabemerkmale  $x_1$  und  $x_2$  verwendet. In Abb. 3.20 sind die Trainingsbeispiele der Trainingsmenge durch ihre Eingabemerkmale und die Klasse dargestellt: die Zeichen 0 und 1 symbolisieren die Elemente der beiden Klassen.

Zu Beginn wird der Parametervektor  $w$  zufällig gewählt. In Abb. 3.20 ist der entsprechende Klassifikator (logistische Regression) durch einen roten und einen blauen Bereich gekennzeichnet: in dem roten Bereich ist die Wahrscheinlichkeit für eine 1 größer als 0,5, im blauen Bereich ist die Wahrscheinlichkeit der 0 größer als 0,5. Die Trennlinie zwischen beiden Bereichen ist eine gerade Linie. Allerdings liegt eine Anzahl der beobachteten Trainingsbeispiele auf der „falschen“ Seite der Linie. Diese fehlklassifizierten Trainingsbeispiele sind magentafarben markiert. Für diese Trainingsbeispiele ist der in  $P(\text{Train}, w)$  eingehende Wahrscheinlichkeitswert kleiner als 0,5 und der Wert von  $P(\text{Train}, w)$  ist entsprechend niedrig.



**Abb. 3.20** Ein Klassifikationsmodell trennt hier die Eingabeebene in zwei Bereiche. Im rötlichen Bereich ist die Wahrscheinlichkeit der Klasse 1 größer als 0,5 und alle Trainingsbeispiele werden als 1 klassifiziert. Im blauen Bereich ist die Wahrscheinlichkeit der Klasse 0 größer als 0,5 und alle Trainingsbeispiele werden als 0 klassifiziert. Viele Trainingsbeispiele sind falsch klassifiziert (magenta), d. h. die Wahrscheinlichkeit der korrekten Klasse ist kleiner als 0,5



**Abb. 3.21** Das Klassifikationsmodell trennt hier in den roten Bereich, in dem alle Trainingsbeispiele als 1 klassifiziert werden, und den blauen Bereich, in dem alle Trainingsbeispiele als 0 klassifiziert werden. Es gibt insgesamt nur 3 falsch klassifizierte Trainingsbeispiele (magenta). Für alle anderen Trainingsbeispiele muss die Wahrscheinlichkeit der korrekten Klasse größer als 0,5 sein

Abb. 3.21 zeigt die Situation bei gleichen Eingabedaten für ein logistisches Regressionsmodell mit einem anderen Parameter  $w^*$ . Hier ist die Trennlinie des Modells sehr gut an die Klassen angepasst. Es gibt nur insgesamt drei Trainingsbeispiele, die auf der „falschen“ Seite liegen: ein 0-Trainingsbeispiel und zwei 1-Trainingsbeispiele. Für alle anderen Trainingsbeispiele ist der in  $P(\text{Train}, w^*)$  eingehende Wahrscheinlichkeitswert größer als 0,5 und der Wert von  $P(\text{Train}, w^*)$  ist entsprechend hoch. Tatsächlich ist in Abb. 3.21 der beste logistische Regressionsklassifikator dargestellt.

Wie man in Abb. 3.21, sieht kann man nicht garantieren, dass jedes Trainingsbeispiel in die korrekte Klasse eingeordnet wird. Es kann geschehen, dass die Wahrscheinlichkeit der korrekten Klasse eines einzelnen Trainingsbeispiel absinkt, weil man durch eine Veränderung von  $w$  die Wahrscheinlichkeiten mehrerer anderer Trainingsbeispiele verbessern kann.

### 3.6 Optimierung, oder wie findet man die besten Parameterwerte

Wir müssen jetzt einen Optimierungsalgorithmus finden, der die 7850 Parameter des Parametervektors  $w$  des Modells so modifiziert, dass die Verlustfunktion  $L(w)$  möglichst klein wird.



**Abb. 3.22** Die Minimierung der Verlustfunktion gleicht dem Abstieg von einem Berg im Nebel. Bildnachweis im Anhang

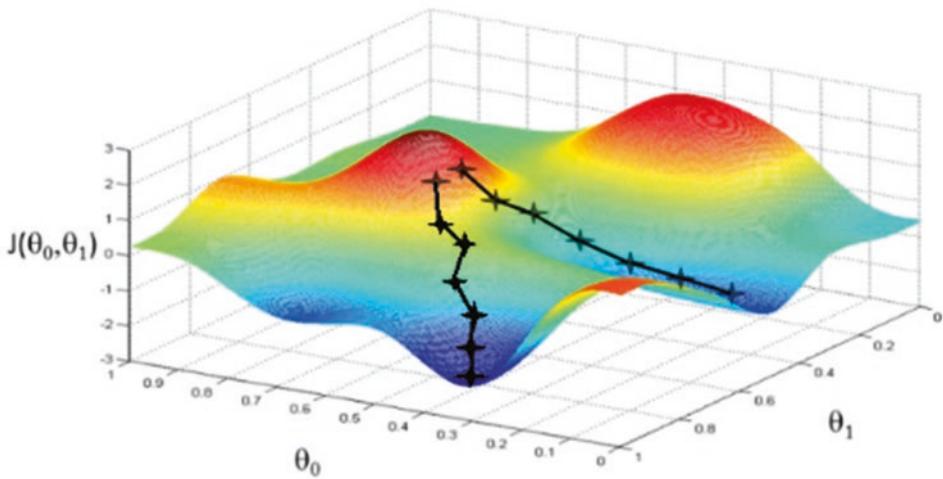
Die Minimierung der Verlustfunktion ist vergleichbar mit der Situation eines Bergsteigers, der im Nebel möglichst schnell zum niedrigsten Punkt des Tals absteigen möchte (Abb. 3.22). Er weiß nicht, wo das Tal liegt, und kann nur seine nächste Umgebung überblicken. Als einziges kann er in seiner unmittelbaren Umgebung erkennen, wo es bergab geht. Daher wählt er immer die Richtung des steilsten Abstiegs an dem Punkt, wo er sich gerade befindet (hierbei nehmen wir an, dass keine Absturzgefahr besteht). Dabei kann es durchaus passieren, dass er in einer Senke landet, von der nach diesem Vorgehen kein weiterer Abstieg möglich ist.

### 3.6.1 Der Gradient zeigt in Richtung des steilsten Anstiegs

Bei unserem Optimierungsproblem kann man die Verlustfunktion  $L(w)$  ebenfalls als eine Berglandschaft interpretieren, allerdings in einem hochdimensionalen Raum, da  $w$  insgesamt 7850 Komponenten hat. In der Mathematik bezeichnet **Gradient** einen Vektor, der in die Richtung des steilsten Anstiegs einer Funktion zeigt. Der steilste Abstieg ist dann in Richtung des mit  $-1.0$  multiplizierten Gradienten. In Abb. 3.23 ist eine solche Funktionslandschaft gezeichnet. Ziel ist es vom Berg aus (rot) den niedrigsten Punkt zu erreichen. Die schwarzen Linien zeigen den Gradienten in der jeweiligen Position. Das Verfahren folgt dem negativen Gradienten für ein kleines Stück. Anschließend muss der Gradient neu bestimmt werden, da sich die Richtung des steilsten Anstiegs geändert hat. Je nach Ausgangspunkt kann diese Gradientenabstiegsoptimierung in unterschiedlichen Senken landen.

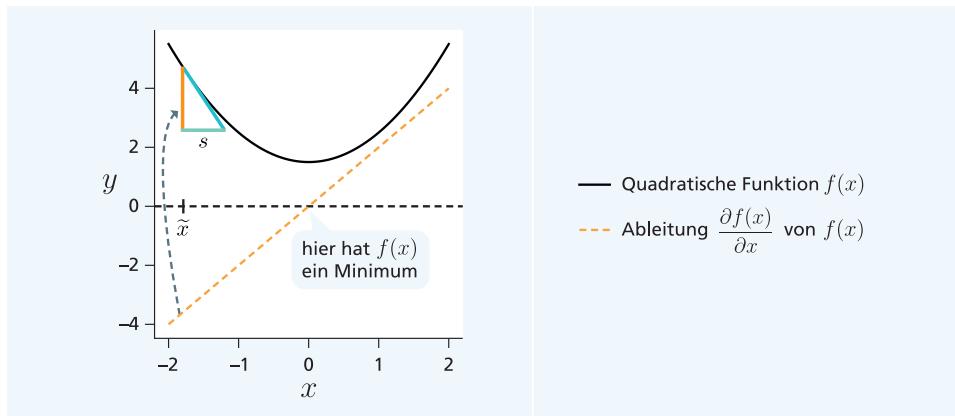
### 3.6.2 Der Gradient für mehrere Dimensionen

Aber wie kann man nun den Gradienten für Funktionen in mehreren Dimensionen bestimmen? Wir möchten zunächst an den Schulunterricht erinnern, in dem die Ableitung einer



**Abb. 3.23** Abstieg in einer Landschaft, die von einer Funktion definiert wird. Die schwarzen Linien zeigen die Richtung des steilsten Abstiegs im jeweiligen Punkt.

Funktion besprochen wurde. In Abb. 3.24 ist eine eindimensionale Funktion und deren Ableitung eingezeichnet. Die Ableitung  $\frac{\partial f(x)}{\partial x}(\tilde{x})$  der Funktion, z. B. im Punkt  $\tilde{x} = -1,8$ , ist die Steigung der Tangente des in der Grafik eingezeichneten Dreiecks, wenn man eine kleine Strecke  $s$  nach rechts geht. Die Tangente ist eine Linie, welche die Kurve im Punkt  $\tilde{x}$  berührt. In der Grafik ist die Strecke  $s$  die untere Seite eines Dreiecks (grün). Die Steigung



**Abb. 3.24** Quadratische Funktion  $f(x)$  und deren Ableitung  $\frac{\partial f(x)}{\partial x}$ . Die Ableitung in einem Punkt  $\tilde{x} = -1,8$  ist der Wert  $s * \frac{\partial f(x)}{\partial x}(\tilde{x})$ , um den die Funktion ansteigt, wenn man um eine kleine Strecke  $s$  nach rechts geht. Die Ableitung in  $\tilde{x}$  (rot) ist negativ, also sinkt die Funktion.

ist hier negativ und bildet die senkrechte Seite des Dreiecks (rot). Die diagonale Seite des Dreiecks (blau) ist gerade die Tangente der Kurve im Punkt  $\tilde{x}$ . Man sieht, dass die Ableitung für den Wert  $\tilde{x}$  negativ ist, die Funktion also nach rechts kleiner wird. Ändert man die Position  $\tilde{x}$ , so ändert sich auch die Ableitung. Im Punkt  $\tilde{x} = 0,0$  hat die Funktion eine waagrechte Tangente und die Ableitung 0. Hier liegt ein Minimum vor. Auf der rechten Seite der Grafik steigt die Funktion an und die Ableitung ist positiv.

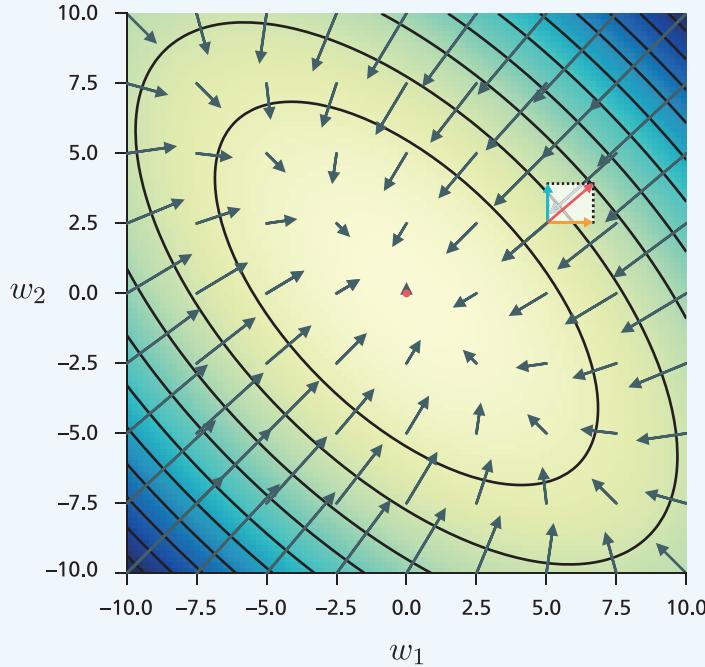
Wenn man eine Funktion  $L(w)$  hat, welche einen Vektor  $w = (w_1, \dots, w_k)$  als Argument besitzt, so hängt diese Funktion von mehreren Variablen ab, den Komponenten  $w_1, \dots, w_k$  des Vektors. Für jede Komponente  $w_i$  des Vektors kann man eine Ableitung in einem Punkt  $\tilde{w}$  berechnen, wobei man die anderen Komponenten nicht verändert. Das ist die Tangente  $\frac{\partial L(w)}{\partial w_i}(\tilde{w})$  der Funktion  $L(w)$  in der  $i$ -ten Dimension der  $k$ -dimensionalen Darstellung von  $L(w)$ . Diese Ableitung  $\frac{\partial L(w)}{\partial w_i}(\tilde{w})$  wird auch *partielle Ableitung* genannt, weil jeweils nur eine der möglichen Variablen verändert wird. Dabei bezeichnet  $\partial L(w)$  die Funktion, nach der abgeleitet wird,  $\partial w_i$  die Variable, nach der abgeleitet wird, und das Argument  $(\tilde{w})$  den Wert des Vektors, für den die partielle Ableitung berechnet wird.

In Abb. 3.25 ist zur Verdeutlichung die Funktion  $L(w)$  mit dem Vektor-Argument  $w = (w_1, w_2)$  abgebildet. Die Funktion hat in dem roten Punkt ihr Minimum und steigt nach allen Seiten hin an. Die niedrigen Werte sind durch einen gelben Farnton gekennzeichnet, die hohe Werte durch einen blauen Farnton. Ebenfalls eingezeichnet sind „Höhenlinien“, d. h. Linien gleichen Funktionswertes. Für einen Beispelpunkt  $\tilde{w} = (5, 0, 2, 5)$  ist die partielle Ableitung  $\frac{\partial L(w)}{\partial w_1}(\tilde{w})$  als orangefarbener Pfeil eingezeichnet und die partielle Ableitung  $\frac{\partial L(w)}{\partial w_2}(\tilde{w})$  als cyanfarbener Pfeil. Fasst man diese beiden Pfeile zu einem Vektor zusammen, so erhält man den Gradienten  $\frac{\partial L(\tilde{w})}{\partial w} = \left( \frac{\partial L(w)}{\partial w_1}(\tilde{w}), \frac{\partial L(w)}{\partial w_2}(\tilde{w}) \right)$ , einen Vektor in die Richtung des steilsten Anstieges der Funktion im Punkt  $\tilde{w}$ . Er ist durch einen roten Pfeil dargestellt. Der Gradient steht immer senkrecht auf den Höhenlinien.

Für die Minimierung benötigen wir aber die Richtung des steilsten Abstiegs. Das ist der dem Gradienten entgegenstehende dunkelgrüne Vektor, auch negativer Gradient genannt. Die negativen Gradienten an anderen Punkten wurden auf die gleiche Weise berechnet. Man sieht, dass man mithilfe des negativen Gradienten von allen Positionen aus immer in Richtung des Minimums gelangt. Allerdings weist der negative Gradient nicht exakt auf das Minimum hin, sondern nach einem gewissen Abstand kann sich die Richtung zum Minimum ändern. Im Minimum (roter Punkt) ist der negative Gradient 0, d. h. auch beide partiellen Ableitungen sind 0.

### 3.6.3 Der Gradient der Verlustfunktion

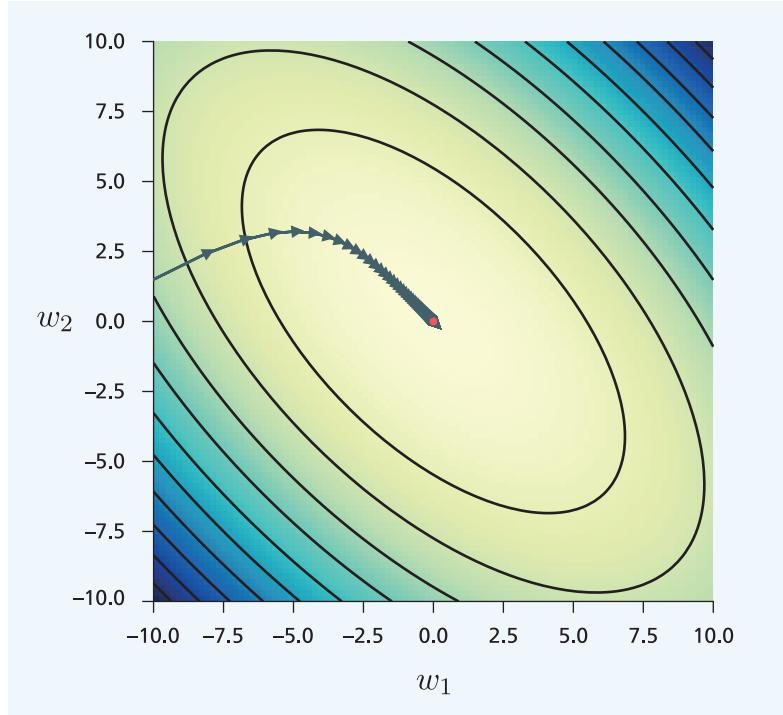
Für die Änderung der Verlustfunktion bezüglich  $w$  muss man also folgende Ableitung berechnen:



**Abb. 3.25** Darstellung einer Funktion  $L(w)$  von zwei Variablen  $w = (w_1, w_2)$ . Kleine Werte sind gelb und große blau hinterlegt, und das Minimum ist am roten Punkt. Der Gradient (roter Pfeil) ist die Vektorsumme der partiellen Ableitungen von  $L(w)$  nach  $w_1$  (orangefarbener Pfeil) und  $w_2$  (grüner Pfeil). Dieser Vektor zeigt in die Richtung des steilsten Anstiegs. Der negative Gradient ist der dem roten Pfeil entgegengesetzte Vektor (dunkelgrüner Pfeil), die Richtung des stellsten Abstiegs. Alle anderen negativen Gradienten wurden analog berechnet.

$$\frac{\partial -\log P(\text{Train} \mid w)}{\partial w} = \frac{\partial \left[ -\log(p_{y^{(1)}}(x^{(1)}, w)) - \dots - \log(p_{y^{(60000)}}(x^{(60000)}, w)) \right]}{\partial w}$$

Die Ableitungsformeln muss man heute aber nicht mehr manuell bestimmen. Vielmehr sind viele Programmierumgebungen für neuronale Netze in der Lage, die Formeln für die Ableitungen selbsttätig symbolisch auszurechnen und zu verwenden. Wenn man das Modell und die Verlustfunktion festgelegt hat, so ist also die eigentliche Arbeit schon getan. Man muss lediglich noch das Optimierungsverfahren aus einer Liste verfügbarer Methoden auswählen und der Lernalgorithmus kann auf den Trainingsdaten ablaufen. Trotzdem sollte man verstehen, nach welchen Prinzipien das Optimierungsverfahren abläuft.



**Abb. 3.26** Minimierung der Verlustfunktion  $L(w)$  durch Gradientenabstieg in Richtung des steilsten Abstiegs der Verlustfunktion. Das Verfahren beginnt mit einem zufällig gewählten Startwert  $w^{(0)}$ .

Der Parameter  $w^{(i)}$  wird immer in Richtung des steilsten Abstiegs  $-\frac{\partial L(w)}{\partial w}(w^{(i)})$  geändert. Die Schrittweite wird durch die Lernrate  $\lambda$  bestimmt

### 3.6.4 Schrittweise Minimierung durch Gradientenabstieg

Die Minimierung der Verlustfunktion lässt sich nun – wie in Abb. 3.26 für 2 Dimensionen dargestellt – in folgenden Schritten durchführen. Dabei werden sukzessive die Parametervektoren  $w^{(0)}, w^{(1)}, w^{(2)}, \dots$  erzeugt, bis das Verfahren stoppt.

1. Der Parametervektor  $w^{(0)}$  in Schritt 0 wird mit zufällig gezogenen Werten gefüllt.
2. Ist  $w^{(i)}$  der aktuelle Parametervektor in Schritt  $i$ , so wird der negative Gradient der Verlustfunktion  $-\frac{\partial L(w)}{\partial w}(w^{(i)})$  für den Parametervektor  $w^{(i)}$  ausgerechnet. Dieser negative Gradient ist selbst ein Vektor und zeigt in die Richtung des steilsten Abstiegs der Verlustfunktion in der Position  $w^{(i)}$ .
3. Der nächste Parametervektor  $w^{(i+1)}$  wird nun berechnet durch

The diagram illustrates the update rule for the parameter vector in gradient descent. A central box contains the equation  $w^{(i+1)} = w^{(i)} - \lambda * \frac{\partial L(w)}{\partial w}(w^{(i)})$ . Three arrows point from labels below to parts of the equation: 'neuer Parametervektor' points to  $w^{(i+1)}$ , '(-1)\* Lernrate' points to  $\lambda$ , and 'Gradient der Verlustfunktion' points to  $\frac{\partial L(w)}{\partial w}(w^{(i)})$ . Above the equation, a callout bubble says 'vorheriger Parametervektor' pointing to  $w^{(i)}$ .

Hierbei ist  $\lambda$  eine kleine Zahl, die *Lernrate*. Damit wird der aktuelle Parametervektor  $w^{(i)}$  ein kleines Stück in die Richtung des steilsten Abstiegs verändert und der Verlust wird etwas kleiner.

4. Ist die Differenz zwischen  $L(w^{(i+1)})$  und  $L(w^{(i)})$  sehr klein, so stoppt das Verfahren und  $w^{(i+1)}$  wird als Parameter mit minimalem Verlust ausgegeben.  
Ansonsten macht das Verfahren weiter mit Schritt 2. Weil der negative Gradient von  $w^{(i+1)}$  ein anderer ist als der von  $w^{(i)}$ , so muss er neu berechnet werden.

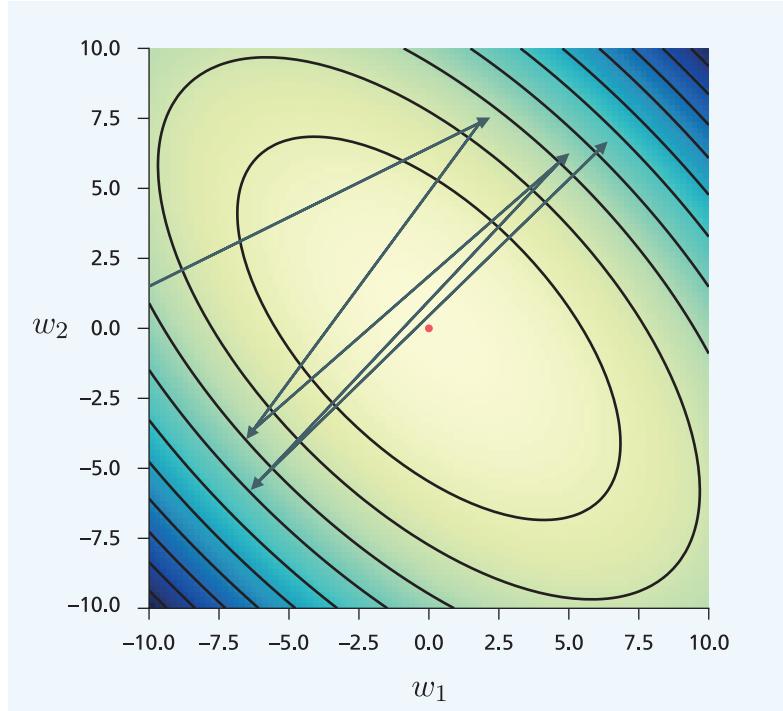
### 3.6.5 Die Lernrate bestimmt die Länge eines Optimierungsschritts

Die Lernrate  $\lambda$  ist eine kleine positive Zahl, die festgelegt werden muss. Sie bestimmt, wie schnell das Lernverfahren das Optimum erreicht. Ist  $\lambda$  sehr klein, so braucht das Verfahren sehr viele Schritte und muss viele Gradienten ausrechnen, um das Minimum zu erreichen. Ist  $\lambda$  zu groß, so kann es sein, dass der nächste Schritt über den minimalen Bereich hinwegführt, das Verfahren ins Schwingen gerät, und der minimale Wert nicht erreicht wird (Abb. 3.27). Die Lernrate ist eine Stellgröße des Optimierungsverfahrens, deren Wert festgelegt werden muss. Sie kann aber nicht einfach wie ein Modellparameter optimiert werden. Es gibt eine Reihe solcher Stellgrößen, die auch *Hyperparameter* genannt werden. Die Bestimmung ihrer Werte wird ab Abschn. 4.8.1 diskutiert.

### 3.6.6 Gradientenabstieg mit Minibatch benötigt viel weniger Rechenaufwand

Um den Gradienten zu bestimmen, muss die Verlustfunktion ausgewertet werden. Um die Verlustfunktion für alle 60.000 Trainingsbeispiele  $(x^{(m)}, y^{(m)})$  der Trainingsmenge zu berechnen, muss man die Modellausgabe  $\text{softmax}(W * x^{(m)} + b)$  für alle diese Trainingsbeispiele bestimmen. Da während des Trainings viele Gradientenschritte durchgeführt werden müssen, führt dies zu einem hohen Rechenaufwand.

Als Alternative kann man die Trainingsmenge in kleine Teilmengen von  $m$  Trainingsbeispielen (z. B.  $m = 30$ ) zerlegen. Diese Teilmengen heißen *Minibatches*. Anschließend wird der Gradient für einen Minibatch berechnet und ein Gradientenschritt durchgeführt. Hierzu muss die Modellausgabe  $\text{softmax}(W * x^{(m)} + b)$  nur für die Trainingsbeispiele des Minibatches berechnet werden, der Rechenaufwand sinkt also dramatisch. Der nächste Gradientenschritt



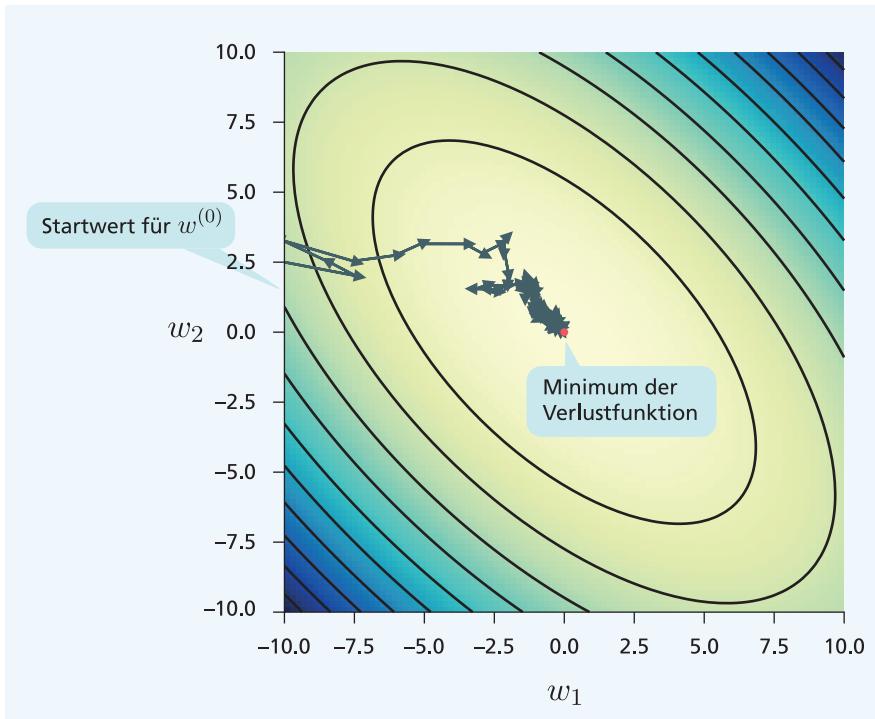
**Abb. 3.27** Minimierung der Verlustfunktion  $L(w)$  durch Gradientenabstieg. Ist die Lernrate  $\lambda$  zu groß, so schwingt der berechnete Parametervektor um das Minimum und konvergiert nicht

wird dann mit einem anderen Minibatch berechnet usw. Sind alle Minibatches verwendet worden, so hat man eine Epoche erreicht, d. h. alle Trainingsbeispiele der Trainingsmenge wurden einmal verwendet. Danach geht das Verfahren erneut durch die Trainingsmenge.

In Abb. 3.28 wird eine Abfolge von Trainingsschritten mit Minibatches für unser Beispiel angezeigt. Es wird deutlich, dass die Trainingsschritte stark von Zufallseinflüssen überlagert sind, da der Minibatch-Gradient stark vom „korrekten“ Gradienten mit allen Trainingsdaten abweichen kann. Trotzdem ist der Minibatch-Gradient im Mittel korrekt und führt meist auch zum Minimum.

Die Berechnung des Minibatch-Gradienten hat drei Vorteile:

- Die Berechnung des Gradienten erfordert um Größenordnungen weniger Rechenaufwand.
- Das Verfahren erreicht die Nähe des minimalen Wertes insgesamt viel schneller.
- Der Gradient weicht zufällig von dem tatsächlichen Gradienten ab, da ja nur eine kleine Stichprobe der Trainingsbeispiele zu seiner Berechnung verwendet wurde. Der Gradient wird also durch Zufallseinflüsse überlagert und ist aber im Mittel korrekt. Dadurch ist das Verfahren oft in der Lage, aus lokalen Senken herauszuspringen, d. h. lokale Minima zu verlassen.



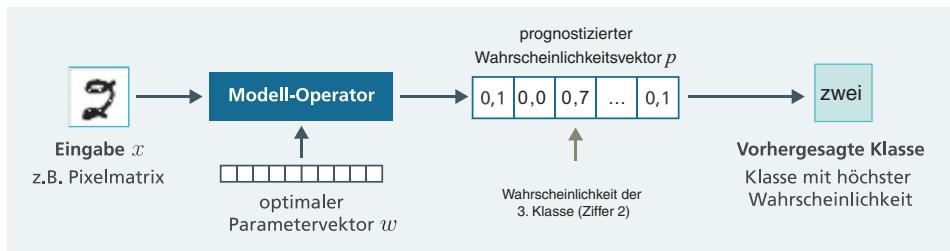
**Abb. 3.28** Wird bei einem stochastischen Gradientenabstieg der negative Gradient nur für einen Minibatch berechnet, so weicht er zufällig vom „korrekten“ negativen Gradienten für sämtliche Trainingsdaten ab. Dadurch wird der Minimierungspfad durch starke Zufallsschwankungen überlagert. Trotzdem kommt das stochastische Gradientenabstiegsverfahren in die Nähe des Optimums

Aus diesen Gründen wird heute fast überall der *Minibatch-Gradientenabstieg* verwendet. Dieses Verfahren hat das Training mit Millionen von Trainingsdaten überhaupt erst möglich gemacht. Wegen der zufälligen Variationen des Gradienten spricht man auch vom *stochastischen Gradientenabstieg*.

## 3.7 Überprüfung des Lernerfolges

### 3.7.1 Anwendung des Modells auf neue Daten

Wenn der Parametervektor  $w$  des Modells durch die Optimierung so angepasst wurde, dass die prognostizierte Wahrscheinlichkeit der beobachteten Klassen  $y$  für die Trainingsdaten möglichst groß wurde, so kann man das Modell auf neue Eingaben anwenden. Dies wird in Abb. 3.29 dargestellt. Hierbei wird nur eine Prognose mit der Eingabe durchgeführt. Als Ausgabe wird die Klasse mit der höchsten Klassenwahrscheinlichkeit gewählt.



**Abb. 3.29** Anwendung des logistischen Regressionsmodells auf neue Daten. Hierbei wird der durch das Training optimierte Parametervektor verwendet und eine Prognose ausgeführt, welche den Wahrscheinlichkeitsvektor für die Klassen berechnet. Die Klasse mit der höchsten Wahrscheinlichkeit wird als Ausgabe gewählt

### 3.7.2 Überprüfung der Genauigkeit auf der Testmenge

Um eine Abschätzung der Qualität des Modells zu erhalten, kann man die Genauigkeit der Klassifikation berechnen. Diese *Genauigkeit* ist die Anzahl der korrekt klassifizierten Beispieldaten durch die Anzahl aller klassifizierten Beispieldaten.

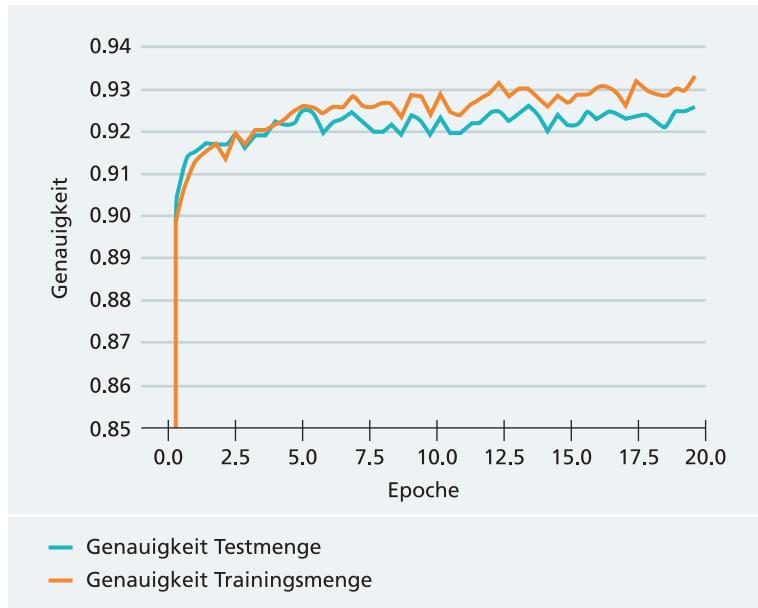
Ziel ist es, eine Genauigkeit zu ermitteln, die auch auf neue Daten übertragbar ist. Das Modell soll also neue Eingaben korrekt zuordnen, die während des Trainings nicht verfügbar waren. Diese Fähigkeit nennt man *Generalisierung* und ist das eigentliche Erfolgskriterium eines maschinellen Lernverfahrens. Also darf man die Genauigkeit nicht auf den Trainingsdaten bestimmen, da ja dort das Modell so optimiert wurde, dass schon möglichst viele Trainingsbeispiele korrekt klassifiziert werden.

Daher muss man die Genauigkeit auf Daten ausrechnen, die **nicht** zum Training verwendet wurden. Es ist nun allgemeine Praxis, die verfügbaren annotierten Daten zufällig in zwei Teilmengen zu zerlegen (Abb. 3.30): die Trainingsmenge und die Testmenge. Oft wählt man die Testmenge so, dass sie etwa 20 % der Datenbeispiele umfasst. Dann ist einerseits der Genauigkeitsverlust beim Training nicht sehr groß, andererseits ist der Schätzwert für die Genauigkeit ausreichend präzise. Die Elemente der Trainingsmenge heißen Trainingsbeispiele und die Elemente der Testmenge Testbeispiele.

Abb. 3.31 zeigt die Genauigkeit auf der Trainings- und Testmenge der MNIST-Daten (Abschn. 3.2.3) nach jeder Epoche der Optimierung. Zu Beginn steigt die Genauigkeit schnell an und wächst oszillierend immer höher bis auf etwa 0,924, d. h. von 100 Ziffern werden etwas mehr als 7 falsch klassifiziert. Die Oszillationen werden durch den Minibatch-



**Abb. 3.30** Zerlegung der annotierten Daten in Trainingsdaten und Testdaten



**Abb. 3.31** Genauigkeit auf der Trainings- und Testmenge für die jeweiligen Epochen der stochastischen Optimierung für das logistische Regressionsmodell und die MNIST-Daten

Gradientenabstieg verursacht, der zu zufälligen Abweichungen in den Gradienten führt. Die Genauigkeit auf der Testmenge ist wie erwartet deutlich geringer als auf der Trainingsmenge. Das Verfahren benötigt unter 10 Sekunden für die Berechnungen auf einem Laptop.

### 3.7.3 Präzision und Recall für Klassen unterschiedlichen Umfangs

In einigen Situationen kann die Genauigkeit ein Maß sein, welches in die Irre führt. Nehmen wir an, dass in unserer Anwendung nur zwei Klassen vorliegen und die zweite Klasse nur bei einem Prozent der Testbeispiele zutrifft. Dann erreicht ein Modell 99 % Genauigkeit, wenn es **alle** Testbeispiele der ersten Klasse zuordnet. Trotz der hohen Genauigkeit ist das Klassifikationsmodell offenbar unbrauchbar.

Daher hat man für die Klassifikation noch zwei zusätzliche Qualitätsmaße für jede Klasse definiert: die Präzision (engl. Precision) der Klasse  $k$  und den Recall (auch Ausbeute, Abdeckung) der Klasse  $k$ . Der F-Wert der Klasse  $k$  ist der harmonische Mittelwert zwischen Präzision und Recall. Die genauen Definitionen finden sich in Abb. 3.32.

Ist z. B. die Präzision einer Klasse 0,90, so sind 90 Prozent der Testbeispiele, die der Klasse zugeordnet wurden, auch korrekt zugeordnet. Allerdings weiß man dann noch nicht, welcher Anteil der Testbeispiele der Klasse auch von dem Klassifikator „gefunden“ wurde, also der Klasse zugeordnet wurde. Dieser Anteil ist gerade der Recall. Wie hoch

$$\text{Genauigkeit} = \frac{\text{Anzahl der korrekt klassifizierten Testbeispiele}}{\text{Anzahl aller Testbeispiele}}$$

$$\text{Präzision}_k = \frac{\text{Anzahl aller der Klasse } k \text{ korrekt zugeordneten Testbeispiele}}{\text{Anzahl aller der Klasse } k \text{ zugeordneten Testbeispiele}}$$

$$\text{Recall}_k = \frac{\text{Anzahl aller der Klasse } k \text{ korrekt zugeordneten Testbeispiele}}{\text{Anzahl aller Testbeispiele, die zu Klasse } k \text{ gehören}}$$

$$F\text{-Wert}_k = 2 * \frac{\text{Präzision}_k * \text{Recall}_k}{\text{Präzision}_k + \text{Recall}_k}$$

**Abb. 3.32** Gebräuchliche Performanzmaße für die Klassifikation. Die Genauigkeit betrachtet alle Klassen gemeinsam, während sich  $\text{Präzision}_k$ ,  $\text{Recall}_k$  und  $F\text{-Wert}_k$  auf die  $k$ -te Klasse beziehen

die Präzision und der Recall sein sollten, hängt von der Anwendung ab. Sollen möglichst viele potentielle Trainingsbeispiele der Klasse identifiziert werden, so sollte der Recall hoch sein. Soll dagegen die Zuordnung zu der Klasse mit hoher Sicherheit korrekt sein, so sollte die Präzision hoch sein. Bei vielen Klassifikatoren gibt es die Möglichkeit, das relative Gewicht von Präzision und Recall zu beeinflussen.

## 3.8 Zusammenfassung

Beim maschinellen Lernen wird ein Modell so an Daten angepasst, dass sich damit relevante Informationen gewinnen lassen. Beim überwachten Lernen annotiert man die Daten direkt mit der gesuchten Antwort, während das Modell beim unüberwachten Lernen Zusammenhänge innerhalb der Daten erfasst. Das Bestärkungslernen beschäftigt sich mit zeitlich ausgedehnten Situationen (z. B. Spiele), bei denen das Endergebnis erst nach einer Reihe von Zeitschritten verfügbar ist.

Das Lernen läuft nach einem allgemeinen Schema ab. Zunächst führt das Modell mit den Eingaben eine Prognose durch. Anschließend vergleicht die Verlustfunktion die Prognosen mit den Trainingsausgaben. Danach ändert die Optimierung die Modellparameter so ab, dass der Verlust geringer wird. Dies geschieht nahezu immer durch die Berechnung des Gradienten, der in Richtung des steilsten Anstiegs bzw. Abstiegs des Verlustes zeigt. Der Verlust wird durch viele kleine Schritte in Gradientenrichtung so lange verringert, bis sich keine Änderung mehr ergibt. Dabei wird der einzelne Gradient meist nur mit wenigen Trainingsbeispielen berechnet, wodurch der Rechenaufwand dramatisch sinkt und lokale Minima vermieden werden.

Die Qualität des Modells wird durch die Berechnung von Performanzmaßen, wie der Genauigkeit oder dem F-Wert, bestimmt. Dabei müssen diese Maße immer auf einer Testmenge berechnet werden, d. h. Beispielen, die nicht zum Training verwendet wurden.

Der Begriff der Künstlichen Intelligenz hat in der letzten Dekade einen Bedeutungswandel erfahren. Zu Beginn hat die KI abstrakte kognitiven Fähigkeiten des Menschen nachgebildet, insbesondere zum logischen „Schließen“. Der Fortschritt in diesen Forschungsfeldern war begrenzt. In den letzten Jahren wurden aber viele Bereiche unter KI subsumiert, die vorher als maschinelles Lernen und Data Mining betrachtet wurden, z. B. Spracherkennung, Bildklassifikation oder auch Optimierung von Spielstrategien. Diese Techniken wurden nicht entwickelt, um das menschliche Denken zu imitieren, sondern um ein praktisches Problem zu lösen. Es besteht die Gefahr, dass durch die Bezeichnung „KI“ diese Techniken zu stark auf die Nachahmung der geistigen Fähigkeiten des Menschen fokussiert werden und andere Lösungsmöglichkeiten außer Acht lassen (Jordan 2018).

---

## Literatur

- Efron, Bradley und Hastie, Trevor. 2016.** *Computer Age Statistical Inference*. New York : Cambridge University Press, 2016.
- Jordan, M. 2018.** Artificial Intelligence — The Revolution Hasn’t Happened Yet. [Online] medium. com 2018. <https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7>.
- LeCun, Y., et al. 1998.** Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. 1998.
- Smith, C.S. 2020.** Computers Already Learn from Us. But Can They Teach Themselves? *New York Times*. 2020, 08.04.20.
- Tensor. 2019.** *Tensor*. [Online] Wikipedia DE, 2019. <https://de.wikipedia.org/wiki/Tensor>.



# Tiefes Lernen kann komplexe Zusammenhänge erfassen

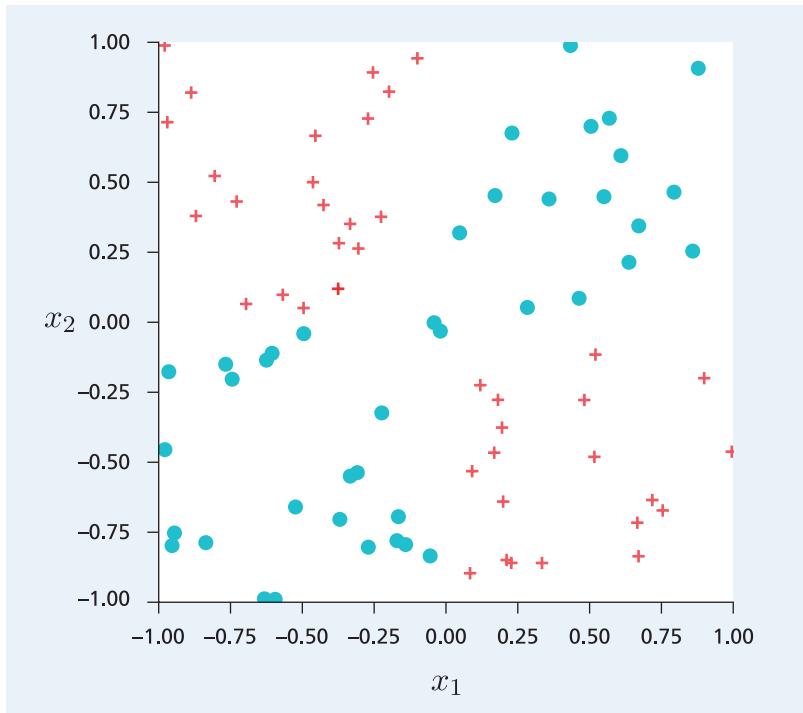
4

Logistische Regressionsmodelle funktionieren in vielen Fällen sehr gut. Der Kaggle Data Science Survey, eine Umfrage unter professionellen Informatikern, hat ergeben, dass die logistische Regression die mit Abstand am häufigsten verwendete Analysemethode ist (Kaggle 2017). Trotzdem hat sich gezeigt, dass die logistische Regression in einer Reihe von Fällen keine befriedigende Lösung finden kann.

Das logistische Regressionsmodell kann auch als ein einfaches neuronales Netz angesehen werden. Tatsächlich wurde eine vereinfachte Variante, das Perzeptron, als ein erstes künstliches neuronales Netz vorgeschlagen (Rosenblatt 1958). Es hatte einen Eingabevektor und eine einzige Ausgabeveriable. Sowohl die Komponenten des Eingabevektors als auch die Ausgabeveriable wurde auch als „Neuronen“ bezeichnet. Die Verbindungen zwischen den künstlichen Neuronen waren gewichtet, das heißt, sie waren nicht alle gleich stark. Ein Eingabe-Neuron leistet also einen individuellen Beitrag zur Erregung oder Hemmung des nächsten Neurons. Es wurde auch ein einfacher Lernalgorithmus präsentiert, der die Gewichte jeweils abänderte, wenn Fehler auftraten.

## 4.1 Beim XOR-Problem gibt es Interaktionen zwischen den Merkmalen

Wenn bei einem logistischen Regressionsmodell ein Eingabewert, z. B.  $x_1$ , steigt, so verändert sich auch die Wahrscheinlichkeit immer in eine Richtung. Das XOR-Problem ist ein Klassifikationsproblem mit einem zweidimensionalen Eingabevektor  $x = (x_1, x_2)$ , bei dem dies nicht gilt. Abb. 4.1 zeigt eine Trainingsmenge, in der 200 Trainingsbeispiele jeweils der Klasse • oder + zugeordnet sind. Die Klasse ist genau dann gleich •, wenn  $x_1 * x_2 > 0$ . Dies entspricht der XOR-Funktion in der Logik.

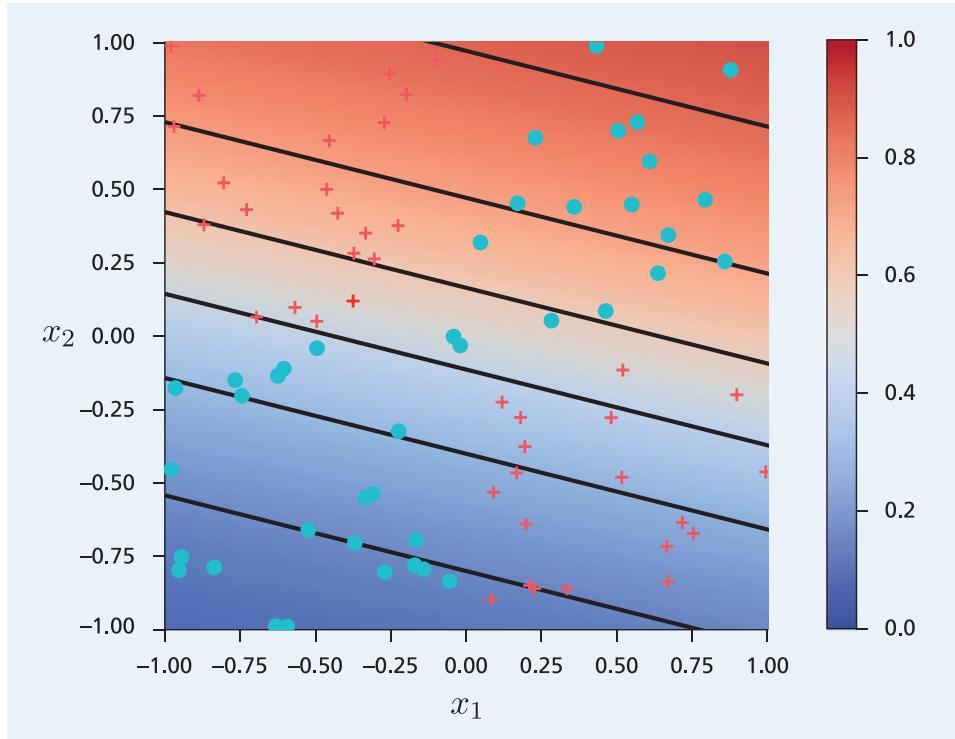


**Abb. 4.1** Der XOR Datensatz enthält Beobachtungen zu zwei Eingabeveriablen  $x_1, x_2$ , welche zwei Klassen • und + zugeordnet sind. Die Lage der Klassenbereiche bilden die XOR-Funktion nach: sie hat den Wert 0 (d. h. +), falls genau eine der Variablen  $x_1, x_2$  größer als 0 ist. Andernfalls hat sie den Wert 1 (d. h. •)

- Ist  $x_1 < 0$ , so sinkt die Wahrscheinlichkeit der Klasse • mit größer werdendem Wert von  $x_2$  von 1 auf 0.
- Ist hingegen  $x_1 > 0$ , so steigt die Wahrscheinlichkeit der Klasse • mit größer werdendem Wert von  $x_2$  von 0 auf 1.

Ist die Klassifikation von einem solchen „Zusammenspiel“ der Eingabemerkmale abhängig, so spricht man von einer statistischen Interaktion.

Abb. 4.2 zeigt ein logistisches Regressionsmodell, welches mit diesen Daten trainiert wurde. Die Trainingsbeispiele sind wieder durch • oder + markiert. Die Farben stellen die Wahrscheinlichkeit der Klassen dar. Die Linien sind „Höhenlinien“ gleicher Wahrscheinlichkeit. Offenbar ist das logistische Regressionsmodell nicht in der Lage, die beobachteten Trainingsbeispiele unterschiedlicher Klassen zu trennen. Wie die Höhenlinien zeigen, kann das logistische Regressionsmodell nur **gerade Trennflächen** zwischen Klassen erzeugen. Dies ist plausibel, da das logistische Regressionsmodell im Wesentlichen aus der



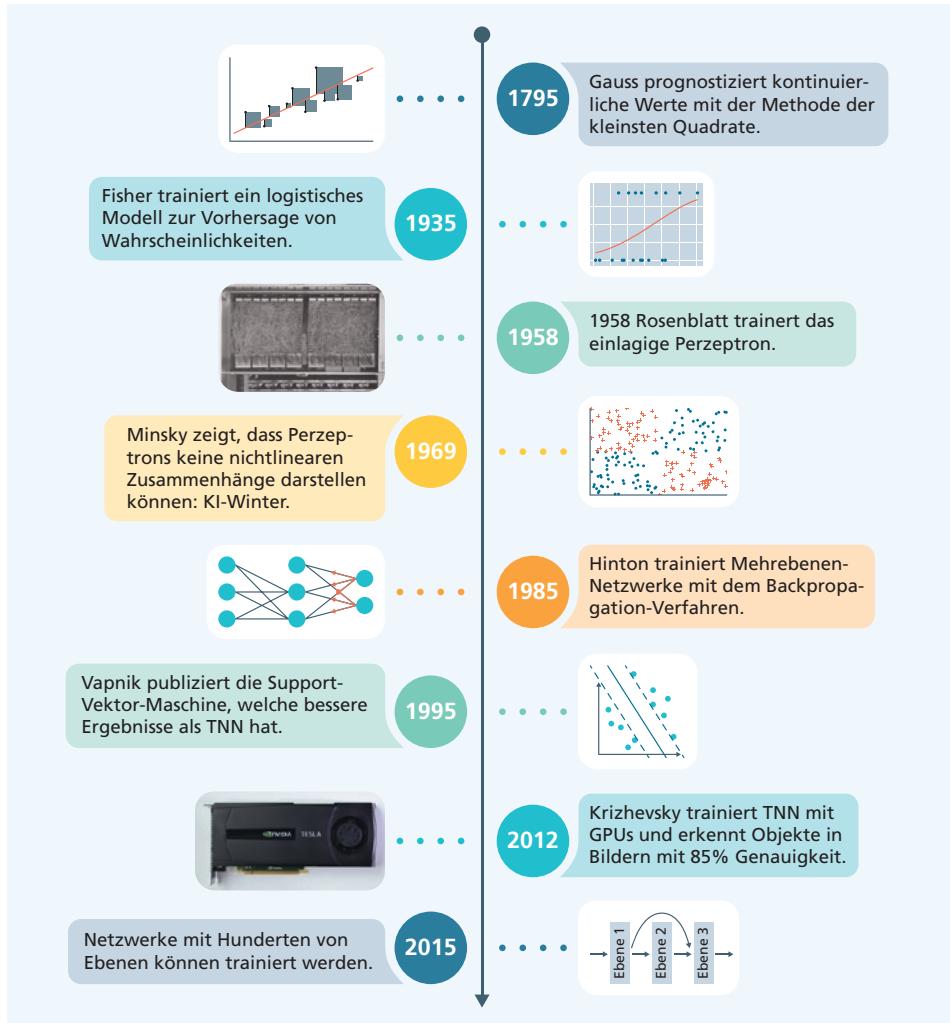
**Abb. 4.2** Die XOR-Daten lassen sich **nicht** mit einem logistischen Modell klassifizieren. Die Farben zeigen die Höhe der prognostizierten Wahrscheinlichkeiten. Rote Werte zeigen eine hohe Wahrscheinlichkeit der Klasse • und blaue Werte eine hohe Wahrscheinlichkeit der Klasse +. Rechts ist eine Farbskala für die Kodierung der Werte. Die Linien sind Höhenlinien gleicher Wahrscheinlichkeit

linearen Transformation besteht, welche gerade Linien immer in gerade Linien überführt (Abb. 3.15). Daher wird die logistische Regression auch als lineares Modell bezeichnet. Bei Klassifikationsproblemen, für die gebogene Trennflächen erforderlich sind, bringt die logistische Regression keine Lösung.

Minsky und Papert (1969) wiesen auf dieses Problem hin (Abb. 4.5) schlossen, dass die damals bekannten neuronalen Netze keine relevanten Probleme lösen können (Abb. 4.5). Diese Erkenntnis war in der Frühzeit der neuronalen Netze ein Schock und brachte die Forschung in diesem Bereich nahezu zum Erliegen. Erst Mitte der 80er-Jahre wurde klar, wie man mit dieser Herausforderung umgehen kann. Abb. 4.3 zeigt die Geschichte der maschinellen und tiefen Lernverfahren.

## 4.2 Nichtlinearitäten erzeugen gebogene Trennflächen

Rumelhart et al. (1985) schlugen die Verwendung von zusätzlichen Schichten innerhalb des neuronalen Netzes vor. Jede dieser Schichten musste eine nichtlineare Funktion enthalten, um gebogene Trennflächen darstellen zu können. In Abb. 4.4 sind die wichtigsten

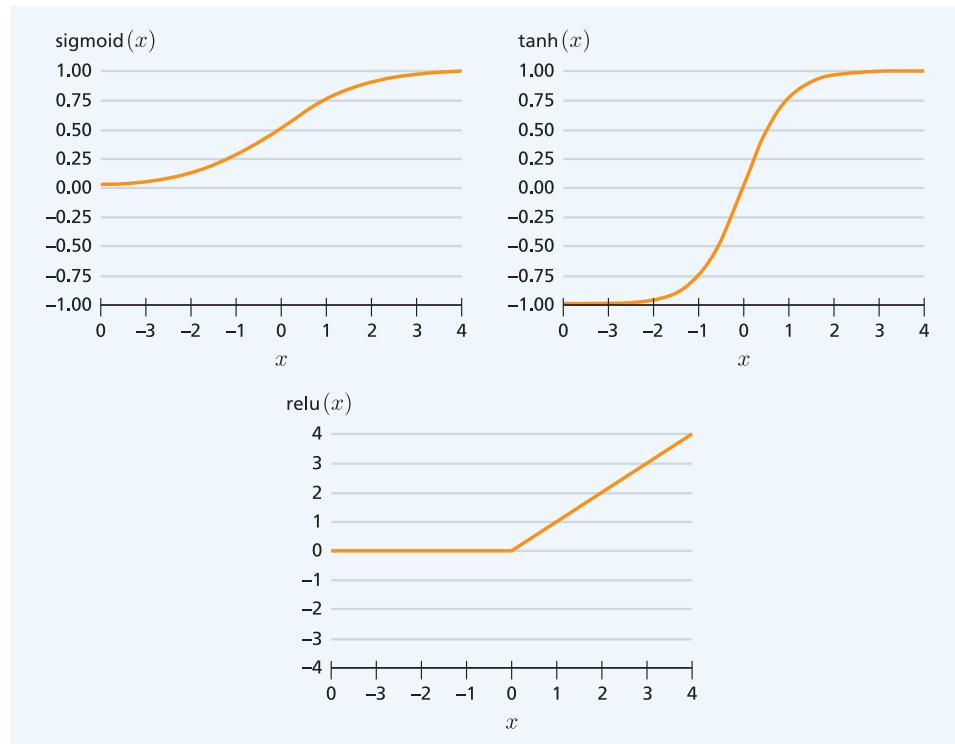


**Abb. 4.3** Geschichte der maschinellen und tiefen Lernverfahren. Nach anfänglicher Euphorie ging das Interesse an neuronalen Netzen im Jahr 1969 stark zurück. Nach einem zwischenzeitlichen Aufschwung erlahmte die Aufmerksamkeit erneut. Erst mit den technischen Entwicklungen der letzten Jahre erlebten TNN eine stürmische Renaissance. Bildnachweis im Anhang

dieser sogenannten Aktivierungsfunktionen dargestellt:

- die Sigmoid-Funktion  $\text{sigm}(x) = 1/(1 + \exp(-x))$   
mit der Ableitung  $\frac{\partial \text{sigm}(x)}{\partial(x)} = \text{sigm}(x)(1 - \text{sigm}(x)),$
- der Tangens Hyperbolicus  $\tanh(x) = (\exp(x) - \exp(-x))/(\exp(x) + \exp(-x))$   
mit der Ableitung  $\frac{\partial \tanh(x)}{\partial(x)} = (1 + \tanh(x))(1 - \tanh(x))$  und
- die abgeschnittene lineare Funktion (Rectified Linear Unit,  $\text{ReLU}(x) = \max(0, x)$ )  
mit der Ableitung  $\frac{\partial \text{ReLU}(x)}{\partial(x)} = 1$  falls  $x > 0$  und  $\frac{\partial \text{ReLU}(x)}{\partial(x)} = 0$  sonst.

Sie werden auf einen Vektor oder Tensor angewendet, indem man jede Komponente für sich transformiert. Eine Sonderstellung hat die abgeschnittene lineare Funktion, da sie eine „Ecke“ hat, in der keine Ableitung existiert. In der Praxis hat sich herausgestellt, dass



**Abb. 4.4** Populäre nichtlineare Aktivierungsfunktionen für neuronale Netze: Sigmoid-Funktion, Tangens Hyperbolicus und abgeschnittene lineare Funktion (Rectified Linear Unit, ReLU)



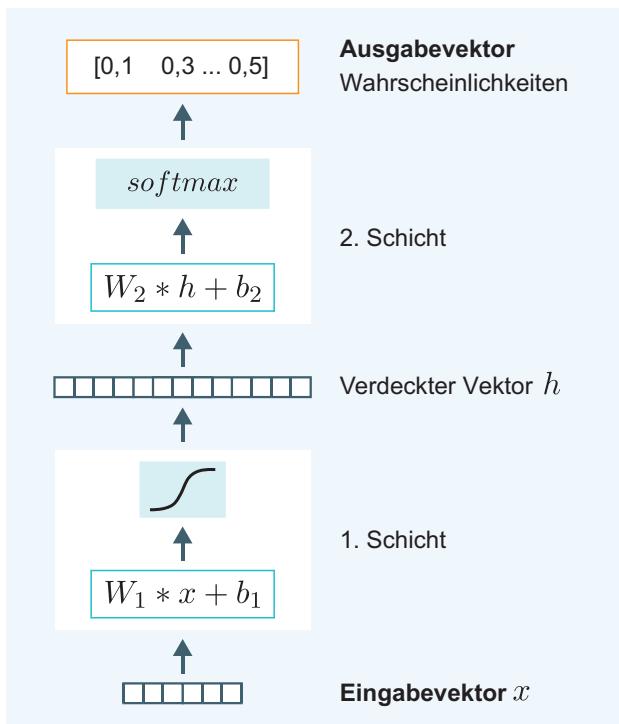
Marvin Minsky leistete von 1944 bis 1945 seinen Militärdienst ab und studierte im Anschluss Mathematik an der Harvard Universität. Für seine Dissertation baute er das erste neuronale Netz (SNARC) in Hardware. 1958 ging er zum MIT und gründete dort mit John McCarthy das AI-Lab. Er entwickelte eine 7,4 Turingmaschine, die beliebige Programme ausführen konnte. Zusammen mit Papert publizierte er das Buch *Perceptrons*, welches Schwächen der einlagigen Perzeptrons verdeutlichte. Beide entwickelten die „Society of Minds Theorie“, welche erklären will, wie Intelligenz aus der Interaktion nichtintelligenter Teile entstehen kann. 1975 folgte eine Theorie der „Frames“, welche eine breite Verwendung fand. 1969 erhielt er den Turing-Award.

**Abb. 4.5** Marvin Minsky, \* 9. August 1927 in New York; † 24. Januar 2016 in Boston, Massachusetts, war einer der Väter der symbolischen KI. Bildnachweis im Anhang

man das Gradientenabstiegsverfahren auch dann verwenden kann, wenn einzelne Ecken im neuronalen Netz enthalten sind, weil man so gut wie nie auf diese Ecke stößt.

In Abb. 4.6 wird die Struktur eines neuronalen Netzes mit zwei Schichten gezeigt. In der ersten Schicht findet eine lineare Transformation  $u_1 = W_1 * x + b_1$  statt, deren Ergebnisvektor dann einer nichtlinearen Transformation (z. B.  $h = \tanh(u_1)$ ) unterworfen wird. Das Resultat dieser Transformation ist ein Vektor  $h$ , dessen Werte nicht in den Trainingsdaten verfügbar sind. Daher nennt man ihn einen verdeckten Vektor. Die Länge dieses

**Abb. 4.6** Neuronales Netz mit zwei Schichten. Wichtig ist die nichtlineare Funktion in der 1. Schicht, die die Repräsentationsfähigkeit verbessert

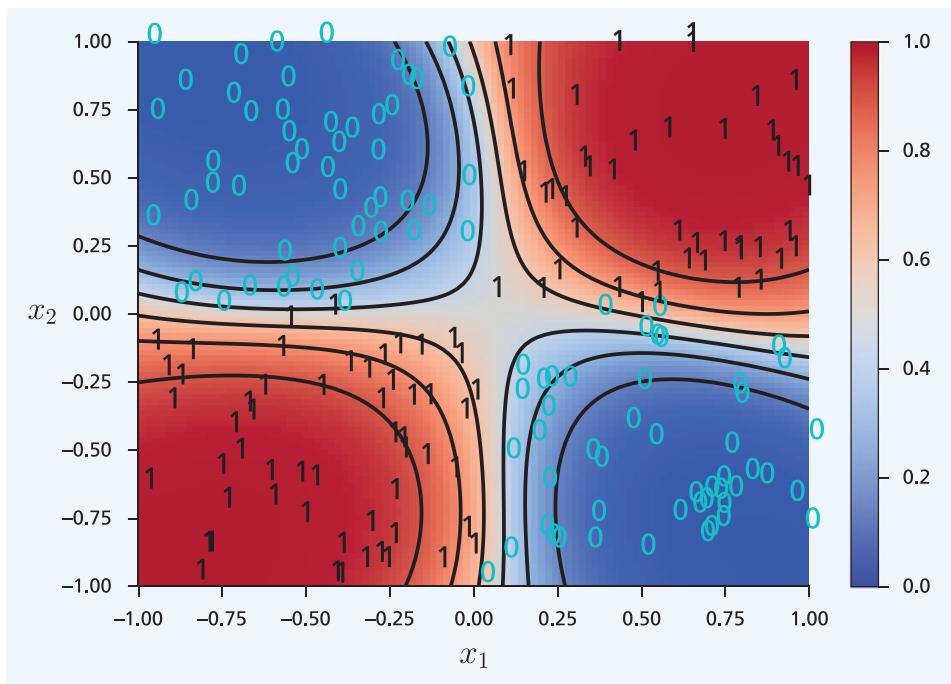


Vektors ist frei wählbar. Eine solche Schicht heißt auch vollverbundene Schicht, weil zwischen jeder Komponente des Eingabevektors und jeder Komponente des Ausgabevektors eine gewichtete Verbindung mit einem zugehörigen Parameterwert existiert.

Die zweite Schicht des neuronalen Netzes besteht aus einem logistischen Regressionsmodell mit einer weiteren linearen Transformation  $u_2 = W_2 * h + b_2$  und einer anschließenden Softmax-Transformation  $\text{softmax}(u_2)$ . Diese produziert den Vektor der Klassenwahrscheinlichkeiten. Insgesamt gibt es jetzt mehr unbekannte Parameter  $w = (W_1, b_1, W_2, b_2)$ , welche in dem Vektor  $w$  zusammengefasst und durch das Optimierungsverfahren modifiziert werden können.

Historisch war die Berechnung der Ableitungen für neuronale Netze mit mehreren Schichten ein großes Problem. Erst ab Mitte der 70er-Jahre wurde der zuvor entwickelte Backpropagation Algorithmus hierfür verwendet (Werbos 1974), der eine effiziente Berechnung des Gradienten ermöglichte. Heute muss man sich nicht mehr um die Bestimmung des Gradienten kümmern; dies wird von dem Programmsystem für neuronale Netze automatisch erledigt.

Wir haben ein neuronales Netz mit zwei Schichten und einem verdeckten Vektor der Länge 4 auf unsere XOR-Daten angewendet. Es hat insgesamt 22 Parameter. Das Ergebnis wird in Abb. 4.7 gezeigt. Die Farben stellen die Wahrscheinlichkeit der Klassen dar. Die

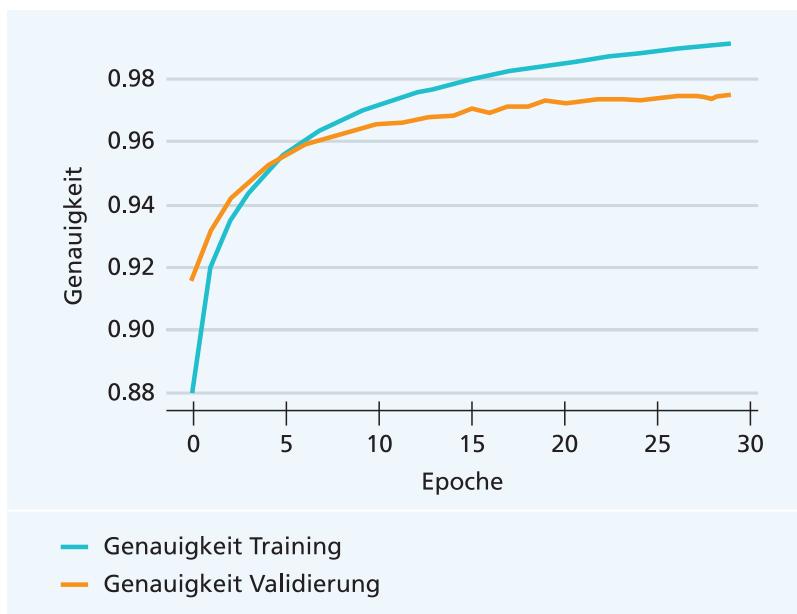


**Abb. 4.7** Klassifikationsergebnis für ein neuronales Netz mit zwei Schichten und einem verdeckten Vektor der Länge 4. Die Farben zeigen die Höhe der prognostizierten Wahrscheinlichkeiten. Rote Werte zeigen eine hohe Wahrscheinlichkeit der Klasse 1 und blaue Werte eine hohe Wahrscheinlichkeit der Klasse 0. Rechts ist eine Farbskala für die Kodierung der Werte. Die Linien sind Höhenlinien gleicher Wahrscheinlichkeit

Linien sind „Höhenlinien“ gleicher Wahrscheinlichkeit. Offenbar ist die Klassifikationsaufgabe durch das Modell wesentlich besser als beim logistischen Regressionsmodell gelöst. Die Genauigkeit auf der Testmenge erreicht 98 %. Der Fehler von 2 % lässt sich durch die zufälligen Unterschiede zwischen Trainings- und Testmenge erklären.

Ein neuronales Netz mit zwei Schichten kann auch auf den MNIST-Daten (Abschn. 3.2.3) trainiert werden. Die Länge des verdeckten Vektors wurde auf 100 festgesetzt. Abb. 4.8 zeigt die Genauigkeit auf der Trainings- und Testmenge der MNIST-Daten nach jeder Epoche. Die Genauigkeit auf der Trainingsmenge steigt stetig auf 0,991, während die Genauigkeit auf der Testmenge auf 0,975 steigt, d. h. von 100 Ziffern werden 2,25 falsch klassifiziert. Die Test-Genauigkeit ist wesentlich höher als die entsprechende Genauigkeit von 0,924 des logistischen Regressionsmodells. Dies bedeutet, dass nichtlineare Trennflächen in dieser Anwendung zu einer besseren Klassifikation führen. Die Genauigkeit auf der Testmenge ist wie erwartet deutlich geringer als auf der Trainingsmenge. Wir werden in den nachfolgenden Abschnitten untersuchen, wie man die Genauigkeit auf den MNIST-Daten noch steigern kann.

Der Erfolg der neuronalen Netze mit mehreren Schichten löste eine Welle der Euphorie aus. Rumelhart et al. (1985) schlugen vor, mithilfe neuronaler Netze mit mehreren Schichten Erkennungsvorgänge im Gehirn nachzubilden. Sie begründeten den Konnektionismus, der davon ausgeht, dass sich die Gehirntätigkeit durch Netze einfacher, verbundener Einheiten nachahmen lässt. Nach dieser Vorstellung entsprechen Neuronen einzelnen Komponenten in den verdeckten Vektoren, und die Synapsen sind vergleichbar mit den Verbindungen im neuronalen Netz, deren Stärke durch die Gewichtsparameter festgelegt ist.



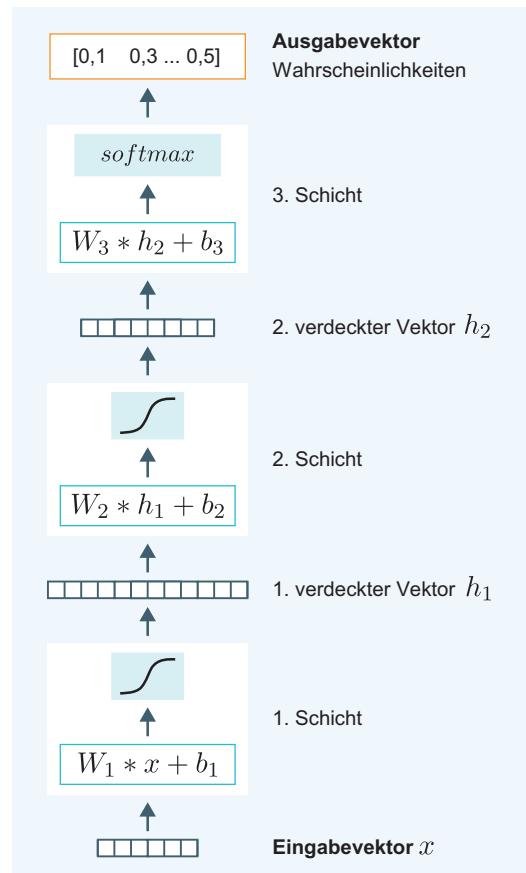
**Abb. 4.8** Genauigkeit eines neuronalen Netzes mit zwei Schichten auf der Trainings- und Testmenge der MNIST Daten

Schließlich konnte sogar formal bewiesen werden (Hornik et al. 1989), dass sich durch neuronale Netze mit zwei Schichten beliebige funktionale Zusammenhänge zwischen Eingaben und Ausgaben repräsentieren und lernen lassen. Bei realen Anwendungen ist es leider oft so, dass die Länge des verdeckten Vektors dann extrem groß werden muss und die Modelle in der Praxis nicht mehr trainierbar sind. Der nächste Abschnitt diskutiert Ansätze, wie man mit diesem Problem umgehen kann.

### 4.3 Tiefe neuronale Netze sind Stapel von nichtlinearen Schichten

Die Erfahrungen der letzten Jahre haben gezeigt, dass man die Erkennungsleistung der neuronalen Netze durch zusätzliche Schichten steigern kann. Abb. 4.9 zeigt ein neuronales Netz mit drei Schichten. Bei der Anwendung wird ein Eingabevektor sukzessive in den einzelnen Schichten transformiert und schließlich eine Ausgabe erzeugt. Daher nennt man solche Netze auch mehrschichtige Feedforward-Netze. Durch die Verwendung zusätzlicher Schich-

**Abb. 4.9** Mehrschichtiges Feedforward-Netz mit drei Schichten. Der Eingabevektor  $x$  wird transformiert in die verdeckten Vektoren  $h_1$  und  $h_2$  und schließlich in einen Wahrscheinlichkeitsvektor, der die Wahrscheinlichkeiten der einzelnen Klassen abschätzt



ten wächst die Flexibilität und Abbildungskapazität der Netze, allerdings auch die Anzahl der Parameter, die Menge der notwendigen Daten und der erforderlichen Rechenleistung.

Netze mit vielen Schichten bezeichnet man mit dem Sammelbegriff tiefes neuronales Netz (TNN). Allgemeinere Netztypen werden in den weiteren Kapiteln ausführlich vorgestellt.

Mittlerweile haben leistungsfähige TNN mehrere Hundert Schichten und viele Millionen Parameterwerte. Alle diese Schichten sind verdeckt, das heißt, die zugehörigen Vektoren sind nicht in den Daten beobachtbar. Die Schichten bilden dabei oft nicht mehr eine einfache Reihe, sondern es gibt auch Parallelverbindungen und Querverbindungen. Die Details werden in den folgenden Kapiteln diskutiert.

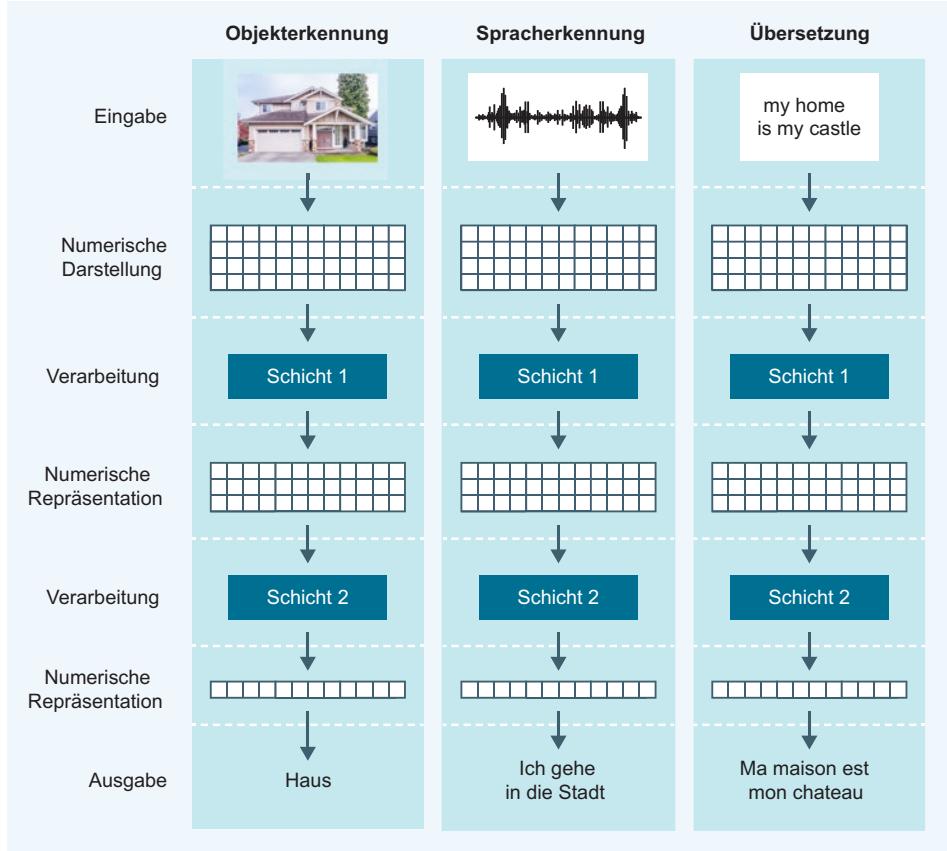
Tiefe neuronale Netze werden nicht programmiert, sie werden **trainiert**. Das heißt, niemand gibt von Hand ein, was welche Komponenten der verdeckten Vektoren genau zu tun haben. Die einzelnen Komponenten erhalten – ausgehend von den zufällig gewählten Anfangsparametern – während der Optimierung eine Funktionalität zugewiesen, sodass abschließend die Genauigkeit des Gesamtsystems möglichst hoch ist. Bei einer neuen zufälligen Verteilung von Anfangswerten wird die Funktionalität im Zuge der Optimierung wieder anders verteilt.

### 4.3.1 Vektoren und Tensoren repräsentieren die umgeformten Inhalte

Das tiefe neuronale Netz in Abb. 4.9 transformiert den Eingabevektor  $x$  in die verdeckten Vektoren  $h_1$  und  $h_2$ . Daher kann man  $h_1$  und  $h_2$  als **Repräsentationen** des Eingabevektors betrachten (Abb. 4.10). Die Repräsentationen werden durch das Training so gewählt, dass die finale Aufgabe, die Klassifikation der Eingabe, möglichst gut gelingen kann. Daher werden die Repräsentationen die wesentlichen Informationen über die Eingabe enthalten, welche eine Klassifikation ermöglichen.

Die Repräsentationen sind also eine Art Sprache, mit der das Netz die Daten beschreibt. Wie wir gesehen haben, können neuronale Netze sehr viele unterschiedliche Eingaben verarbeiten: Bilder, Töne, Sprache, Videos, Messwerte usw. Alle diese Eingaben werden durch Vektoren, Matrizen oder Tensoren dargestellt. Innerhalb der TNN werden während des Trainings sukzessive weitere Repräsentationen generiert, welche schließlich erlauben, die gewünschte Ausgabe des TNN zu erzeugen.

Die automatische Erzeugung von verbesserten Repräsentationen der Eingaben kann man als eine charakteristische Eigenschaft tiefer neuronaler Netze betrachten. Im Gegensatz zu den klassischen Ansätzen des maschinellen Lernens muss nicht der Anwender selbst geeignete Merkmale und Repräsentationen der Daten festlegen, sondern das tiefe Lernen generiert automatisch die besten Repräsentationen, die dann in optimaler Weise die gewünschte Prognose oder Klassifikation erlauben. Die theoretische Analyse von TNN hat gezeigt, dass die Darstellungsmächtigkeit tiefer neuronaler Netze exponentiell wächst: jede zusätzliche Schicht vervielfacht also die Anzahl der darstellbaren Zusammenhänge



**Abb. 4.10** Die Eingabe eines TNN wird in eine numerische Darstellung (Vektor, Matrix, Tensor) übersetzt. In jeder Schicht des Netzwerkes wird eine neue numerische Repräsentation der Eingabe erzeugt, sodass schließlich das Ziel des Netzwerkes (z. B. die Klassifikation von Objekten) möglichst gut gelingen kann. In realen Anwendungen ist die Struktur der Netze für die einzelnen Aufgaben sehr unterschiedlich. Bildnachweis im Anhang

(Eldan and Shamir 2016). Weiterhin kann man theoretisch ableiten, dass tiefere neuronale Netze räumlich gedrehte, verschobene oder transformierte Objekte wesentlich besser identifizieren können als flachere Netze (Bruna und Mallat 2013).

Man muss allerdings festhalten, dass das TNN nicht der einzige Modelltyp ist, der nichtlineare Trennflächen erzeugt. Alternativ sind nahezu alle fortgeschrittenen maschinellen Lernverfahren, wie etwa nichtlineare Supportvektormaschinen, Entscheidungsbäume, Random Forests,  $k$ -nächste Nachbarn usw. dazu in der Lage. Diese Ansätze sind allerdings meist nicht aus unterschiedlichen Schichten aufgebaut, sodass sich keine Hierarchien von jeweils komplexeren verborgenen Merkmalen entwickeln können. Daher ist die Genauigkeit der klassischen maschinellen Lernverfahren bei komplexen Problemstellungen meist nicht so gut wie die Genauigkeit der TNN.

## 4.4 Training von TNN mit Backpropagation-Verfahren

Ein TNN besteht aus einer Serie von Operatoren. In Abb. 4.9 hat jeder Operator einen Vektor als Eingabe und erzeugt einen Ausgabevektor, welcher Eingabe des nächsten Operators ist. Statt Vektoren können auch Skalare, Matrizen oder beliebige Tensoren als Eingabe und Ausgabe verwendet werden.

Eine Schwierigkeit bei einer Serie von Operatoren ist die Berechnung des Gradienten. Ausgangspunkt ist wieder die Trainingsmenge  $Train = \{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$  von Ein-/Ausgabepaaren. Wie in Abschn. 3.1.5 sollten die Parameter so gewählt werden, dass die Wahrscheinlichkeit der gesamten Trainingsmenge möglichst groß wird. Erst Mitte der 80-Jahre wurde der zuvor entwickelte Backpropagation Algorithmus hierfür verwendet (Werbos 1974), der eine effiziente Berechnung des Gradienten ermöglichte. Dieses Verfahren beruht auf der Kettenregel (Goodfellow et al. 2016, S. 199), welche die Berechnung des Gradienten von ineinander geschachtelten Funktionen erlaubt.

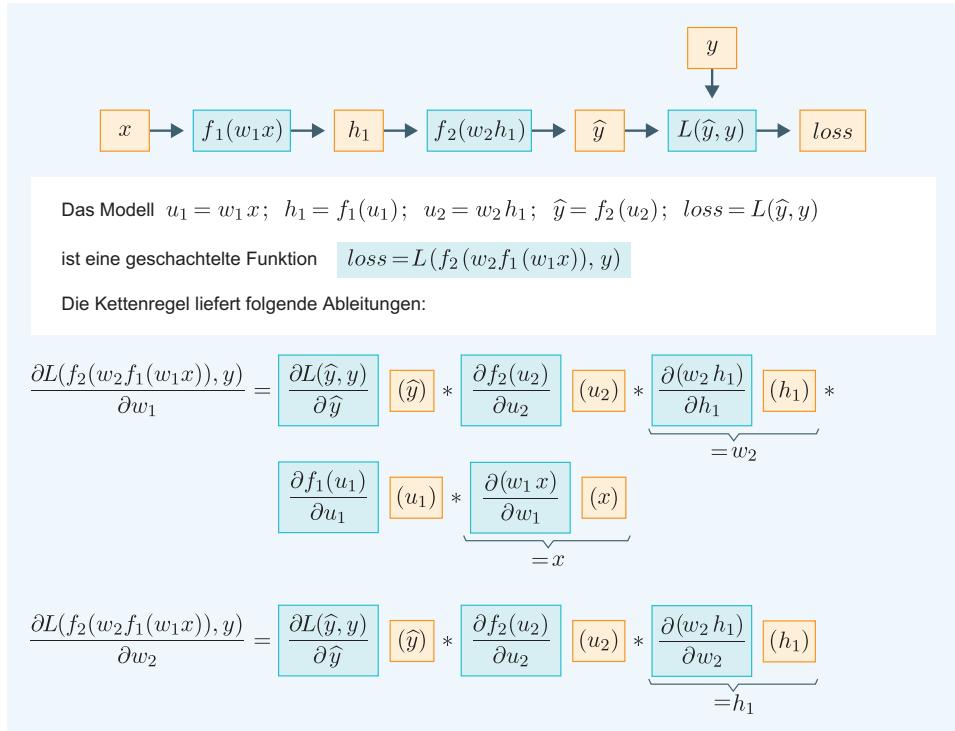
Die heutigen Toolkits berechnen die Ableitungen automatisch, sobald das Modell und die Verlustfunktion spezifiziert sind. Trotzdem ist es wichtig zu verstehen, welche Berechnungen notwendig sind um die Ableitung zu bestimmen und wie die zeitliche Reihenfolge der Berechnungen ist.

Als Beispiel nehmen wir ein extrem einfaches Modell mit einer Eingabe  $x$  der Länge 1, einer verdeckten Schichten mit Vektor  $h_1 = f_1(w_1 x)$  der Länge 1, einer Ausgabe  $\hat{y} = f_2(h_1)$  der Länge 1 und einer Verlustfunktion  $loss = L(\hat{y}, y)$ . Ziel ist es, durch Veränderung der Parameter  $w_1$  und  $w_2$  den Verlustwert  $loss$  zu minimieren. Das Modell bildet eine geschachtelte Funktion (s. Abb. 4.11). Die Berechnung der Ableitung nach den Parametern  $w_1$  und  $w_2$  gemäß der Kettenregel ist in Abb. 4.11 dargestellt.

Die Berechnung hat folgende wichtige Eigenschaften:

- Die Ableitungen der einzelnen Funktionen  $f_1(x)$ ,  $f_2(h)$  und  $L(\hat{y}, y)$  werden separat ausgerechnet (blau unterlegt), z. B. mit den Ableitungen aus Abschn. 4.2, und dann durch einfache Operationen (Multiplikation) miteinander verknüpft.
- Die Ableitungen sind selbst Funktionen (blau unterlegt), z. B.  $\frac{\partial f_2(u_2)}{\partial u_2}(u_2)$ , welche für bestimmte Zwischenergebnisse (gelb unterlegt) ausgewertet werden, z. B.  $u_2 = w_2 f_1(w_1 x)$ . Diese Zwischenergebnisse sind das Ergebnis der Vorwärtsberechnung für die vorherigen, inneren Funktionen.
- In den Ableitungen der unterschiedlichen Parameter  $w_1$  und  $w_2$  tauchen gemeinsame Ausdrücke auf, z. B.  $\frac{\partial L(\hat{y}, y)}{\partial \hat{y}}(\hat{y}) * \frac{\partial f_2(u_2)}{\partial u_2}(u_2)$ , die wiederverwendet werden können.

Diese Eigenschaften lassen sich nahezu unverändert auf Funktionen (also auch TNN) übertragen, welche mehrdimensionale Tensoren als Eingabe/Ausgabe haben. Daher verläuft der Backpropagation-Algorithmus zur Berechnung von Ableitungen für ein Trainings-Beispiel  $(x, y)$  in folgenden Schritten:



**Abb. 4.11** Für ein stark vereinfachtes Modell, bei dem Eingabe und verdeckte Schichten die Länge 1 haben, wird die Ableitung berechnet. Die Kettenregel zeigt, dass die Ableitung für ein Trainingsbeispiel  $(x, y)$  ein Produkt der partiellen Ableitungen (blau unterlegt) der einzelnen Funktionen ist. Sie werden jeweils für die prognostizierten Werte einer Vorrätsprognose (gelb unterlegt) berechnet. Für die Aktivierungsfunktionen  $f_1(u_1)$  und  $f_2(u_2)$  lassen sich die bekannten Ableitungen einsetzen

1. **Vorwärts-Propagierung:** Aus der Eingabe  $x$  wird eine Prognose nacheinander für die einzelnen Schichten des TNN berechnet und sämtliche Zwischenergebnisse, z. B.  $x, u_1, h_1, u_2, \hat{y}$  gespeichert.

2. **Rückwärts-Propagierung:**

Für die Schichten des TNN:

- a. Die Ableitung der letzten Schicht  $\frac{\partial L(\hat{y}, y)}{\partial \hat{y}}(\hat{y})$ , der vorletzten Schicht  $\frac{\partial f_2(u_2)}{\partial u_2}(u_2)$ , ..., der ersten Schicht  $\frac{\partial f_1(u_1)}{\partial u_1}(u_1)$  und  $\frac{\partial (w_1 x)}{\partial w_1}(x)$ , werden für die Werte der zuvor berechneten Zwischenergebnisse ausgewertet.

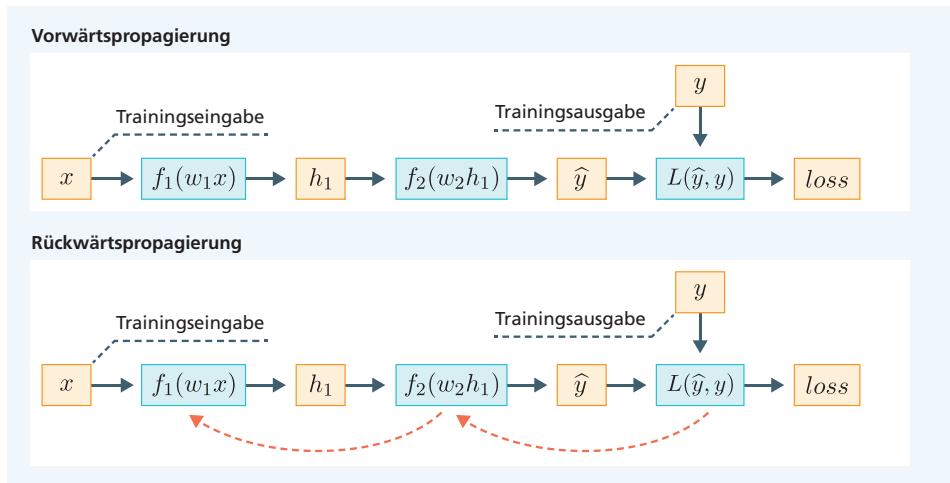
- b. Die Ergebnisse werden durch einfache Operationen (Multiplikation, Addition) miteinander verknüpft.

Kennt man das tiefe neuronale Netz, so können die Operationen automatisch bestimmt werden, welche notwendig sind, um den Gradienten auszurechnen. Dies wird in modernen Toolkits zur Programmierung von TNN realisiert.

Abb. 4.12 verdeutlicht den Informationsfluss während des Trainings für ein Trainingsbeispiel  $(x, y)$ . Bei der Vorwärts-Propagierung (oben) wird eine Prognose  $\hat{y}$  aus der Eingabe  $x$  berechnet und der zugehörige Verlust bestimmt. Alle Zwischenergebnisse werden gespeichert. Bei der Rückwärts-Propagierung werden die partiellen Ableitungen für die letzte, die vorletzte usw. Schicht für die gespeicherten Zwischenergebnisse ausgewertet und die jeweiligen partiellen Ableitungen der unbekannten Parameter berechnet.

Die Ableitungen für die einzelnen Trainingsbeispiele werden aufaddiert, da die Verlustfunktion eine Summe ist (Abschn. 3.5.2) und die Ableitung einer Summe gleich der Summe der Ableitungen ist. Im ursprünglichen Verfahren des Gradientenabstiegs wurden die Gradienten sämtlicher Trainingsbeispiele aufaddiert und anschließend der Parametervektor verändert (Abschn. 3.6.4). Heute werden meist die Gradienten für einzelne Mini-batches – zufällig ausgesuchten kleinen Teilmengen der Trainingsbeispiele – aufaddiert und anschließend die Parameter angepasst (Abschn. 3.6.6). Hierdurch lässt sich der Rechenaufwand sehr stark senken. Geoffrey Hinton hat die Verwendung Backpropagation Algorithmus maßgeblich vorangetrieben (Abb. 4.13).

Der Rechenaufwand zur Bestimmung des Gradienten ist für jedes Trainingsbeispiel mindestens so groß wie der doppelte Zeitbedarf für eine reine Vorwärts-Prognose. Hat man sehr viele Trainingsbeispiele, so ergibt sich ein großer Rechenaufwand. Hat man für



**Abb. 4.12** Informationsfluss in einem TNN beim Backpropagation-Verfahren für ein Trainingsbeispiel  $(x, y)$ . Oben: Bei der Vorwärtspropagierung (schwarze Pfeile) wird die Eingabe  $x$  durch das TNN propagiert und eine Ausgabe  $\hat{y}$  prognostiziert und damit der zugehörige Verlust berechnet, wobei alle Zwischenergebnisse gespeichert werden. Unten: Bei der Rückwärtspropagierung (gestrichelte rote Pfeile) werden die partiellen Ableitungen für die letzte, die vorletzte usw. Schicht mit den Werten der Zwischenergebnisse berechnet und dabei die Gradienten der unbekannten Parameter bestimmt



**Geoffrey Hinton** studierte Psychologie in Cambridge. Sein Ziel: „I wanted to understand how the mind worked“. Er erkannte „that psychologists didn't have a clue.“ In den 80er Jahren propagierte er die Verwendung von neuronalen Netzen zusammen mit dem Backpropagation Algorithmus zur Berechnung der Gradienten. „The brain sure as hell doesn't work by somebody programming in rules.“ In den Folgejahren trieb er an der Uni Toronto unbirrt die Weiterentwicklung neuronaler Netze voran. 2012 gelang ihm und seinen Doktoranden die Entwicklung des AlexNet, welches als eines der ersten TNN die Fehlerrate der Bildklassifikation stark reduzierte. 2018 erhielt er mit Yoshua Bengio und Yann LeCun den Turing-Award, die höchste Ehrung für Informatiker.

**Abb. 4.13** Geoffrey Hinton, \* 06.12.1947, ist einer der Väter der tiefen neuronalen Netze. Bildnachweis im Anhang

jedes Element der Trainingsmenge einmal den Gradienten berechnet, so wird dies eine Epoche genannt. Häufig müssen aber Gradienten in Hunderten von Epochen ausgewertet werden, wodurch sich ein extremer Rechenzeitbedarf ergibt.

---

## 4.5 Toolkits erleichtern die Formulierung und das Training von TNN

### 4.5.1 Parallele Berechnungen beschleunigen das Training von TNN

Das Training von TNN mit vielen Millionen Parametern dauerte oft mehrere Tage und Wochen. Aus diesem Grunde wurden Möglichkeiten untersucht, die Berechnungen durch die Verwendung paralleler Prozessoren zu beschleunigen. Dazu gibt es eine Reihe von Alternativen:

- Moderne Computer besitzen heute eine Anzahl (meist 4–32) Rechenkerne, welche separate Berechnungen auf dem gemeinsamen Speicher durchführen können.
- Ein Grafikprozessor (Graphical Processing Unit, GPU) ist ein auf die Erzeugung von Grafiken spezialisierter Computer mit derzeit bis zu 4600 Rechenkernen (Abb. 4.14). Diese GPUs wurden im Grafik- und Videospielbereich entwickelt, lassen sich aber auch über eine spezielle Programmiersprache CUDA für allgemeine Rechnungen nutzen. GPUs werden meist über einen normalen Computer angesprochen.
- Ein Cluster ist eine Gruppe von einzelnen Computern, gegebenenfalls mit GPUs, welche über ein schnelles Netzwerk verbunden sind. Compute-Cluster sind auch bei externen Anbietern via Internet verfügbar, d. h. in der „Cloud“.

**Abb. 4.14** Der Nvidia Tesla 2075 Grafikprozessor (GPU) mit 448 Rechenkernen hat eine Rechenleistung von bis zu 1,03 Teraflops, d. h. 1 Million mal 1 Million Flops (floating point operations). Bildnachweis im Anhang



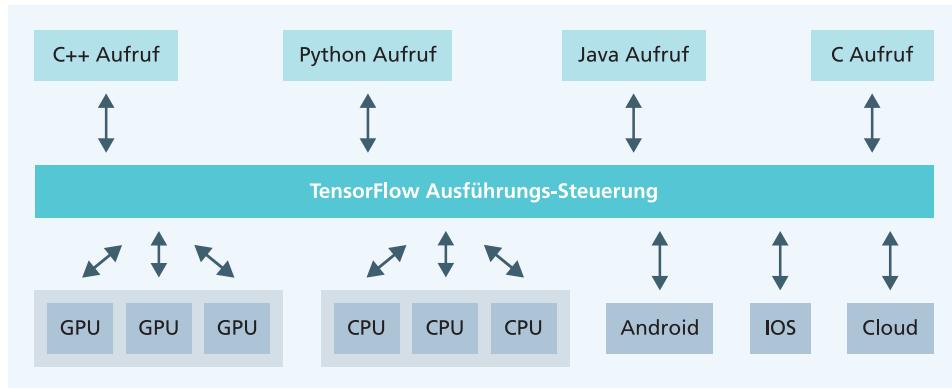
Die Parallelisierung des Trainings erfolgt meist in der Art, dass die verfügbaren Trainingsdaten eines Minibatches zwischen den Rechenkernen aufgeteilt werden. Jeder Rechenkern berechnet die Ableitungen für seine Daten separat. Die Ergebnisse werden anschließend aufaddiert und ergeben die Ableitung für den Minibatch. Um eine einfache Verwendung paralleler Rechner zu ermöglichen wurden Toolkits für tiefe neuronale Netze entwickelt.

### 4.5.2 Toolkits erleichtern die Arbeit mit TNN

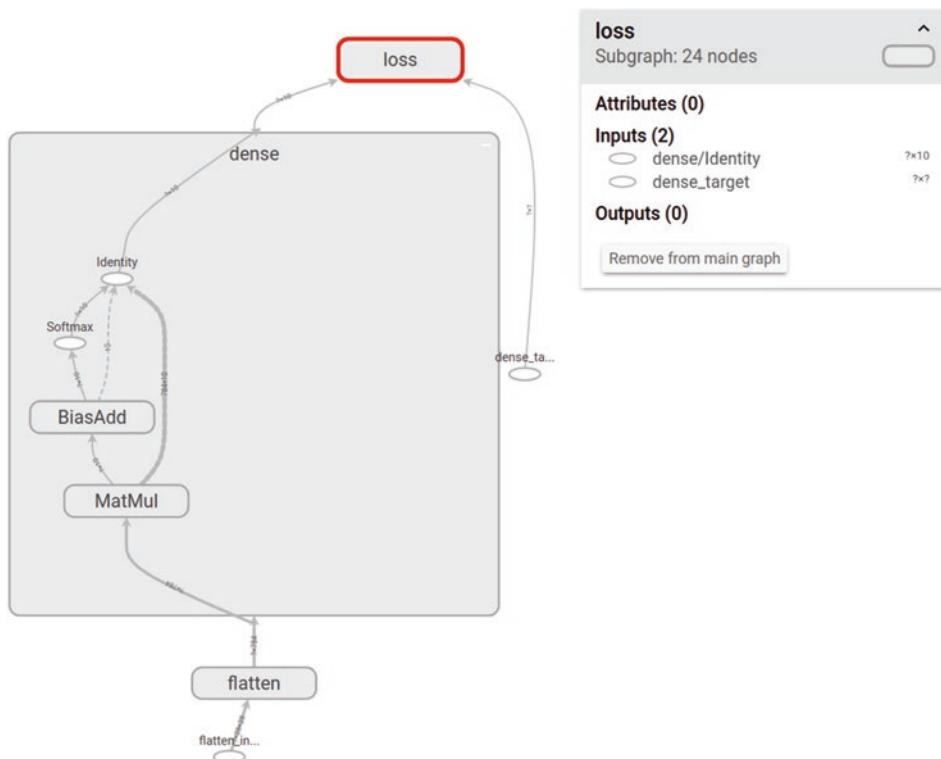
Wenn man ein TNN mit einer gebräuchlichen Programmiersprache programmiert, so gibt es viele Fehlerquellen. Spezielle Toolkits erlauben es, ein TNN einfach zu spezifizieren, zu trainieren und anzuwenden (Abb. 4.15).

Es gibt eine Reihe dieser Toolkits; einen Überblick findet man in (DL-Software 2019), von denen CNTK, PyTorch (PyTorch 2019) und TensorFlow (Abadi et al. 2016) die populärsten sind. Derzeit ist TensorFlow am weitesten verbreitet. TensorFlow wurde vom Google Brain-Team entwickelt und unter der Apache-2.0-Open-Source-Lizenz veröffentlicht. Es bietet folgende Funktionalität:

- Zusammenstellung des TNN aus unterschiedlichen Operatoren, welche beliebige Vektoren, Matrizen und Tensoren als Eingaben und Ausgaben haben können. Die Operatoren müssen nicht alle nacheinander ausgeführt werden, sondern es sind auch parallele Informationsflüsse erlaubt. Daher bilden die Operatoren einen gerichteten Graphen (Abb. 4.16).
- Die Ableitungen werden symbolisch ausgerechnet und als zusätzliche Operatoren dem Graphen hinzugefügt.
- Es sind eine Reihe von Optimierungsverfahren verfügbar, insbesondere der stochastische Gradientenabstieg (SGD) und dessen Varianten.
- Die Berechnungen werden automatisch auf die zur Verfügung stehende parallele Hardware ausgelagert und die Übertragung der Daten zwischen den Komponenten sicher gestellt. Abb. 4.15 zeigt die Architektur des TensorFlow-Systems.



**Abb. 4.15** Aufbau des TensorFlow-Toolkits. Die Berechnungen können auf CPUs, GPUs, Mobiltelefonen und in der Cloud ausgeführt werden. Der Aufruf ist aus verschiedenen Programmiersprachen heraus möglich



**Abb. 4.16** Von TensorFlow erzeugte Darstellung des Datenflusses in einem TNN

```

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(lr=0.1), metrics='accuracy')

```

**Abb. 4.17** Keras Programmcode für ein neuronales Klassifikationsnetz mit drei Schichten. Die erste Zeile definiert die Dimension der Eingabe. Zeilen 2-4 spezifizieren die Eigenschaften der Schichten. Die letzten beiden Zeilen legen die Verlustfunktion, den Optimierer und die Bewertungsmetrik fest

Das Toolkit ist für unterschiedliche Programmiersprachen verfügbar, bevorzugt in Python, aber auch Java, JavaScript, C++ usw. Keras (Chollet 2019) ist eine vereinfachte, nutzerfreundliche Bedienungssprache für TensorFlow und andere Programmsysteme. Ziel ist es, die gebräuchlichsten Netztypen flexibel und übersichtlich zu spezifizieren. Abb. 4.17 zeigt den Keras Programmcode für ein Klassifikationsmodell mit drei Schichten.

## 4.6 Wie mache ich das Netz besser?

Ein TNN hat eine große Anzahl von Hyperparametern (Stellgrößen), z. B. die Anzahl der Schichten, deren Verknüpfung oder auch die Lernrate des Optimierungsverfahrens. Beim Entwurf eines Modells ist es eine große Herausforderung, die Anzahl der Schichten, die Länge der verdeckten Vektoren und das Verbindungsmuster der Schichten festzulegen. Leider lassen sich derartige Hyperparameter nicht durch die normalen Optimierungsverfahren bestimmen.

Daher liegt es nahe, mehrere TNN mit unterschiedlichen Hyperparametern anhand ihrer Genauigkeit auf der Testmenge zu vergleichen. Anschließend kann das beste TNN ausgewählt und für die eigentliche Prognoseaufgabe verwendet werden. Bei einem solchen Vorgehen gibt es aber ein großes Problem: die Genauigkeit des besten TNN auf der Testmenge ist systematisch zu hoch, falls man mehrere Varianten des TNN vergleicht. Man kann sich also nicht mehr darauf verlassen, dass bei der Anwendung des TNN auf neue Daten im Mittel die gleiche Genauigkeit wie auf den Testdaten resultiert. Die Verzerrung liegt daran, dass ja gerade die Modellvariante mit der besten Genauigkeit auf den Testdaten ausgewählt wurde.

### 4.6.1 Iterative Konstruktion eines guten Modells mit der Validationsmenge

Wenn der finale Test des Modells mit der Testmenge eine zuverlässige Genauigkeit angeben soll, so muss man eine zusätzliche, zufällig bestimmte Validationsmenge auswählen, die wie die Testmenge beim Training nicht verwendet werden darf. Abb. 4.18 verdeutlicht diese zufällige Aufteilung der annotierten Daten. Auf der Validationsmenge kann man



**Abb. 4.18** Zufällige Aufteilung der Daten in eine Trainings-, eine Validierungs- und eine Testmenge

dann prüfen, welche Kombination von Hyperparametern die beste ist. Die Genauigkeit der ausgesuchten Modellvariante kann man dann zum Schluss einmal (und nicht öfter) auf der Testmenge abschätzen.

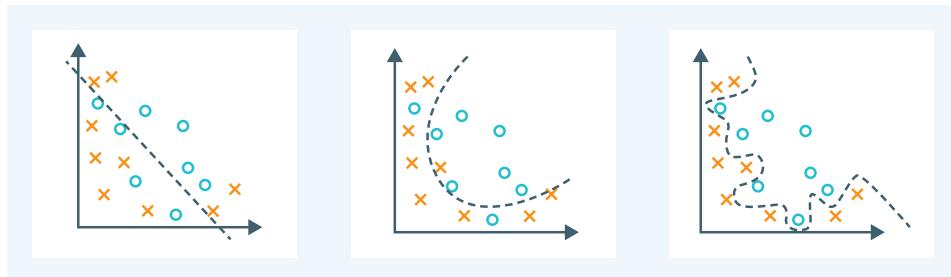
Dabei ist natürlich zu beachten, dass die Schätzung auf der verkleinerten Trainingsmenge nicht so genau sein kann wie auf der größeren Trainingsmenge. Es gibt theoretische Untersuchungen, welche besagen, dass eine Vervierfachung der Anzahl der Trainingsdaten zu einer Halbierung der Schwankungsbreite (Standardabweichung) der Modellprognosen führt. Daher ist es immer besser, eine Schätzung auf einer größeren Menge von Trainingsbeispielen durchzuführen.

#### 4.6.2 Unteranpassung und Überanpassung führen zu höheren Fehlern

Überprüft man die Genauigkeit eines neuronalen Netzes auf der Validationsmenge, so kann es zwei Arten von Problemen geben:

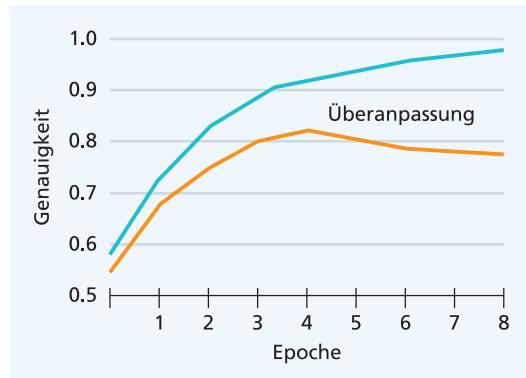
Bei der Unteranpassung bleibt die Genauigkeit sehr niedrig. Das kann z. B. geschehen, weil das Modell nicht in der Lage ist, sich den Trainingsdaten anzupassen. Dies wird in Abb. 4.19 auf der linken Seite dargestellt. Diese Situation lag beispielsweise beim logistischen Modell für die Ziffernklassifikation (MNIST-Daten) vor. Dann kann man die Komplexität des Modells erhöhen, indem man z. B. zusätzliche Schichten verwendet oder auf andere Art die Anzahl der Modellparameter erhöht. Es kann aber auch daran liegen, dass das Modell nicht lange genug trainiert wurde oder die Schrittweite bei der Optimierung zu groß ist.

Das Gegenteil der Unteranpassung ist die Überanpassung. In diesem Fall wird die Genauigkeit auf den Trainingsdaten viel höher als auf den Validations- oder Testdaten. Es kann sogar geschehen, dass die Genauigkeit auf den Validationsdaten ab einer bestimmten



**Abb. 4.19** Die Situation der Unteranpassung (links), der idealen Anpassung (Mitte) und der Überanpassung (rechts) bei einem Klassifikationsproblem

**Abb. 4.20** Verlauf der Genauigkeit auf den Trainingsdaten (blau) und den Validationsdaten (orange) bei Überanpassung. Es ergibt sich eine hohe Differenz. Im weiteren Verlauf des Trainings kann die Genauigkeit auf den Validationsdaten sogar fallen



Epoche wieder abfällt, während die Genauigkeit auf den Trainingsdaten weiter ansteigt. Dieser Zusammenhang ist in Abb. 4.20 dargestellt. Das Modell hat dann eine zu hohe Komplexität und zu viele Parameter. Wie in Abb. 4.19 auf der rechten Seite gezeigt, passt sich das Modell dann einzelnen Beobachtungen in der Trainingsmenge an und lernt die systematischen Zusammenhänge nur noch unzureichend.

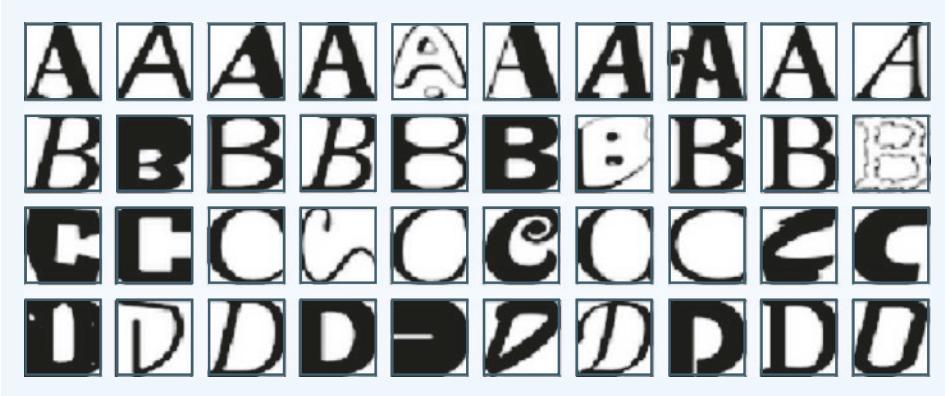
Eine einfache Technik, mit der Überanpassung umzugehen, ist das vorzeitige Stoppen. Hierbei wird während eines Optimierungslaufes die Genauigkeit auf den Validierungsdaten überwacht. Hat diese Genauigkeit einen Höchstwert erreicht und beginnt zu sinken, so wird die Optimierung abgebrochen und das Modell mit der höchsten Validierungsgenauigkeit verwendet. Diese Methode liefert manchmal ein brauchbares Modell und reduziert die Rechenzeit. Allerdings ist es durchaus möglich, dass es noch ein besseres Modell mit anderer Architektur für das Problem gibt.

Das ideale Verhalten des Modells liegt zwischen Unteranpassung und Überanpassung. Das Modell muss komplex genug sein, um die relevanten Zusammenhänge in den Daten zu erfassen, sollte aber nicht zu komplex sein und zufällige Variationen der Trainingsbeispiele lernen. Maßstab ist dabei immer der Generalisierungsfehler auf neuen Daten, d. h. die Genauigkeit auf den Validations- und Testdaten.

### 4.6.3 Ein Beispiel für Überanpassung

Um das Problem der Überanpassung zu demonstrieren, untersuchen wir ein Bildklassifizierungsproblem, welches schwieriger ist als die MNIST-Ziffernklassifikation. Dieser notMNIST-Datensatz besteht aus den Bildern der Buchstaben A bis J in einer Reihe unterschiedlicher Zeichensätze. Die Bildgröße beträgt 28x28 Pixel in 10 Klassen, wie bei MNIST (Abb. 4.21).

Trainiert man auf notMNIST-Trainingsdaten mit zehntausend Trainingsbeispiele ein 2-Schichten TNN mit einem verdeckten Vektor der Länge 1024, so erhält man 84,1 % Genauigkeit auf der Validationsmenge (dunkelblau in Abb. 4.22). Trainiert man dasselbe

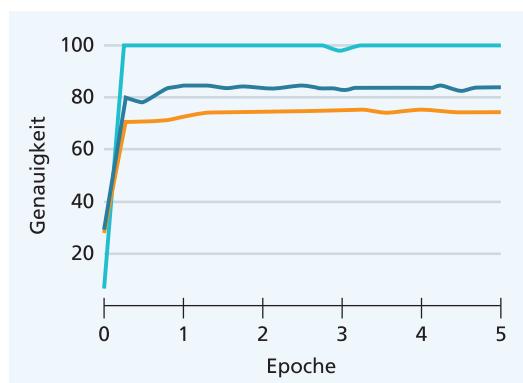


**Abb. 4.21** Trainingsbeispiele der Buchstaben A-D aus dem notMNIST Datensatz. Bildnachweis im Anhang

Netz mit nur 300 Trainingsbeispielen, so ergibt sich eine völlig andere Situation. Die Genauigkeit auf der kleinen Trainingsmenge geht gegen 100 % (hellblau) und die Genauigkeit auf den Validationsdaten (orange) stagniert für dieses Modell bei 75,1 %. Dies ist ein klarer Fall von Überanpassung. Das Netz ist in der Lage, alle Beispiele in der kleinen Trainingsmenge exakt zu prognostizieren. Es passt sich also an zufällige Variationen an und ist nicht mehr fähig, die charakteristischen Eigenschaften der Buchstaben zu erfassen. Konsequenterweise ist die Genauigkeit auf der Validationsmenge geringer.

Dies zeigt, dass auch ein Mangel an Daten zu einer Überanpassung führen kann. Falls weitere Daten verfügbar sind, sollte man sie also nutzen. Wenn  $N$  die Anzahl der Trainingsbeispiele ist, so zeigen Shalev-Shwartz und Ben-David (2014), dass der Prognosefehler durch  $\sqrt{(\text{Anzahl Parameter}) * \log(N) / N}$  beschränkt ist. Dies bedeutet, dass man näherungsweise die Anzahl der Trainingsbeispiele im gleichen Maße erhöhen muss wie die Anzahl der Modellparameter, damit der Prognosefehler nicht ansteigt.

**Abb. 4.22** Die Genauigkeit eines neuronalen Netzes mit zwei Schichten auf notMNIST. Bei Training auf 300 Beispielen: Genauigkeit auf den Trainingsdaten (hellblau), Genauigkeit auf den Validationsdaten (orange). Beim Training auf allen Daten (dunkelblau) ergibt sich eine höhere Genauigkeit



#### 4.6.4 Regularisierungsverfahren reduzieren den Generalisierungsfehler

Es gibt nun eine Reihe von speziellen Verfahren, welche eine Überanpassung reduzieren oder vermeiden können. Diesen Vorgang nennt man Regularisierung. Die meisten dieser Ansätze schränken die Komplexität des Modells ein:

- Die Bestrafung hoher Parameterwerte führt zu weniger starken Variationen (Schwankungen) der Modellfunktion.
- Frühes Abbrechen der Optimierung vermeidet die Trainingsphase, in der der Fehler auf den Validierungsdaten wieder wächst.
- Dropout zwingt das neuronale Netz, sich nicht auf einzelne Parameter zu verlassen.
- Die Batch-Normalisierung beschränkt die Variationsbreite der Werte in den verdeckten Vektoren.
- Die Reduktion der Anzahl der Modellparameter durch weniger Schichten oder kürzere verdeckte Vektoren verringern ebenfalls Variation der Ausgaben.

Diese Regularisierungsverfahren werden nachfolgend dargestellt.

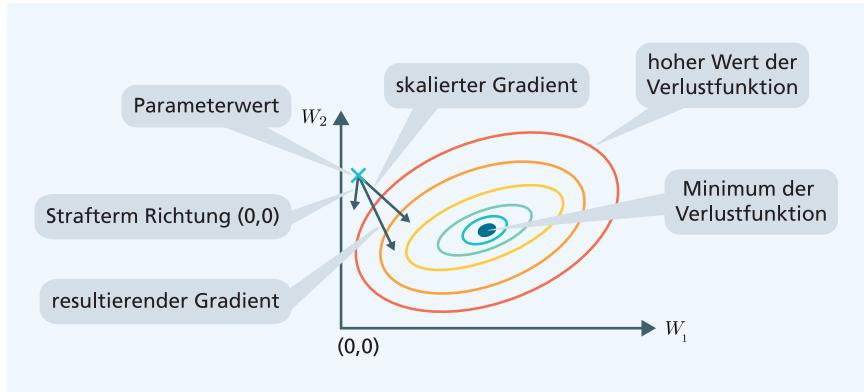
##### **Bestrafung großer Parameterwerte reduziert abrupte Ausgabeänderungen**

Bei der Bestrafung großer Parameterwerte ist das Ziel, hohe positive oder negative Werte der Modellparameter zu vermeiden. Denn diese hohen Modellparameterwerte führen zu starken Schwingungen in den Werten der Ausgabevektoren, welche bei der Überanpassung zu beobachten sind. Diese Bestrafung wird erreicht, indem man zu der Verlustfunktion des Modells einen weiteren Term hinzufügt, der einen zusätzlichen Verlustbeitrag bei hohen Parameterwerten produziert. Die Log-Likelihood Verlustfunktion (Abschn. 3.5.2) wird z. B. folgendermaßen modifiziert

$$\text{loss} = -\log P(\text{Train} \mid w) + a * (w_1^2 + \dots + w_n^2)$$

Die Summe der Quadratwerte aller  $n$  Parameter soll also reduziert werden, wobei  $a > 0$  ein Gewichtsfaktor ist. Das Optimierungsverfahren wird dann tendenziell die hohen Parameterwerte vermeiden und damit abrupte Schwingungen reduzieren. Das Verfahren heißt auch Weight Decay oder L2-Regularisierung und wird in Abb. 4.23 in zwei Dimensionen grafisch dargestellt. Dieser klassische Ansatz zur Regularisierung hat eine gute theoretische Begründung, weil der Prognosefehler durch die Summe der quadrierten Gewichte beschränkt ist (Neyshabur et al. 2015), und führt in vielen Fällen zum Erfolg.

Es gibt auch eine Variante, bei welcher nicht die Summe der Quadrate der Parameter, sondern die Summe der Absolutwerte der Parameter minimiert wird. Diese Art der Regularisierung heißt L1-Regularisierung. Sie hat als Nebeneffekt, dass viele Parameter auf 0,0 gesetzt werden. Allerdings können auch einzelne hohe Parameterwerte vorkommen, sodass die Regularisierungswirkung meist nicht so gut ist wie bei der L2-Regularisierung.

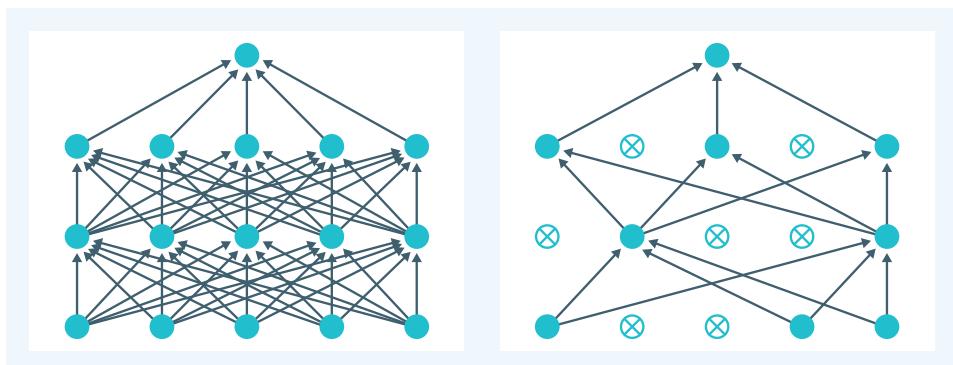


**Abb. 4.23** Regularisierung der Parameter durch einen Strafterm, der bei der Optimierung dem Gradienten hinzugefügt wird. Dadurch werden Parameter mit hohen positiven oder negativen Werten gegen  $(0,0)$  „gezogen“

### Dropout legt Teile des Netzes lahm

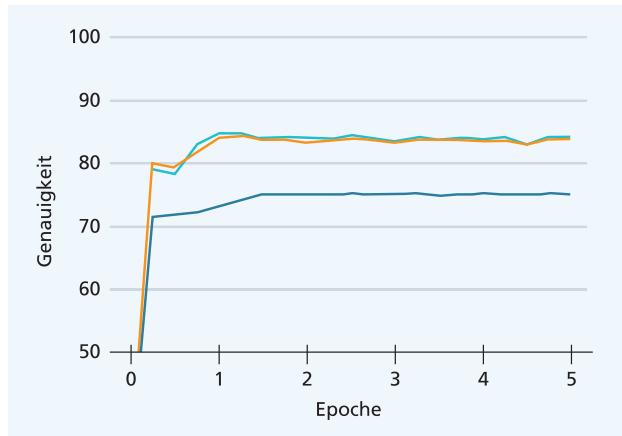
Beim Dropout werden während eines Trainingsschrittes zufällig ausgewählte Parameter des Modells auf 0 gesetzt. Dies wird in Abb. 4.24 dargestellt. Beim nächsten Minibatch zur Gradientenberechnung werden andere Parameter zufällig ausgewählt und die zugehörige Verbindungsstärke auf 0 gesetzt. Das neuronale Netz kann sich daher nicht auf einzelne Verbindungen „verlassen“, sondern muss zur Sicherstellung der Genauigkeit mehrere Verbindungen für die Repräsentation eines Zusammenhangs benutzen. Hierdurch ergibt sich eine starke Regularisierung. Im Allgemeinen erhalten jeweils 50 % der Parameter den Wert 0. Bei der späteren Anwendung eines mit Dropout trainierten Modells werden immer alle Verbindungen genutzt.

Wir können die Dropout-Regularisierung z. B. für das Training des oben beschriebenen notMNIST-Modells auf einem Teildatensatz von 300 Trainingsbeispielen verwenden. Die Er-



**Abb. 4.24** Beim Dropout werden während des Trainings bei einer Gradientenberechnung zufällig einzelne Parameter des vollen Modells (links) auf 0 gesetzt. Es entsteht das rechte Modell. Bei der nächsten Gradientenberechnung werden andere zufällig bestimmte Parameter auf Null gesetzt

**Abb. 4.25** Genauigkeit eines 2-Schichten neuronalen Netzwerkes auf den notMNIST-Validationsdaten. Beim Training auf 300 Beispielen mit Dropoutregularisierung ergibt sich eine Genauigkeit (hellblau), welche nahezu gleich hoch ist wie beim Training auf allen Daten (orange) und viel höher als ohne Regularisierung (dunkelblau)

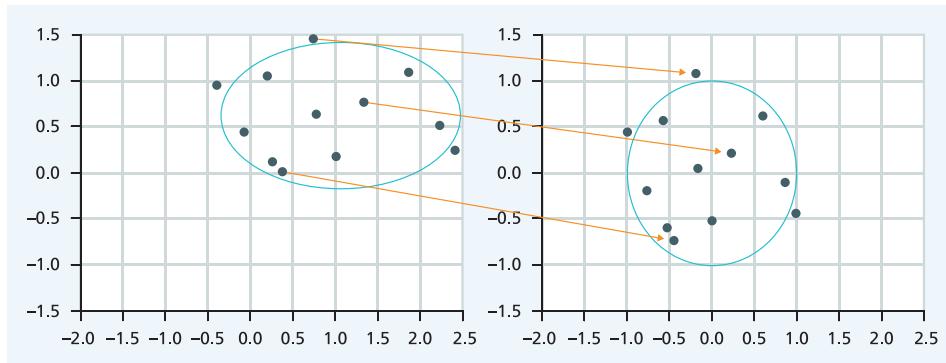


gebnisse sind in Abb. 4.25 dargestellt. Das Phänomen der Überanpassung verschwindet und Trainings- und Testgenauigkeit laufen nicht mehr auseinander. Die Validierungsgenauigkeit für das Teildatenmodell ist bei Dropout viel höher (83,8 %) als ohne Dropout (75,1 %). Überraschend ist, dass die Validierungsgenauigkeit nahezu gleich der Validierungsgenauigkeit (84,1 %) des Modells mit den vollen 10.000 Trainingsbeispielen ohne Regularisierung ist. Dies zeigt, dass die Dropout-Regularisierung ein wirksames Instrument bei der Reduktion von Überanpassung ist. Dropout wird bei einer großen Anzahl tiefer neuronaler Netze verwendet.

### Batch-Normalisierung vermeidet zu kleine und große Werte von verdeckten Vektoren

Die Batch-Normalisierung hat das Ziel, extreme Werte der verdeckten Vektoren zu vermeiden. Ein wichtiger Grund hierfür sind die nichtlinearen Funktionen, die sich in jeder Schicht des neuronalen Netzes finden. Hat eine Eingabekomponente des verdeckten Vektors einen hohen Wert – z. B. 100 – so sind die nichtlinearen Aktivierungsfunktionen (z. B.  $\tanh(x)$ ) in diesem Bereich sehr flach. Dementsprechend sind die Ableitungen nahezu 0 und es ist sehr schwer, aus diesem Wertebereich durch das Gradientenverfahren herauszufinden.

Die Batch-Normalisierung wird als eine spezielle Schicht in das Netz eingebaut und sorgt dafür, dass die Komponenten der Inputvektoren für die nächste Schicht für einen Minibatch im Mittel die Varianz 1 haben. Dazu werden die Minibatch-Mittelwerte jeder Komponente auf 0.0 normalisiert und die Minibatch-Varianzen auf 1.0. In Abb. 4.26 wird dies für zwei Dimensionen dargestellt; jedoch ist das Verfahren auf beliebig viele Dimensionen anwendbar. Es bewirkt einerseits, dass die Ableitungen nicht zu groß und nicht zu klein werden. Andererseits hat es zur Folge, dass das Netz extreme Wertschwankungen in den Ausgaben vermeidet. Damit ist die Batch-Normalisierung als Regularisierungsmethode verwendbar. Weiterhin ist das Netz weniger abhängig von einer passenden Initialisierung der Parameter und die Konvergenz der Optimierung wird verbessert. Das Verfahren hat sich insbesondere bei TNN mit sehr vielen Schichten bewährt.



**Abb. 4.26** Der Ablauf der Batch-Normalisierung eines verdeckten Vektors. Für die Trainingsbeispiele in einem Minibatch wird für jede Komponente des verdeckten Vektors (hier sind 2 Komponenten dargestellt) jeweils der Mittelwert und die Standardabweichung berechnet (links). Dann wird jede Komponente so transformiert, dass sie den Mittelwert 0.0 und die Standardabweichung 1.0 erhält (rechts).

### Mathematischer Beweis: Der stochastische Gradientenabstieg findet gut generalisierende TNN

Wir haben bisher ein TNN nach heuristischen Überlegungen aus Schichten von linearen Transformationen und nichtlinearen Aktivierungsfunktionen zusammengesetzt. Anschließend nutzten wir ein simples Optimierungsverfahren, den stochastischen Gradientenabstieg, um die Parameter so zu optimieren, dass die Verlustfunktion minimal wird. Niemand garantiert uns zunächst, dass dieses Vorgehen in der Lage ist, den tatsächlichen Zusammenhang zwischen Ein- und Ausgaben zu rekonstruieren. Allerdings konnten in der letzten Zeit durch theoretische Untersuchungen bewiesen werden, dass TNN durchaus die gesuchte Lösung finden können. Hardt et al. (2016) und Bassily et al. (2018) haben theoretisch bewiesen, dass der stochastische Gradientenabstieg nicht nur zu einer Verbesserung der Konvergenzeigenschaften führt, sondern auch eine Regularisierung des TNN bewirkt. Brutzkus und Globerson (2019) konnten für ein zweischichtiges TNN (unter einigen Annahmen) formal herleiten, dass beim Training mit dem stochastischen Gradientenabstieg das globale Minimum erreicht wird, und dass das TNN gut generalisiert. Damit konnte im Nachhinein die Eignung von TNN zur Approximation eines beliebigen Zusammenhangs zwischen Ein- und Ausgabedaten theoretisch und formal gezeigt werden.

---

## 4.7 Unterschiedliche Anwendungen erfordern Netze unterschiedlichen Aufbaus

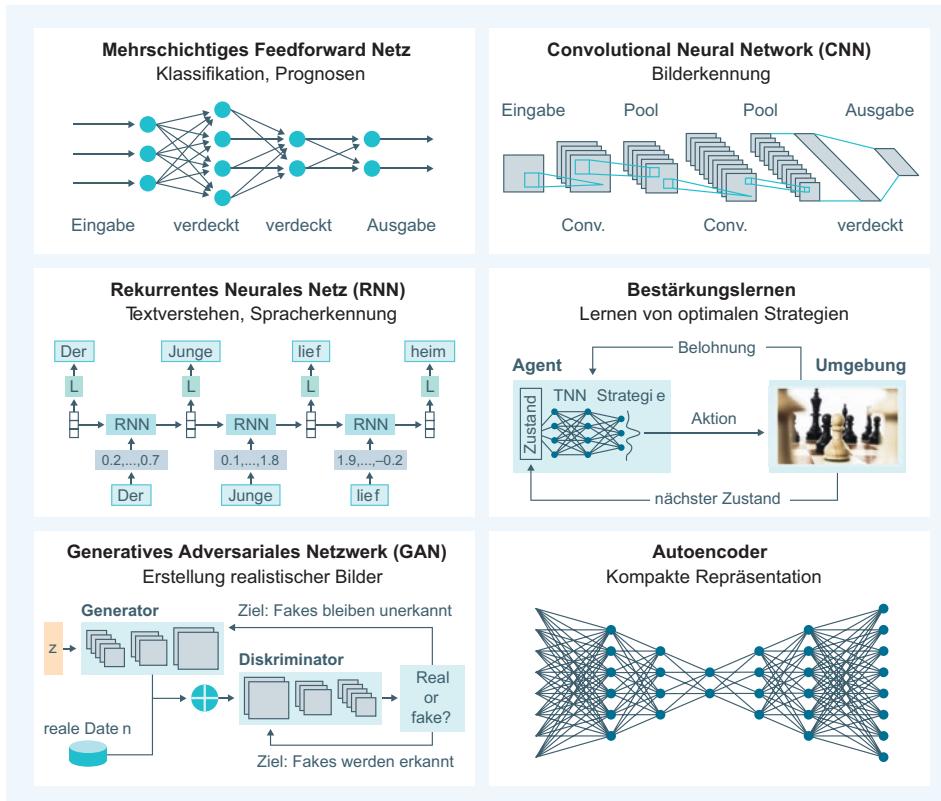
Bisher haben wir die Situation betrachtet, dass ein TNN aus einer Reihe von Schichten besteht und zum Schluss eine Ausgabe, z. B. eine Klassifikation, erzeugt. Für wichtige Anwendungen gibt es aber unterschiedliche Netztypen, welche so konstruiert sind, dass

sie die Erkennungsaufgabe durch konstruktive Eigenschaften unterstützen. Die wichtigsten dieser Netztypen sind in Abb. 4.27 zusammengestellt worden.

### 4.7.1 Mehrschichtiges Feedforward-Netz

Diese Art von TNN erhält einen Tensor als Eingabe und berechnet daraus eine Ausgabe. Soll das Netz die Eingabe in Klassen einteilen, so ist die Ausgabe meist ein Wahrscheinlichkeitsvektor mit den Wahrscheinlichkeiten der einzelnen Klassen. Dieses Netz wurde in Abschn. 4.3 beschrieben.

Ziel kann aber auch sein, eine oder mehrere kontinuierliche Größen zu prognostizieren. Man spricht dann von einem Regressionsproblem und das resultierende Modell heißt Regressionsmodell. In diesem Fall ist es erforderlich, den Unterschied zwischen der Prognose und der beobachteten Größe in den Trainingsdaten durch eine andere Verlustfunktion zu bewerten. Meist wählt man den quadratischen Abstand zwischen Prognose und Beobachtungswert als Verlustfunktion (Abb. 3.19). Mit dieser Änderung der Verlustfunktion



**Abb. 4.27** Zusammenstellung von sechs unterschiedlichen Netzwerkarchitekturen. Der Name der Architektur steht über der Grafik, darunter folgt der hauptsächliche Anwendungsbereich. Bildnachweis im Anhang

kann man dann das Modell zur Prognose kontinuierlicher Größen durch Optimierung trainieren.

### 4.7.2 Convolutional Neural Network (CNN)

Dieser Netztyp wurde speziell für die Erkennung von Bildern entwickelt. Es hat sich herausgestellt, dass man bei Bildern zunächst kleinteilige Eigenschaften, wie Ecken, Kanten und Farbflecken identifizieren muss. Sehr wichtig ist, dass diese Eigenschaften unabhängig von der Position auf dem Bild sind, also überall auftauchen können. Die Erkennung erfolgt durch eine sogenannte Convolution-Schicht mit einem sehr kleinen quadratischen Eingangsbereich (rezeptives Feld) von nur wenigen Pixeln. Dieses rezeptive Feld wird über das Pixelbild „geschoben“ und kann an unterschiedlichen Positionen die gleiche Eigenschaft identifizieren, z. B. eine Kante mit einer spezifischen Orientierung.

Typischerweise werden sehr viele alternative Eigenschaften erkannt, und das Netz konstruiert diese Eigenschaften selbsttätig, sodass sie zusammengenommen besonders aussagekräftig sind. In den höheren Schichten werden diese Eigenschaften zusammengefasst (Pooling) und abstraktere Merkmale durch weitere Convolution-Schichten erkannt. Die höchste Schicht bildet typischerweise ein logistisches Regressionsmodell zur Klassifikation der Bildinhalte. Diese Netze werden in Kap. 5 beschrieben.

### 4.7.3 Rekurrentes neuronales Netz (RNN)

Eine wichtige Art von Eingabedaten sind Sequenzen, z. B. Worte eines geschriebenen Textes oder Tonschwingungen bei gesprochener Sprache. Der Einfachheit halber bezeichnen wir die Komponenten der Sequenz im Folgenden als Wörter. Hier ist eine wichtige Eigenschaft, dass die Bedeutung der Wörter der Sequenz oft unabhängig von ihrer Position ist, solange der Kontext erhalten bleibt. Das bedeutet z. B., dass die gleiche Aussage zu Beginn oder am Ende eines Textes erscheinen kann. Man hat daher rekurrente neuronale Netze entwickelt, welche die Worte einer Sequenz sukzessive berücksichtigen. Dabei wird die Information über schon bearbeitete Worte der Sequenz in einem umfangreichen verdeckten Vektor gespeichert. Die Art der Kodierung der Inhalte wird automatisch während des Trainings festgelegt. Mit RNN kann man Texte von einer Sprache in eine andere übersetzen, neue Texte erzeugen oder aber ein Tonsignal der Sprache in einen Text transkribieren. Netze dieses Typs werden in den Kap. 6 und 7 dargestellt.

### 4.7.4 Bestärkungslernen-Netz

Eine wichtige Art von Problemen taucht in Situationen auf, bei denen das endgültige Ergebnis (z. B. Gewinn oder Verlust) erst nach einer Anzahl von Schritten bekannt wird. In jedem Schritt wird die Lage durch einen Zustand beschrieben, auf den man durch eine

neue Aktion reagieren muss. Dies ist beispielsweise für Spielsituationen und Robotersteuerungen der Fall. Daher ist es notwendig, nicht nur eine einzelne Aktion zu berechnen, sondern eine Strategie zu bestimmen, welche für beliebige Zustände eine gute Aktion auswählt. Das Bestärkungslernen stellt unterschiedliche Ansätze zur Verfügung, mit denen man eine solche Strategie berechnen kann, indem man sehr häufig die möglichen Spielzüge „durchspielt“ und die Ergebnisse als Trainingsdaten nutzt. Tiefe neuronale Netze werden dabei verwendet, um solche Strategien darzustellen und zu optimieren. Diese Art von tiefen neuronalen Netzen werden in Kap. 8 beschrieben.

#### 4.7.5 Generatives adversariales Netz (GAN)

Konventionelle TNN haben die Aufgabe, die Beziehung zwischen einer Eingabe und einer Ausgabe zu rekonstruieren. Manchmal möchte man aber auch neue Daten (im Weiteren Bilder genannt) erzeugen, die den Bildern in der Trainingsmenge ähneln. Generative adversariale Netze wurden für diese Aufgabe entworfen. Sie bestehen aus zwei Teilnetzen: Ein Generator-Netz erzeugt neue Bilder, während das Diskriminator-Netz versucht, echte Bilder von den künstlich erzeugten Bildern zu unterscheiden. Beide Netze werden gleichzeitig trainiert. Dabei versucht das Generator-Netz Bilder zu erzeugen, die vom Diskriminator als echt akzeptiert werden. Das Diskriminator-Netz soll dagegen alle generierten Bilder identifizieren. Im Erfolgsfall kann man dann die künstlich generierten Bilder nicht mehr von den echten Bildern unterscheiden. Dieser Typ von TNN wird in Abschn. 9.1 beschrieben.

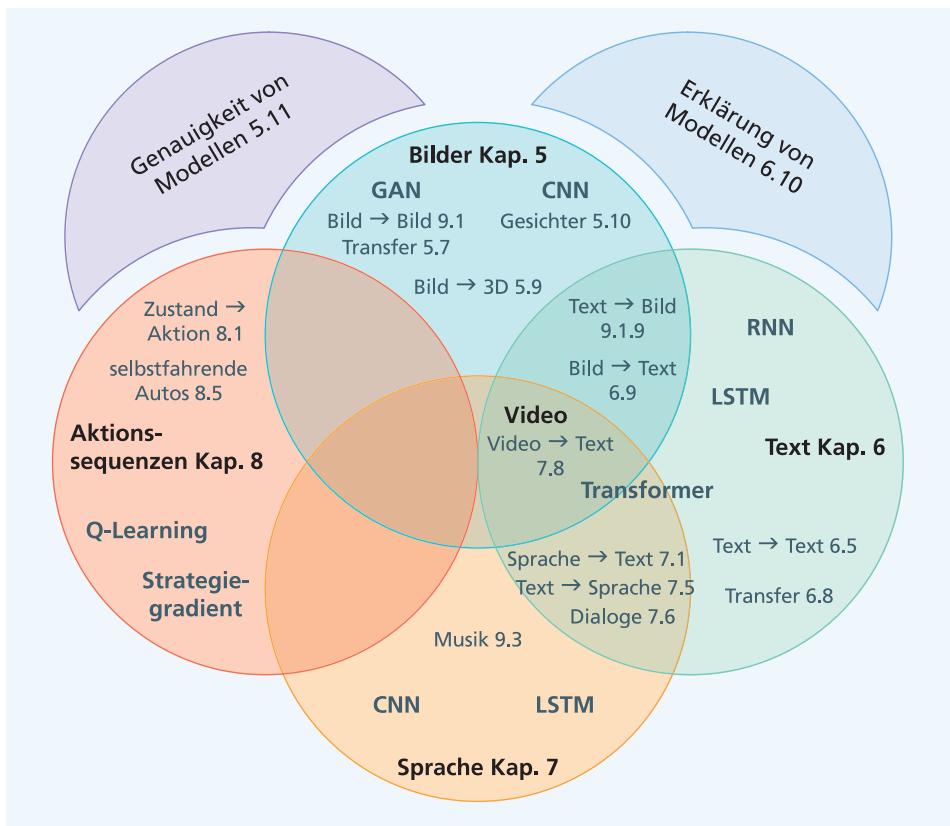
#### 4.7.6 Autoencoder-Netze erzeugen eine komprimierte Darstellung

Autoencoder-Netze sind eine Variante der mehrschichtigen Feedforward-Netze. Hierbei wird die Eingabe auch als Ausgabe verwendet, sodass das Netz die Eingabe reproduzieren muss, also keine Annotationen benötigt. Wichtig ist, dass eine dazwischenliegende Schicht signifikant kleiner ist als die Eingabe. Sie wird Informations-Flaschenhals genannt. Der Autoencoder ist daher gezwungen, die Informationen der Eingabe in diesen kleinen verdeckten Vektor zu komprimieren. Damit erzeugt es eine möglichst effiziente Kodierung des Eingabevektors. Diese TNN kann man verwenden, um effiziente Repräsentationen von Inhalten zu erzeugen oder auch um neue Daten zu generieren (Tschannen et al. 2018).

#### 4.7.7 Architekturen für bestimmte Medien und Anwendungsbereiche

Die Architekturen von TNN sind stark zugeschnitten auf die Medien und Anwendungsbereiche und bestimmen die Gliederung des Buches (Abb. 4.28). Bilder werden in Kap. 5,

Texte in Kap. 6, gesprochene Sprache in Kap. 7 und die Aktionssequenzen des Bestärkungslernen werden in Kap. 8 beschrieben. Wichtige Methoden zu den Medien sind in blauer Schrift dargestellt. Darüber hinaus gibt es noch einige Überlappungsbereiche, in denen ein Medium in ein anderes transformiert wird. Dies wird in der Grafik durch einen Pfeil ( $\rightarrow$ ) gekennzeichnet und dahinter durch den Abschnitt mit der Beschreibung dieses Anwendungsgebietes. Hier gibt es beispielweise die Übertragung Sprache  $\rightarrow$  Text in der Spracherkennung oder die Transformation Text  $\rightarrow$  Text bei der Übersetzung von einer Sprache in eine andere. „Transfer“ besagt, dass Modelle in einem Anwendungsbereich trainiert und auf einen anderen Anwendungsbereich übertragen werden. Oben rechts und links sind noch zwei allgemeine Abschnitte zur Genauigkeit und Erklärung von Modellen aufgeführt.



**Abb. 4.28** Die in diesem Buch betrachteten Medien und Anwendungsgebiete mit den zugehörigen Modellarchitekturen. Transformationen zwischen Medien sind durch  $\rightarrow$  und die zugehörige Abschnittsnummer gekennzeichnet

## 4.8 Die Konstruktion eines tiefen neuronalen Netzes ist ein Suchprozess

Der vorige Abschnitt hat deutlich gemacht, dass maschinelle Lernverfahren meistens keine Genauigkeit von 100 % erreichen. Vielmehr ist es ein langer Prozess, bis man ein Modell gefunden und trainiert hat, welches eine ausreichende Genauigkeit für den jeweiligen Anwendungsbereich besitzt. Dieser Prozess beinhaltet die Suche geeigneter Daten, die Erprobung unterschiedlicher Modellvarianten und die aussagekräftige Evaluation der Modelle.

### 4.8.1 Auswahl der Anzahl der Parameter und der Hyperparameter

Netzwerk-Hyperparameter	Optimierungs-Hyperparameter
<ul style="list-style-type: none"> <li>• Anzahl der Schichten und deren Verknüpfung</li> <li>• Typ der Schichten</li> <li>• Länge der verdeckten Vektoren jeder Schicht</li> <li>• Typ der nichtlinearen Funktion</li> <li>• Hyperparameter der Regularisierung</li> <li>• Art der Verlustfunktion</li> <li>• ...</li> </ul>	<ul style="list-style-type: none"> <li>• Auswahl des Optimierungsverfahrens</li> <li>• Globale Lernrate</li> <li>• Schema zu Verringerung der Lernrate</li> <li>• Auswahl des Genauigkeitsmaßes auf der Validationsmenge</li> <li>• Festlegung von Speicherungsintervallen für das Modell</li> <li>• Größe des Minibatches</li> <li>...</li> </ul>

Wir haben gesehen, dass die Genauigkeit von TNN von einer Vielzahl von Designentscheidungen und Stellwerten abhängt. Zu viele Parameter (zu große Komplexität) führen zur Überanpassung, während eine zu geringe Flexibilität (geringe Darstellungsmächtigkeit) des TNN eine Unteranpassung zur Folge hat. Daher hat der Netzkonstrukteur die Aufgabe, die richtige Kombination von Hyperparametern auszuwählen. Dabei gilt es eine Balance zu finden zwischen zu geringer und zu großer Komplexität.

Wie der vorstehende Kasten zeigt, gibt es eine ganze Reihe von Hyperparametern, die festzulegen sind. Sie bestimmen dann die Genauigkeit der Modellprognosen auf der Validations- und Testmenge.

Im Allgemeinen wird die Suche nach den besten Hyperparametern manuell durchgeführt (Goodfellow et al. 2016, S. 416). Ausgehend von den verfügbaren Daten und der Fragestellung wird man zunächst einmal im Internet nach einem Netz für eine vergleichbare Anwendung suchen. Eine unschätzbare Quelle hierfür ist die GitHub-Webseite (GitHub 2019), die eine riesige Sammlung von ausführbaren Programmen enthält. Hier werden viele Problemlösungen schon mit Beispieldaten für TensorFlow, PyTorch oder einer anderen Plattform für TNN zur Verfügung gestellt. Die Autoren geben dazu Informationen zu verfügbaren Daten, beschreiben die Qualität der Ergebnisse und verweisen auf zugehörige Publikationen. Diese Suche wird wertvolle Hinweise geben auf die grundlegende

Definiere ein einfaches Modell mit wenigen Schichten und wiederhole folgende Schritte:

- modifiziere eine oder mehrere Stellgrößen: Anzahl Schichten, Länge der verdeckten Vektoren, Lernrate, Regularisierungsparameter, ...
- trainiere das Modell auf den Trainingsdaten
- berechne die Genauigkeit auf den Validationsdaten
- falls das Modell besser ist als die vorherigen, so speichere die Kombination der Stellgrößen

**Abb. 4.29** Suche nach guten Kombinationen von Hyperparametern

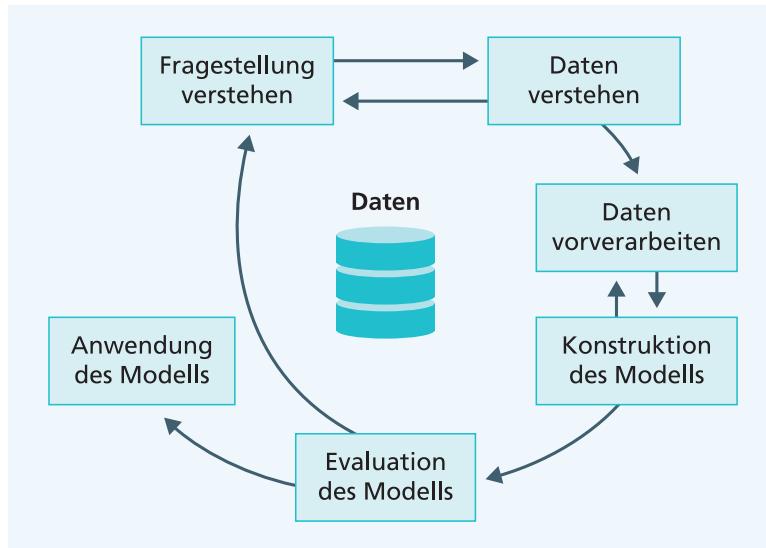
Netzarchitektur für das aktuelle Problem und liefert vielleicht schon das Grundgerüst für die geplante Anwendung.

Leider gibt es keine magische Formel, um die beste Architektur zu bestimmen. Man muss die beste Kombination von Hyperparametern in Experimenten finden. Die Suche wird meist stufenweise durchgeführt und startet mit relativ wenigen Schichten und Parametern. Anschließend werden die Größe und die Anzahl der Schichten schrittweise erhöht, bis der Verlust auf den Validationsdaten stagniert oder ansteigt. Abb. 4.29 beschreibt diesen Vorgang, bei dem nacheinander verschiedene Modellvarianten erzeugt werden und hinsichtlich ihrer Genauigkeit auf den Validationsdaten verglichen werden.

Häufig ist das Ergebnis eines Modellaufes teilweise von zufälligen Einflüssen überdeckt. Beispielsweise werden die Startwerte der Modellparameter oft zufällig gewählt und die Ergebnisse auf der Testmenge sind hiervon abhängig. Um den Mittelwert der Ergebnisse zu bestimmen, muss man dann diese Experimente mehrfach mit unterschiedlichen Startwerten wiederholen. Alternativ kann man die Startwerte für den Zufallszahlengenerator auf einen Wert fixieren. Dies ist besonders wichtig, falls man den Effekt vieler Hyperparameter untersucht. Dann empfiehlt es sich, ein kontrolliertes Experiment (Goodfellow et al. 2016, S. 415) durchzuführen, um zufällige Effekte von signifikanten Zusammenhängen zu trennen. Beachtet man diese Vorgehensweise nicht, so können „signifikante“ Ergebnisse in Kontrollexperimenten nicht reproduziert werden. Dieses Phänomen der „Replication Crisis“ tritt leider bei vielen aktuellen Forschungspublikationen auf (Replication 2019).

## 4.8.2 Der Standard-Prozess der Modellsuche führt zu besseren Modellen

Bei der iterativen Suche eines Modells kann man viele Fehler machen und wichtige Aspekte vergessen. Daher wurde ein Standardvorschlag für diesen Prozess entwickelt. Der Cross-Industry Standard Process for Data Mining (CRISP-DM) ist eine von vielen interessierten Firmen und Forschungsinstituten gemeinsam entwickelte Leitlinie (Shearer 2000). Sie sollte nicht nur für TNN, sondern bei beliebigen maschinellen Lernverfahren verwendet werden, und umfasst folgende Schritte (Abb. 4.30):



**Abb. 4.30** Phasen bei der Konstruktion eines Modells nach dem CRISP-DM Schema

*Fragestellung verstehen.* Hierbei geht es zunächst um das Geschäftsziel der Organisation, welches erreicht werden soll. Daraus ergeben sich das Ziel der Analyse und die Erfolgskriterien, mit denen die gesamte Analyse bewertet werden soll. Dies kann nicht nur ein optimaler Fehlerwert sein, sondern auch Kriterien wie Rechenzeitbedarf, Robustheit oder Erklärbarkeit.

- Bei einer Anwendung im Bereich der Bildklassifikation könnte das Analyseziel z. B. die Erkennung von fehlerhaften Produkten anhand von Fotos sein.

*Daten verstehen.* Dies beinhaltet die Suche und Festlegung geeigneter Daten und ihre Untersuchung sowie eine erste Prüfung auf Fehler und Ausreißerwerte.

- Bei der Klassifikation von Bildern sind beispielsweise Fotos in einer geeigneten Auflösung erforderlich, welche dann annotiert sind, z. B. als „fehlerhaft“ oder „ok“.

*Daten vorverarbeiten.* Dieser Schritt erfordert meist einen sehr großen Arbeitsaufwand. Hierzu gehört die eigentliche Sammlung und Erfassung der Daten, die Konsolidierung und Säuberung einschließlich der Erkennung von Datenfehlern. Anschließend müssen geeignete Trainingsbeispiele und Merkmale ausgewählt werden. Des Weiteren müssen die Daten zufällig in eine Trainings-, Test- und Validationsmenge aufgeteilt werden. Dabei muss Repräsentativität für die Anwendung erhalten bleiben, d. h. die Daten müssen die Verteilung der Trainingsbeispiele in der Anwendung widerspiegeln. Oft sind auch noch Datentransformationen notwendig.

- Bei der Klassifikation von Bildern ist beispielsweise eine Trainingsdatenmenge von annotierten Fotos zu sammeln und die Fotos auf Abbildungsfehler zu prüfen. Gegebenenfalls muss die Beleuchtung bei der Aufnahme von Fotos geändert werden.

*Konstruktion des Modells.* Hierzu gehört die Auswahl des Modells unter Beachtung des Analyseziels und der Eigenschaften der Daten. Dies muss nicht immer ein tiefes neuronales Netz sein, denn in vielen Anwendungsfällen sind klassische maschinelle Lernverfahren effizienter und robuster. Die Modellkonstruktion ist ein iterativer Prozess, in dem das Modell anhand der Trainingsdaten trainiert wird. Eine detaillierte Darstellung findet sich im nächsten Abschnitt.

- In unserer Beispielanwendung müssen unterschiedliche Bildklassifikationsverfahren trainiert und die Ergebnisse miteinander verglichen werden. Diese Verfahren werden in Kap. 5 dargestellt.

*Evaluation des Modells.* Hierbei wird gemessen, wie gut das Modell auf Validationsdaten arbeitet, welche nicht zum Training verwendet wurden. Die Qualitätsmaße hängen von dem Analyseziel und von dem ausgewählten Modelltyp ab. Nur wenn die Modellkonstruktion endgültig abgeschlossen ist, darf die Testmenge zur Evaluation verwendet werden. Ergeben sich bei der Evaluation grundsätzliche Probleme, so muss die Fragestellung geändert und der gesamte Prozess neu gestartet werden.

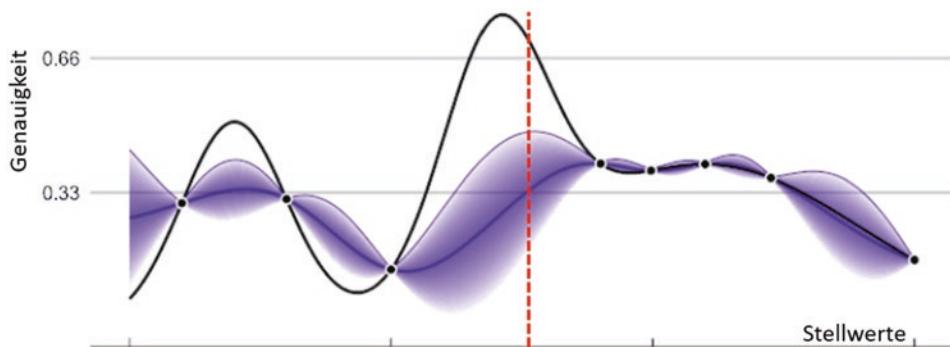
- In der Beispielanwendung zur Bildklassifikation muss geprüft werden, ob die erzielbare Genauigkeit der Klassifikation ausreicht. Zudem muss geprüft werden, ob das Verfahren robust gegen mögliche Störungen im laufenden Betrieb (Schmutz, Bildfehler, Vibrationen etc.) ist. Ist dies nicht der Fall, so muss ggf. der Prozess neu starten.

*Anwendung des Modells.* Hierzu gehört die Anwendung des Modells auf neue Daten. Oft wird das Modells auch in die Geschäftsprozesse und den Arbeitsablauf eingebunden. Falls sich die Daten im Ablauf der Zeit ändern, muss eine Neubestimmung des Modells vorgenommen werden.

- In unserer Beispielanwendung wird untersucht, ob die vorhergesagte Qualität der Bildklassifikation tatsächlich eingehalten werden kann.

### 4.8.3 Automatische Suche von Modellarchitekturen und Hyperparametern

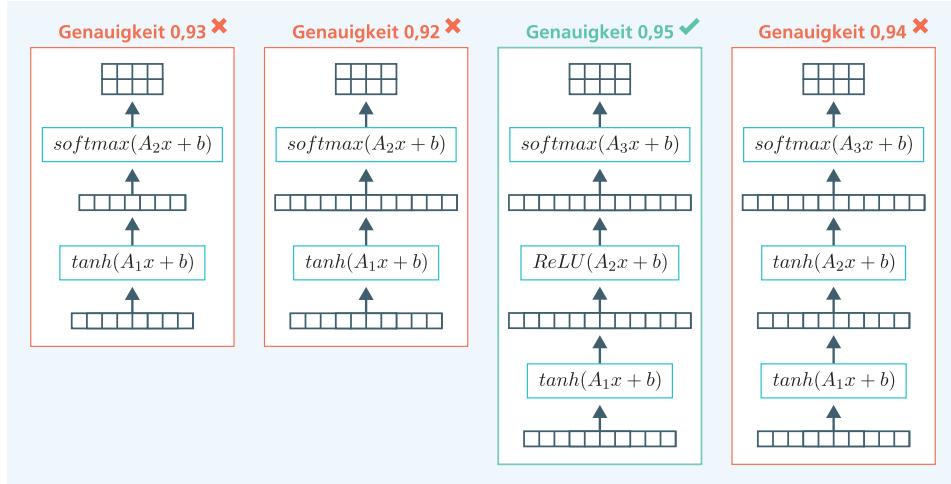
Es liegt nahe, die Suche nach der besten Architektur automatisch durchzuführen. Offenbar handelt es sich hierbei um ein Optimierungsproblem höherer Art. Dies kann kompliziert sein, da sich ja z. B. durch die Einführung einer weiteren Schicht im TNN die Anzahl der Stellwerte (Länge des verdeckten Vektors, Regularisierungs-Hyperparametern



**Abb. 4.31** Die Bayessche Optimierung berechnet aus den Genauigkeiten für die bisherigen evaluierten Stellwerte und den zugehörigen Genauigkeiten (•) eine Mittelwertkurve (blau) und Schwankungsbreiten (lila) für die Schätzung der Genauigkeit. Der nächste Stellwert wird dort ausgewählt, wo ein hoher Genauigkeitswert plausibel ist (rot gestrichelt). Die schwarze Linie symbolisiert die unbekannte tatsächliche Genauigkeit. Bildnachweis im Anhang

etc.) vergrößert. Zudem haben diese geschachtelten Optimierungen einen hohen Rechenzeitbedarf. Eine Reihe von Verfahren (Goodfellow et al. 2016, S. 420) wurden für diese Hyperparameter-Optimierung vorgeschlagen (Abb. 4.31):

- Die Gittersuche, d. h. die erschöpfende Berechnung der Performanz auf den Validierungsdaten für alle Merkmalskombinationen. Sie ist für mehr als zwei Stellwerte ineffizient.
- Bei der Zufallssuche wird die Kombination der Hyperparameter jedes Mal zufällig gewählt. Dies kann besser sein als die Gittersuche, insbesondere wenn die endgültige Modellgüte nur durch wenige Hyperparameter bestimmt wird, oder wenn viele Hyperparameter angepasst werden sollen.
- Die Bayessche Optimierung nimmt zufallsgesteuerte Modifikationen vor, konzentriert sich aber auf die potentiell besten Kombinationen der Hyperparameter. Hierbei wird die Performanz der bisher evaluierten Hyperparameter-Werte durch ein Modell beschrieben, z. B. einem Gaußschen Prozessmodell. Dieses Modell prognostiziert nicht nur die mittleren zu erwartenden Performanzwerte, sondern auch die Ungenauigkeit der Prognose. Der nächste zu evaluierende Hyperparameter wird nun in einem Bereich ausgesucht, in dem unter Berücksichtigung der Unsicherheit hohe Performanzwerte möglich sind. Damit verfolgt das Modell einen Ausgleich zwischen der Exploration der Hyperparameter und deren Auswertung (Konzentration auf die besten Kombinationen). In der Praxis erzielt die Bayessche Optimierung bessere Ergebnisse als die Gittersuche und die Zufallssuche.
- Man kann die Bayessche Optimierung auch auf den Fall übertragen, dass nicht nur kontinuierliche Parameter modifiziert, sondern auch diskrete Änderungen der Modellstruktur vorgenommen werden, z. B. Hinzufügung einer zusätzlichen Schicht in einem neuronalen Netz (Jin et al. 2019). Die Autoren haben dieses Verfahren implementiert und unter dem Namen AutoKeras als Modul des Keras-Toolkits zur Verfügung gestellt (Abb. 4.32).

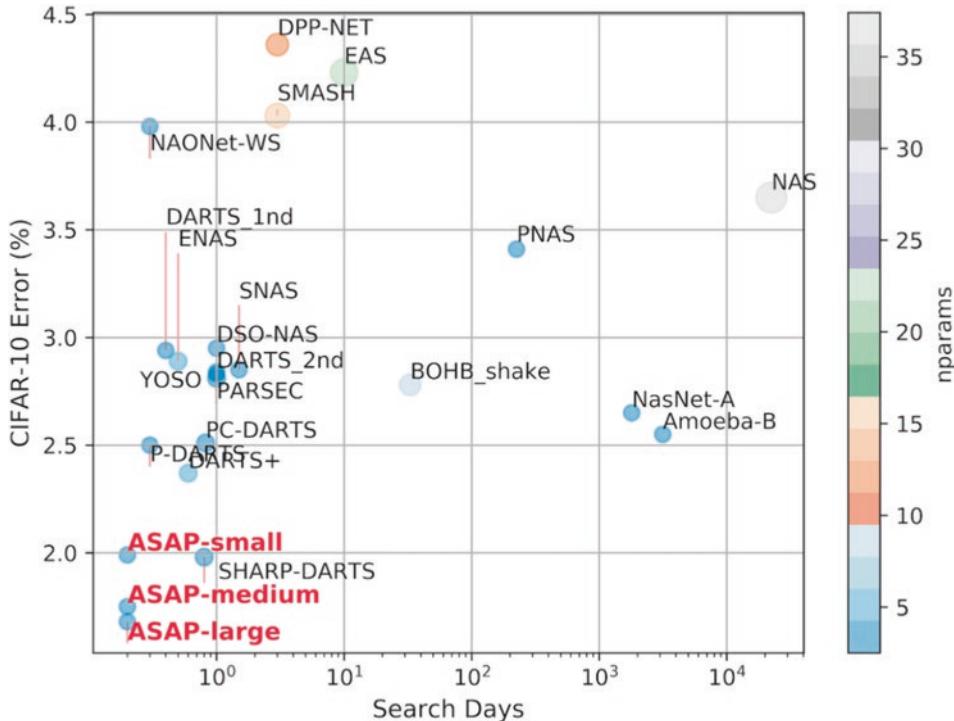


**Abb. 4.32** AutoKeras ist ein automatisches Verfahren, welches selbsttätig die optimale Netzwerkarchitektur und die Werte der Hyperparameter sucht. Dabei wird die Länge verdeckter Vektoren verändert, neue Schichten eingefügt, Aktivierungsfunktionen geändert usw. Ein Performanzmaß, z. B. Genauigkeit, dient als Auswahlkriterium

Google nutzt diese Techniken ausgiebig für die Entwicklung tiefer neuronaler Netze und stellt auch eine Möglichkeit zur Verfügung, mit der ein Nutzer seine neuronale Netzarchitektur optimieren kann (Google 2019).

Die bisherigen Methoden zur Suche einer optimalen Netzarchitektur erzielen gute Ergebnisse, benötigen aber einen sehr hohen Rechenaufwand (oft Tausende von GPU-Tagen). Hierbei wechseln sich normale Parameteroptimierungen ab mit diskreten Änderungen, z. B. Hinzufügung zusätzlicher Schichten oder Operatoren.

Ein alternativer Ansatz zur Hyperparameter-Optimierung konstruiert ein umfassendes differenzierbares Suchproblem, welches in Gänze durch Gradientenabstieg gelöst werden kann. Noy et al. (2020) führen zusätzlich ein sogenanntes „Abkühlen“ (Annealing) ein, welches aus der Optimierungsliteratur bekannt ist. Hierbei wird durch „Temperaturparameter“ die Gestalt der Kostenfunktion so modifiziert, dass kleine Gewichte „abgeschaltet“ werden können und sich ganze Netzwerkteile stilllegen lassen. Im Vergleich zu den bisherigen Ansätzen benötigt ihre ASAP-Methode (Architecture Search, Anneal and Prune) um Größenordnungen weniger Rechenzeit und verringert die Dauer der Architektursuche von GPU-Jahren auf wenige GPU-Stunden. Abb. 4.33 zeigt die Suchdauer und erzielte Genauigkeit für die CIFAR-10 Bildklassifikation mit 10 Klassen. Hier kann ASAP den Klassifikationsfehler von 2,55 % auf 1,99 % verbessern mit einem Rechenaufwand von nur 0,2 GPU-Tagen.

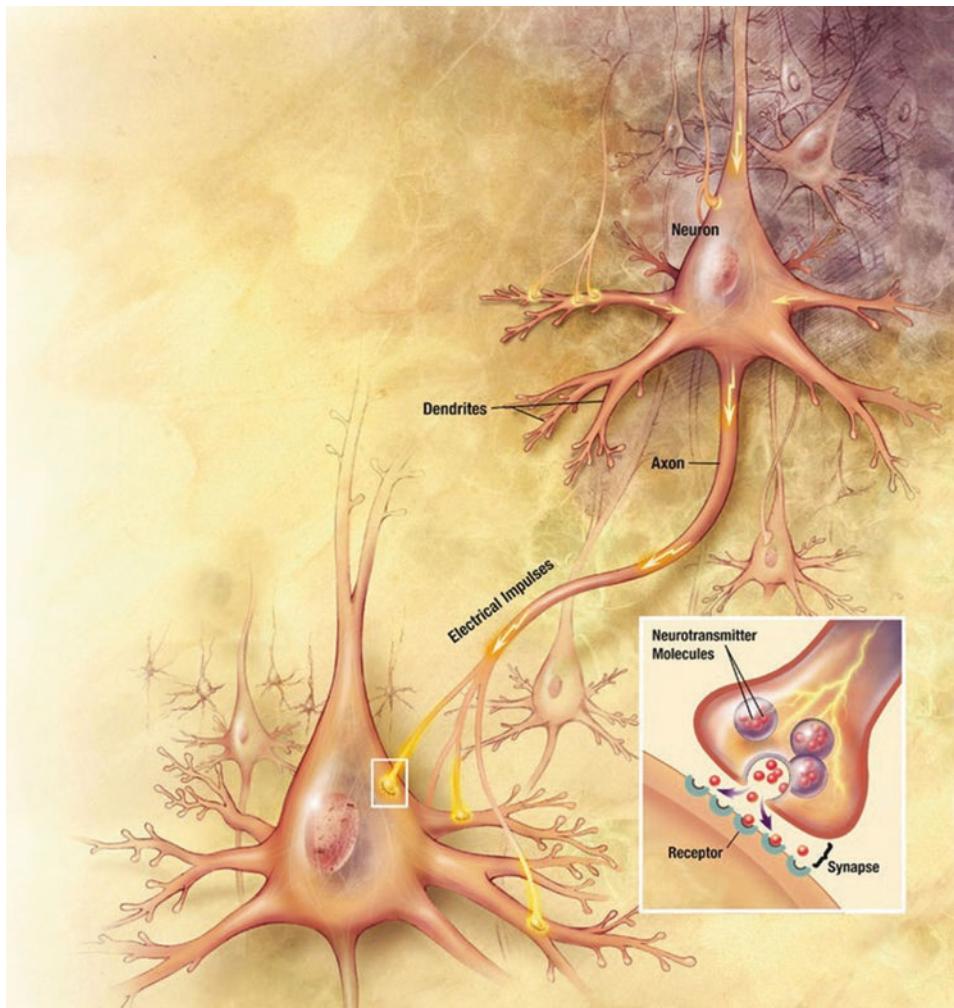


**Abb. 4.33** Suchdauer und Klassifikationsfehler verschiedener Architektursuchen für die CIFAR-10 Bildklassifikation. Die Suchkosten werden durch GPU-Tage gemessen. Die Größe der Kreise zeigt den Speicherplatzbedarf und die Farbe kodiert die Anzahl der Parameter. Das ASAP-Verfahren benötigt dramatisch weniger Rechenzeit und erzielt bessere Genauigkeitswerte. Noy et al. 2020

## 4.9 Biologische neuronale Netze funktionieren anders

Obwohl künstliche neuronale Netze von den biologischen Prozessen inspiriert wurden, die Wissenschaftler ab den 50er-Jahren im Gehirn beobachten konnten, unterscheiden sie sich in vielerlei Hinsicht von ihren biologischen Gegenstücken. In ähnlicher Weise haben Vögel moderne Flugzeuge und Pferde die Lokomotiven und Autos inspiriert, aber keines der heutigen Transportfahrzeuge ähnelt einem dieser lebenden Tiere.

Das Gehirn von Tieren und Menschen ist ein biologisches neuronales Netz und besteht aus Neuronen, speziellen Körperzellen (NeuronalesNetz 2019). Die Neuronen können über einen langen Fortsatz, das Axon, einen Impuls über eine Mischung elektrischer und chemischer Mechanismen auf andere Neuronen übertragen (Abb. 4.34). Die Ankopplung an andere Neuronen durch sogenannte Synapsen ist dabei gewichtet, d. h. von unterschiedlicher Intensität. Wenn die gewichtete Summe der Eingangssignale eines Neurons eine gewisse Schwelle überschreiten, so „feuert“ das Neuron und sendet selbst Impulse aus.



**Abb. 4.34** Ein Neuron sendet eine Reihe von Impulsen über das Axon. Dabei werden sowohl elektrische als auch chemische Übertragungsmechanismen genutzt, um die Informationen auf andere Neuronen zu übertragen. Die Verbindungen zu anderen Neuronen (Synapsen) sind gewichtet, also unterschiedlich „stark“. Wichtig ist, dass die Informationen auf dem Axon durch ihre Häufigkeit kodiert sind (Frequenzmodulation). Bildnachweis im Anhang

In den 60er-Jahren fanden Hubel und Wiesel heraus, dass das Sehsystem der Säugetiere in Schichten organisiert ist. Die Signale der Netzhaut im Auge werden durch unterschiedliche Schichten geschleust und dabei werden immer komplexere Bildeigenschaften extrahiert (Abschn. 5.1.2). Daraufhin wurde künstliche neuronale Netze entwickelt, welche aus mehreren Schichten von linearen Transformationen und anschließender Nichtlinearität bestehen. Die Nichtlinearitäten sind erforderlich, damit die Netze komplizierte Zusammenhänge zwischen Ein- und Ausgaben abbilden können (Abschn. 4.2). Als Trainingsalgorithmus wurde Backpropagation (Abschn. 4.4) vorgeschlagen.

Aus heutiger Sicht weisen biologische und künstliche neuronale Netze eine Reihe von Unterschieden auf:

- **Verbindungsmuster:** Biologische Neuronen sind nicht strikt in Schichten organisiert. Sie haben eine kleine Zahl von stark miteinander verknüpften Neuronen und viele wenige verbundene Neuronen.
- **Informationsübertragung:** die Impulsweiterleitung entlang des Axon eines Neurons ist frequenzmoduliert, seine Information wird also durch die Häufigkeit der Impulse kodiert. Dies ist eine substantiell andere Darstellung als die festen Werte in den Vektoren und Tensoren eines tiefen neuronalen Netzes.
- **Lernen:** Wir verstehen immer noch nicht genau, wie biologische Gehirne lernen. Hirnfasern wachsen und verbinden sich mit anderen Neuronen, Verbindungen können in ihrer Funktion verändert werden, und Synapsen können je nach Relevanz gestärkt oder geschwächt werden. Nervenzellen, die zusammen aktiv sind, bilden allmählich Verbindungen aus. Künstliche neuronale Netze hingegen haben derzeit meist eine feste Struktur von Verbindungen. Das Backpropagation-Verfahren zu Berechnung von Ableitungen ist biologisch unplausibel, insbesondere, weil die Synapsen Impulse nur in eine Richtung weiterleiten können. Allerdings werden mittlerweile biologisch realistischere Lernverfahren diskutiert (Mesnard et al. 2019).

Daher können tiefe neuronale Netze nur sehr beschränkt als Modell für die Informationsverarbeitung im Gehirn dienen.

---

## 4.10 Zusammenfassung und Trends

Bei komplexeren Problemen müssen die Modelle nichtlineare Trennflächen ausbilden, um die Prognose- und Klassifikationsaufgabe zu lösen. Dies ist nur möglich, wenn zusätzliche Schichten mit nichtlinearen Funktionen in das neuronale Netz eingefügt werden. Durch Training bilden sich in diesen Schichten immer komplexere Repräsentationen der Eingaben, mit deren Hilfe dann durch die letzte Schicht die gewünschten Ausgaben generiert werden können. Es gibt zwar auch nichtlineare klassische maschinelle Lernverfahren, aber deren Performanz wird bei umfangreichen Daten fast immer von TNN übertroffen.

In den letzten Jahren wurden Toolkits für TNN eingeführt, die eine einfache Formulierung der gängigen Modelle erlauben und zudem selbsttätig die Gradienten ausrechnen und die Optimierung durchführen. Darüber hinaus erlauben sie auch die Evaluation des Modells und die Ausführung der Berechnungen auf paralleler Hardware.

Wegen der vielen Stellmöglichkeiten muss die Struktur von TNN angepasst werden, um eine gute Prognose zu erreichen. Dies geschieht in einem schrittweisen Verfahren, wobei man die Güte des Modells auf einer gesonderten Validationsmenge überprüft. Wichtig ist dabei, die Unter- bzw. die Überanpassung der Modelle zu überprüfen. Unteranpassung tritt ein, wenn das Modell eine zu geringe Darstellungsmächtigkeit für die Pro-

gnoseaufgabe besitzt. Bei Überanpassung ist das Modell zu komplex und passt sich an zufällige Variationen in den Daten an. Es gibt eine Reihe von Regularisierungsverfahren zur Vermeidung von Überanpassung, etwa Dropout oder Batch-Normalisierung.

In den letzten Jahren haben sich eine Reihe wichtiger Netztypen herausgebildet, unter anderem Feedforward-Netze, Convolutional Neural Networks (CNN), rekurrente neuronale Netze (RNN), Netze zum Bestärkungslernen, generative adversariale Netze (GAN) und Autoencoder. Diese Netze werden in den weiteren Kapiteln diskutiert. Schließlich werden noch Methoden zur Suche von Netzstrukturen dargestellt. Während dies oft noch durch Trial-and-Error geschieht, wurden in der letzten Zeit automatische Verfahren hierzu vorgeschlagen, welche um Größenordnungen effizienter sind.

### Trends

- Zwischen den Deep-Learning Plattformen PyTorch und TensorFlow gibt es ein Wettrennen um die Gunst der Nutzer. TensorFlow punktet mit einer großen Anzahl von Beispielmodellen, und PyTorch holt auf mit einem flexibleren Sprachdesign.
- Die automatische Suche der Struktur durch AutoML-Techniken wird wichtig. Hierbei wird die Modellarchitektur mittlerweile durch Gradientenabstieg gefunden, was wesentlich weniger Rechenaufwand benötigt.
- Tiefe neuronale Netze haben häufig sehr viele Parameter. Um den damit verbundenen Speicherbedarf und die Rechnerleistung zu reduzieren (z. B. für Smartphones) kann man die Anzahl der Parameter reduzieren. (Frankle und Carbin 2019) verwenden dabei für das reduzierte Modell die vorherigen Initialisierungswerte und können so oft die Anzahl der Parameter ohne Genauigkeitsverlust auf 10 % verringern.

---

## Literatur

- Abadi, M., et al. 2016.** TensorFlow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265–283). 2016.
- Bassily, R., Belkin, M. und Ma, S. 2018.** On exponential convergence of sgd in non-convex over-parametrized learning. [Online] 2018. [Zitat vom: 10. 02. 2020.] arXiv preprint arXiv:1811.02564.
- Bruna, J. und Mallat, S. 2013.** Invariant scattering convolution networks. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 8, pp. 1872–1886, Aug 2013. 2013.
- Brutzkus, A. und Globerson, A. 2019.** Why do Larger Models Generalize Better? A theoretical Perspective via the XOR Problem. [Online] 2019. [Zitat vom: 25. 09. 2019.] arXiv preprint arXiv:1810.03037v2.
- Chollet, F. 2019.** Keras. [Online] 2019. [Zitat vom: 05. 03. 2019.] <https://keras.io/>.
- DL-Software. 2019.** DL-Software. *Comparison of deep-learning software*. [Online] 2019. [https://en.wikipedia.org/wiki/Comparison\\_of\\_deep-learning\\_software](https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software).
- Eldanand, R. und Shamir, O. 2016.** The power of depth for feedforward neural networks. *COLT*. 2016.

- Frankle, J. und Carbin, M.** 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *ICLR*. [Online] 2019. [Zitat vom: 27. 01. 2020.] <https://openreview.net/forum?id=Jl-b3RcF7>.
- Github.** 2019. Github. [Online] 2019. <https://github.com/>.
- Goodfellow, Ian, Bengio, Yoshua und Courville, Aaron.** 2016. *Deep Learning*. s.l. : MIT Press, 2016.
- Google.** 2019. AutoML. [Online] 2019. <https://cloud.google.com/vision/automl/docs/beginners-guide>.
- Hardt, M., Recht, B. und Singer, Y.** 2016. Train faster, generalize better: Stability of stochastic gradient descent. *ICML*. 2016.
- Hornik, K., Stinchcombe, M. und White, H.** 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366. 1989.
- Jin, H., Song, Q., & Hu, X.** 2019. Auto-keras: An efficient neural architecture search system. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1946–1956). ACM. 2019.
- Kaggle.** 2017. Kaggle 2017 Survey Results. [Online] 2017. <https://www.kaggle.com/amberthomas/kaggle-2017-survey-results>.
- Mesnard, T., et al.** 2019. Ghost Units Yield Biologically Plausible Backprop in Deep Neural Networks. [Online] 2019. [Zitat vom: 03. 01. 2020.] arXiv preprint arXiv:1911.08585.
- Minsky, M. und Papert, S.** 1969. *Perceptron: an introduction to computational geometry*. s.l. : The MIT Press, Cambridge, expanded edition, 19(88), 2., 1969.
- NeuronalesNetz.** 2019. Neuronales Netz. [Online] 2019. [Zitat vom: 25. 08. 2019.] [https://de.wikipedia.org/wiki/Neuronales\\_Netz](https://de.wikipedia.org/wiki/Neuronales_Netz).
- Neyshabur, B., Salakhutdinov, R. und Srebro, N.** 2015. Path-SGD: Path-normalized optimization in deep neural networks. *NIPS*. 2015.
- Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., ... & Zelnik, L.** 2020. Asap: Architecture search, anneal and prune. In International Conference on Artificial Intelligence and Statistics (pp. 493–503). PMLR.
- PyTorch.** 2019. PyTorch. [Online] Facebook, 22. 01. 2019. [Zitat vom: 05. 03. 2019.] <https://pytorch.org/>.
- Replication.** 2019. Replication crisis. [Online] 2019. [Zitat vom: 29. 05. 2019.] [https://en.wikipedia.org/wiki/Replication\\_crisis](https://en.wikipedia.org/wiki/Replication_crisis).
- Rosenblatt, F.** 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386. 1958.
- Rumelhart, D., Hinton, G. und Williams, R.** 1985. *Learning internal representations by error propagation*. (No. ICS-8506). California University San Diego LA Jolla Inst. for Cognitive Science. : s.n., 1985.
- Shalev-Shwartz, S. und Ben-David, S.** 2014. *Understanding machine learning: from theory to algorithms*. s.l. : Cambridge University Press, 2014.
- Shearer C.** 2000. The CRISP-DM model: the new blueprint for data mining, . *J Data Warehousing* (2000); 5:13–22. 2000.
- Tschannen, M., Bachem, O. und Lucic, M.** 2018. Recent advances in autoencoder-based representation learning. [Online] 2018. [Zitat vom: 13. 08. 2019.] arXiv preprint arXiv:1812.05069.
- Werbos, P.** 1974. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. s.l. : PhD thesis, Harvard University, Cambridge, MA, 1974.



# Bilderkennung mit tiefen neuronalen Netzen

5

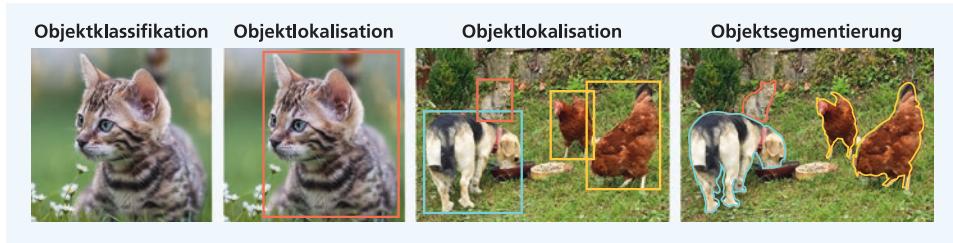
## 5.1 Was bedeutet eigentlich Bilderkennung?

In der Bilderkennung werden automatische Verfahren gesucht, mit denen man Objekte in einem Bild oder Foto identifizieren kann. Diese Methoden sollen also den Objekten ihren Namen (Klasse) zuordnen. Bildverstehen hat darüber hinaus eine weitergehende Beschreibung der in einem Bild dargestellten Szene als Ziel. In diesem Kapitel wird hauptsächlich die Bilderkennung diskutiert, während Aspekte des Bildverständens in den Abschn. 6.9 und 7.8 dargestellt werden.

### 5.1.1 Arten der Objekterkennung in Bildern

Es gibt unterschiedliche Arten von Bilderkennungsaufgaben (Abb. 5.1). Die Objektklassifikation in Bildern ermittelt in einem Bild den Namen (Klasse) eines darin enthaltenen Objektes, welches meist das wichtigste oder prominenteste ist. Ist man auch daran interessiert, an welcher Position im Bild das Objekt zu finden ist, so handelt es sich um die Aufgabe der Objektlokalisation in Bildern. Die Lage eines Objektes wird dabei mit einer Bounding-Box (Hüllrechteck) angegeben. Im Allgemeinen enthält ein Bild mehrere Objekte, von denen bei der Objektlokalisation die wichtigsten klassifiziert und durch Bounding-Boxen verortet werden. Weil die Positionsbestimmung durch Bounding-Boxen relativ ungenau ist, möchte man oft die Pixel des Bildes bestimmen, durch welche ein Objekt dargestellt ist. Diese Aufgabe heißt Objektsegmentierung.

Es gibt viele Anwendungsbereiche für die Bildklassifikation. Eine naheliegende Verwendung ist die Organisation der persönlichen Fotosammlung der Nutzer, z. B. durch Einordnung der Bilder nach inhaltlichen Kriterien oder zur Unterstützung der Bildersuche. Diese Techniken kann man auch für große Bilddatenbanken verwenden, z. B. im



**Abb. 5.1** Unterschiedliche Aufgaben der Bilderkennung: Die Objektklassifikation weist das „wichtigste“ Objekt des Bildes einer Klasse zu. Die Objektlokalisation markiert die Position eines Objektes durch eine Bounding Box. Die Objektsegmentierung identifiziert die Pixel eines Objektes. Bildnachweis im Anhang

medizinischen Bereich. Eine weitere Anwendung ist die Suche von jugendgefährdenden Abbildungen in sozialen Medien. Schließlich ist die Bilderkennung eine grundlegende Technologie für selbstfahrende Autos.

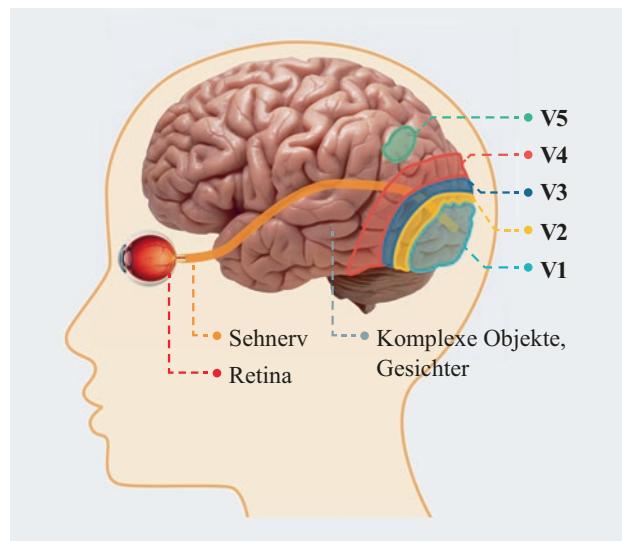
### 5.1.2 Inspirationen aus der Biologie

Schon die griechischen Philosophen versuchten, den Vorgang des menschlichen Sehens zu ergründen. Aristoteles meinte, dass Licht von Gegenständen reflektiert wird, und wenn dieses reflektierte Licht auf das Auge trifft, so „findet Sehen statt“ (Krämer 2016). Johannes Kepler stellte fest, dass das Licht auf die Augenlinse trifft. Es werde dort so gebrochen und gebündelt, dass auf der Netzhaut ein umgekehrtes verkleinertes Abbild der Außenwelt erscheine, das dann vom Verstand weiterbearbeitet werde. Aber es war völlig unklar, wie die Erkennung von Gegenständen durch den Verstand ablaufen könnte.

Erst Hubel und Wiesel konnten die Funktionsweise des Sehsystems mithilfe von Messungen an einzelnen Neuronen von Katzen und Affen aufdecken (Hubel und Wiesel 1962). Sie fanden heraus, dass es im V1-Areal des Gehirns „einfache“ Neuronen gibt, welche nur auf Merkmale reagieren, die sich an einer bestimmten Position auf der Netzhaut befinden. Ein solches Merkmal kann z. B. ein Lichtbalken in einer bestimmten Richtung sein. Damit war nachgewiesen, dass es im Gehirn lokale Detektoren von Kanten gibt, die nur kleine, umgrenzte Gebiete der Bildrezeptoren in der Netzhaut betreffen. Zudem gibt es „komplexe“ Neuronen, die die Ausgabe der lokalen Detektoren in einer kleinen Umgebung zusammenfassen. In den Arealen V2, V4, usw. des Gehirns werden die gefundenen Informationen aus V1 weiterverarbeitet und immer komplexere Objekte, z. B. Gesichter, erkannt. Dies wird in Abb. 5.2 dargestellt und in (Visual 2020) detailliert beschrieben. Hubel und Wiesel erhielten für ihre Forschungen den Nobelpreis.

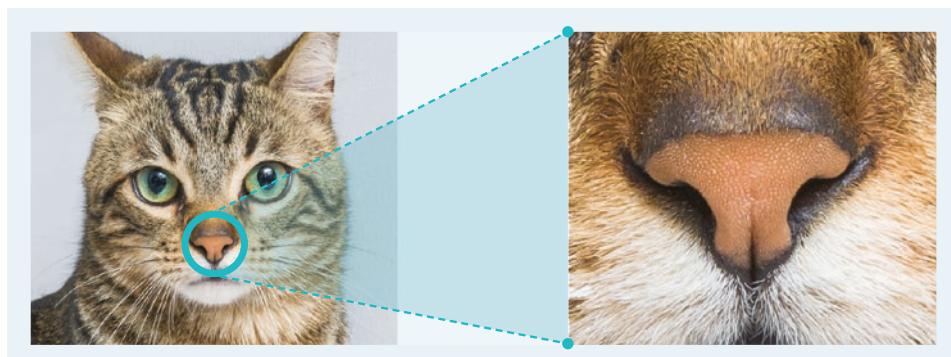
Beispielsweise geht die Erkennung einer Katze in einem Bild nicht in einem Schritt vonstatten, sondern es werden zunächst kleine Teilmuster in einer kleinen Region eines Bildes erkannt, z. B. die Nase einer Katze (Abb. 5.3). Die Nase einer anderen Katze wird ähnlich aussehen und außerdem kann ein kleines Teilmuster mit geringem Aufwand identifiziert werden. Da eine Katzennase fast überall in einem Bild auftauchen kann, muss der

**Abb. 5.2** Die Bildinformationen des Auges werden von der Retina zum Bereich V1 im visuellen Cortex übermittelt, wo einfache Formen, Ecken und Linien erkannt werden. In den Bereichen V2, V3, V4 und V5 werden mittlere visuelle Formen und Merkmalsgruppen identifiziert und in benachbarten Hirnregionen komplexe Objekte und Gesichter.  
Bildnachweis im Anhang

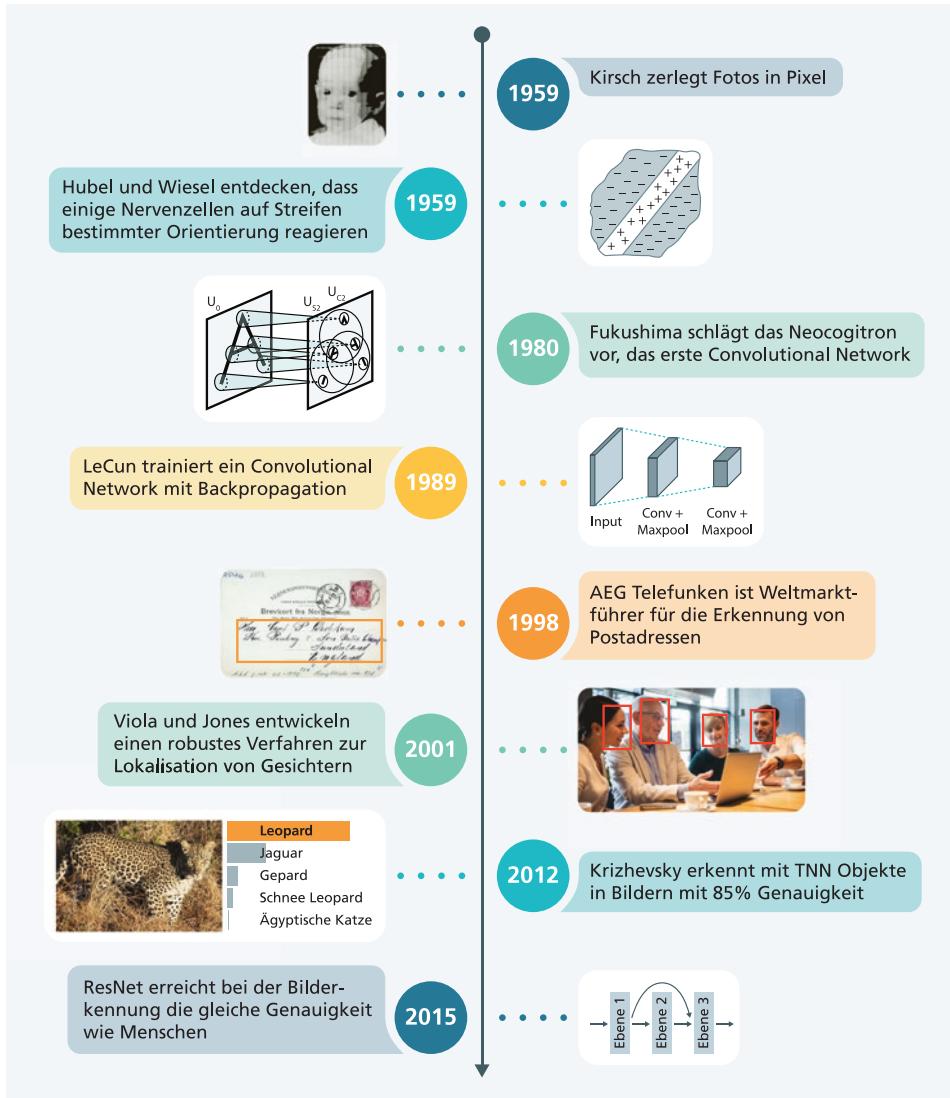


Katzennasen-Detektor über jede Teilregion des Bildes verschoben werden können. Entscheidend ist, dass die Detektoren für die Teilmuster nicht vorprogrammiert sind, sondern automatisch durch visuelle Eindrücke trainiert werden. Hubel und Wiesel konnten z. B. zeigen, dass der visuelle Kortex einer Katze irreversibel gestört wird, wenn einem Kätzchen zu Beginn seines Lebens die normale visuelle Erfahrung entzogen wird. Schließlich wird durch das Zusammenspiel vieler Einzeldetektoren die Katze als Ganzes erkannt und z. B. von einem Hund unterschieden.

Ein erstes neuronales Netz, welches diese Erkenntnisse aufgriff, wurde 1980 vorgestellt (Fukushima 1980). Dieses Neocognitron enthielt Schichten, welche die Eingabe in viele Teilbereiche einteilten und jeden dieser Teilbereiche auf die gleiche Weise verarbeiteten. Andere Schichten führten eine lokale Mittelwertbildung über viele kleine Teilberei-



**Abb. 5.3** Der erste Schritt zur Erkennung eines großen Objektes (Katze) kann anhand kleiner Muster (z. B. Katzennase) durchgeführt werden. Bildnachweis im Anhang



**Abb. 5.4** Die Geschichte der automatischen Bilderkennung. Bildnachweis im Anhang

che durch, das sogenannte Pooling. Diese Netzsichten wurden mehrfach übereinander angeordnet. Aber erst um die Jahrtausendwende gelang es, leistungsfähige Netze mit dieser Struktur erfolgreich einzusetzen, die Convolutional Neural Networks (LeCun et al. 1998). Abb. 5.4 zeigt die Geschichte der automatischen Bilderkennung.



**Abb. 5.5** Variationen in den Bildern von Katzen (links), welche eine Objektklassifikation erschweren: Wechsel der Perspektive, geänderte Beleuchtung, Formänderung, Verdeckung und unruhiger Hintergrund. Innerhalb einer Klasse von Objekten in ImageNet gibt es starke Unterschiede, hier z. B. bei der Klasse „Brücke“ (rechts). Bildnachweis im Anhang

### 5.1.3 Warum ist eine Bilderkennung schwierig?

Bei der Objektklassifikation in Bildern geht es darum, die Objekte innerhalb eines Bildes einer vorgegebenen Anzahl von Klassen zuzuordnen, z. B. Hund, Katze, Lastwagen, Flugzeug usw.

Dabei gibt es eine Reihe von Effekten, die eine Klassifikation erschweren können (Abb. 5.5 links). Schon eine Änderung der Entfernung zum Objekt oder die Spiegelung bzw. Drehung des Objektes kann das Bild stark verändern. Unterschiede in der Beleuchtung modifizieren oder unterdrücken Konturen des Objektes. Objekte können ihre Form ändern und Verdeckungen lassen Teile des Objektes verschwinden. Schließlich kann je nach Umgebung das Objekt nur noch schwierig vom Hintergrund unterschieden werden. Darüber hinaus sind auch oft Objekte unterschiedlicher Farben und Formen in einer Klasse zusammengefasst, z. B. Brücken in Abb. 5.5 rechts.

---

## 5.2 Die Bestandteile eines Convolutional Neural Networks

### 5.2.1 Der Kernel einer Convolution-Schicht analysiert kleine Bildbereiche

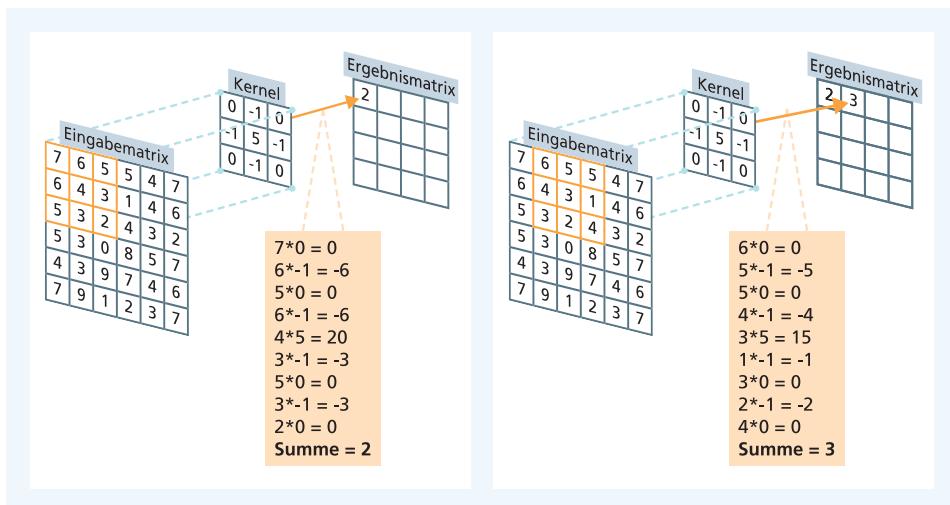
Ein Convolutional Neural Network (CNN, deutsch auch Faltungsnetz) ist vom Sehsystem der Säugetiere inspiriert. Es besteht aus einer oder mehreren Convolution-Schichten, gefolgt von jeweils einer Pooling-Schicht. Diese Abfolge von Schichten kann sich wiederholen. Bei der Bildverarbeitung sind die Eingaben immer mindestens zweidimensional, z. B. die Pixelmatrix eines Grauwertbildes. Die wichtigste Anwendung von CNNs ist die Objektklassifikation in Bildern mit dem Ziel, unterschiedliche Klassen von Objekten (Hund, Katze, Auto, ...) in diesen Bildern automatisch zu identifizieren. Diese Identifika-

tion der Objekte wird anhand von umfangreichen Trainingsdaten gelernt, welche zu jedem Bild die Klassennamen der darin enthaltenen Objekte beinhalten.

Dabei hat sich gezeigt, dass die besten Ergebnisse erzielt werden, wenn man das Bild in kleine überlappende Teilbereiche zerlegt und diese zunächst für sich betrachtet. Der Kernel einer Convolution-Schicht ist eine kleine Matrix von Parametern. Sie hat typischerweise die Ausmaße  $3 \times 3$  oder  $5 \times 5$ . Eine Eingabematrix zusammen mit einem Kernel ist in Abb. 5.6 dargestellt. Der Übersichtlichkeit halber wurden die Eingabematrix und der Kernel mit ganzen Zahlen gefüllt; normalerweise sind dies reelle Zahlen.

Der Kernel wird zu Beginn in die obere linke Ecke der Eingabematrix gelegt (rot eingezzeichnet in Abb. 5.6 links). Jeder Wert in der roten Teilmatrix – des rezeptiven Feldes – wird nun mit dem entsprechenden Wert des Kernels multipliziert und das Ergebnis addiert (siehe Nebenrechnung weiter rechts). Das Ergebnis 2 wird nun in die obere linke Ecke der Ergebnismatrix gespeichert. Der zweite Schritt zeigt den gleichen Vorgang für eine verschobene Position des Kernels (Abb. 5.6 rechts).

Für den zweiten Berechnungsschritt der Convolution-Schicht wird das rezeptive Feld in der Eingabematrix um eine Position nach rechts verschoben und die Berechnung mit dem gleichen Kernel wiederholt (Abb. 5.6 rechts). Das Ergebnis wird eine Position weiter rechts in die Ergebnismatrix gespeichert. Dieser Vorgang wird für alle Positionen der Eingabematrix wiederholt, solange bis die Ergebnismatrix gefüllt ist. Wie man sieht, ist die Ergebnismatrix etwas kleiner als die Eingabematrix. In der Regel wird zu jedem Wert der Ergebnismatrix noch eine Zahl  $b$  addiert, welche das Ergebnis unabhängig von den Eingabewerten erhöht bzw. erniedrigt. Dieser Bias wurde hier der Einfachheit halber weggelassen.



**Abb. 5.6** Erster Berechnungsschritt (links) und zweiter Berechnungsschritt (rechts) in der Convolution-Schicht für jeweils einen verschobenen kleinen Bereich der Eingabematrix. Dabei wird der Kernel sukzessive über die gesamte Eingabematrix „geschoben“ und die Ergebnismatrix gefüllt

Wichtig ist, dass ein Wert in der Ergebnismatrix allein von den Pixelwerten im jeweiligen rezeptiven Feld abhängt. Weiterhin ändern sich die Parameterwerte eines Kernels beim Verschieben nicht, die von einem Kernel extrahierten Merkmale sind also translationsinvariant.

Je nach den Werten in der Kernelmatrix werden unterschiedliche lokale Merkmale aus

der Eingabematrix extrahiert. Die Kernelmatrix  $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$  beispielsweise heißt diskreter

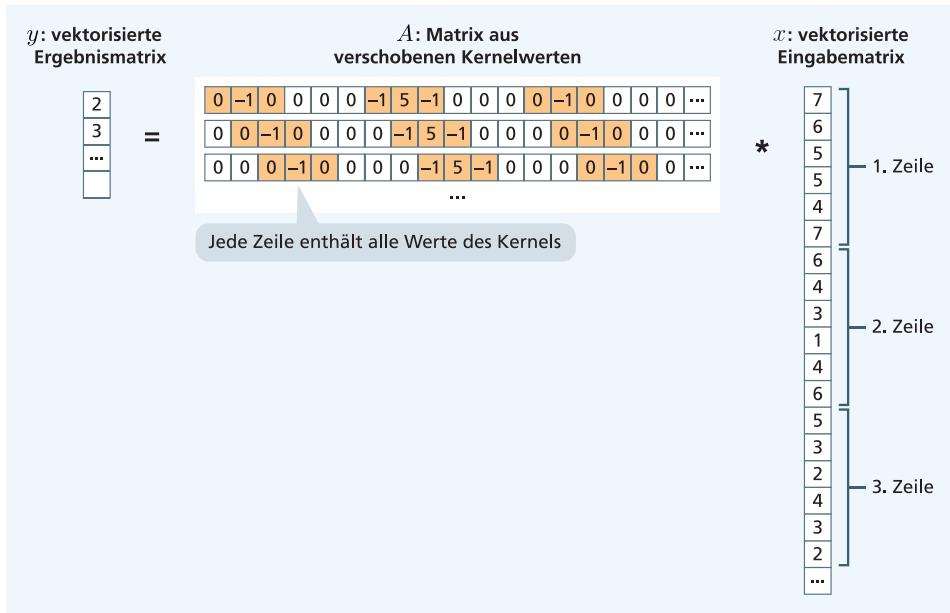
Laplace-Operator und erzeugt hohe Werte an einer Kante, also an einem Übergang zwischen Hell und Dunkel. Abb. 5.7 zeigt für eine Eingabematrix die Ergebnismatrix für diesen Kernel.

Man kann auch einen Kernel nicht nur um 1 Feld verschieben, sondern auch z. B. um 2 Positionen. Ist diese Schrittweite (engl. stride) größer als 1, so verringert sich natürlich entsprechend die Größe der Ergebnismatrix.

Die Eingabematrix kann auch zeilenweise als Vektor  $x$  geschrieben werden. Setzt man die Werte des Kernels an bestimmte Stellen eines Vektors  $a_1$ , so liefert das Skalarprodukt  $a_1 * x$  den ersten Wert der vektorisierten Ergebnismatrix. Abb. 5.8 zeigt, wie man dies für alle Werte der vektorisierten Ergebnismatrix fortsetzen kann und aus den  $a$ -Vektoren eine Matrix  $A$  bilden kann. Diese Berechnung liefert das gleiche Ergebnis wie die zugehörige Convolution-Schicht. Eine Convolution-Schicht ist also eine lineare Transformation  $y = A * x + b$ , bei der die Transformationsmatrix  $A$  nur sehr wenige von Null verschiedene



**Abb. 5.7** Eingabebild (links) und Ergebniswerte in Grauwertdarstellung für den Laplace-Kernel. Bildnachweis im Anhang



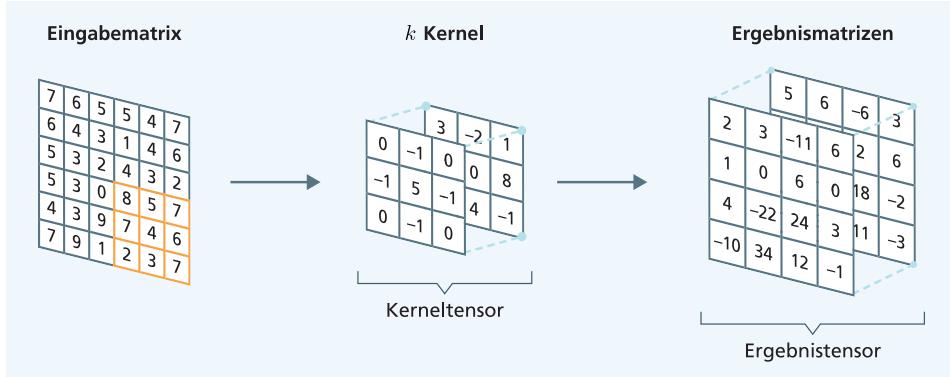
**Abb. 5.8** Man kann die Eingabematrix aus der Convolution-Schicht in Abb. 5.6 zeilenweise als Vektor anordnen (rechts). Weiter kann man eine Matrix  $A$  bilden, in der in jeder Zeile die Elemente des Kernels an unterschiedlichen Positionen auftauchen (Mitte). Diese Matrixmultiplikation führt zum gleichen Ergebnis wie die Convolution. Damit entspricht eine Convolution-Schicht mit einem Kernel einer linearen Transformation  $y = A * x + b$

Werte hat. Zudem tauchen alle neun Parameter des Kernels in jeder Zeile von  $A$  auf. Eine solche Mehrfachverwendung von Parametern nennt man auch Parameter-Sharing. Hierdurch wird die Gefahr der Überanpassung reduziert.

### 5.2.2 Convolution-Schicht mit vielen Kernen reagiert auf viele Merkmale

In einer Convolution-Schicht werden meist eine Anzahl  $k$  von Kernen (z. B.  $k = 128$ ) verwendet und damit  $k$  Ergebnismatrizen erzeugt, die zu einem Tensor zusammengefasst werden. Dies wird in Abb. 5.9 dargestellt. Hierdurch können in einer Schicht parallel sehr viele komplementäre Merkmale berechnet werden, welche zu einer genaueren Klassifikation des Bildes genutzt werden können.

Die Kernel des CNN sind kleine Matrizen von Parametern. Sie werden also nicht vorgegeben, sondern trainiert, also automatisch so angepasst, sodass das resultierende Netz das gewünschte Endergebnis (z. B. Klassennamen des im Bild enthaltenen Objektes) prognostizieren kann. Die Anzahl der Parameter eines Kernels ist sehr klein, z. B. 9 Parameter bei einem  $3 \times 3$  Kernel. Selbst wenn 128 verschiedene Kernel verwendet werden, so ist die Anzahl der Parameter ( $9 * 128$ ) noch wesentlich geringer als bei einer vollverbundenen Schicht. Die hätte z. B. bei einem Bild mit 1000 Pixeln insgesamt etwa 1 Million Parameter.



**Abb. 5.9** Eine Convolution-Schicht enthält  $k$  Kernel und  $k$  Ergebnismatrizen, welche jeweils zu Tensoren zusammengefasst werden

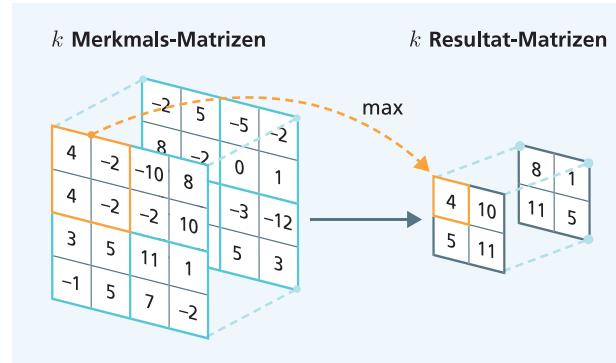
Die Ergebnismatrizen werden anschließend noch mit einer nichtlinearen Aktivierungsfunktion transformiert. Bei CNNs ist dies üblicherweise die abgeschnittene lineare Funktion (ReLU, Abb. 4.4).

### 5.2.3 Die Pooling-Schicht wählt die wichtigsten Merkmale aus

Hat man  $k = 128$  Kernel, so enthalten die Ergebnismatrizen zusammen wesentlich mehr Merkmalswerte als das eingegebene Pixelbild. Daher muss die Anzahl der Merkmalswerte durch eine Pooling-Schicht reduziert werden. Hierzu werden mehrere benachbarte Merkmale zusammengefasst. Dies reflektiert die Tatsache, dass die exakte Position eines Merkmals nicht so wichtig ist, die ungefähre Lage reicht aus.

Beim Max-Pooling werden für jede Merkmalsmatrix quadratische Teilbereiche, z. B.  $2 \times 2$  Quadrate, durch eine Zahl, das Maximum, zusammengefasst und in eine Resultatmatrix gespeichert (Abb. 5.10). Die anderen Werte des Bereichs werden ignoriert. Welches Merkmal als Maximum gewählt wird, hängt natürlich vom jeweiligen Eingabebild ab. Diese Aggregation kann für jede Merkmalsmatrix wiederholt werden und eine weitere Resultatmatrix produzieren. Hierdurch ergibt sich z. B. für  $2 \times 2$  Quadrate eine Datenreduktion auf ein Viertel. Es kann aber auch eine Pooling-Operation über alle Werte der Resultatmatrizen an einer Position erfolgen mit einer entsprechend höheren Datenreduktion. Insgesamt hat dieses Vorgehen drei wichtige Vorteile:

- Die Bildmerkmale werden näherungsweise invariant gegenüber kleinen Verschiebungen des Eingabebildes, da das Max-Pooling nur die Maxima unterschiedlicher Eingabepositionen abbildet.
- Hat die erste und die zweite CNN-Schicht jeweils einen  $5 \times 5$ -Kernel, so werden die Merkmale der zweiten Schicht von den Pixeln eines  $9 \times 9$ -Pixelbereichs beeinflusst. Die rezeptiven Felder werden also größer, wobei das Pooling eine zusätzliche Vergrößerung bewirkt. Damit können größere Objektmerkmale erfasst werden.



**Abb. 5.10** In einer Pooling-Schicht werden die Merkmale der Ergebnismatrizen der Convolution-Schicht lokal zusammengefasst. Dabei wird für kleine Bereiche, z. B.  $2 \times 2$  Felder, eine Zusammenfassungsfunktion ausgerechnet. In der Regel wird eine Maximierung durchgeführt. Dies geschieht meist getrennt für jede Ergebnismatrix

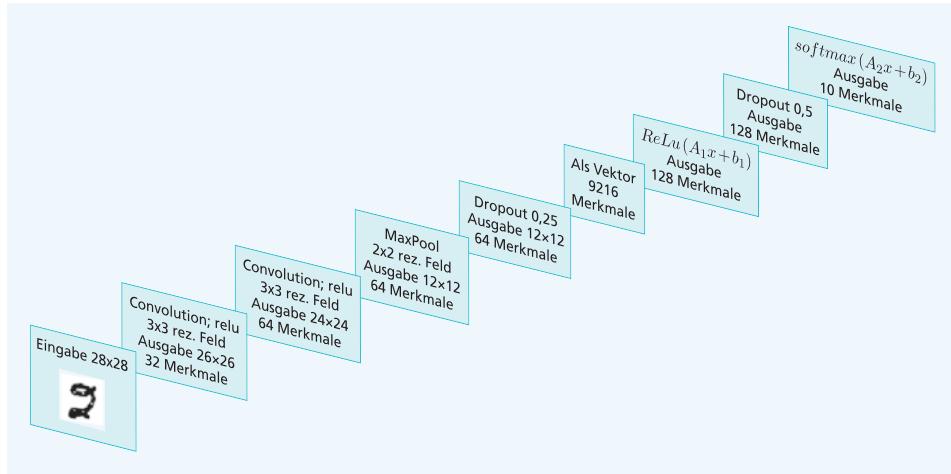
- Die erforderliche Rechenleistung und der Speicherbedarf werden verringert.

Die Erfahrung hat gezeigt, dass das Pooling durch Durchschnittsbildung zu schlechten Klassifikationsergebnissen führt als Bildung des Maximums.

### 5.3 Ein einfaches Convolutional Neural Network für die Ziffernerkennung

Für unseren MNIST-Datensatz zur Ziffernerkennung (Abschn. 3.2.3) wurde ein CNN vorgeschlagen (Keras 2019), dessen Schichten in Abb. 5.11 dargestellt werden. Es verwendet zu Beginn zwei Convolution-Schichten mit rezeptiven Feldern der Größe  $3 \times 3$  und 32 bzw. 64 Kerneln sowie einer ReLU-Aktivierungsfunktion. Es ergibt sich ein Bildbereich der Größe  $24 \times 24$  mit 64 Merkmalen. Eine MaxPool-Schicht berechnet jeweils in einem rezeptiven Feld von  $2 \times 2$  das Maximum. Dies ergibt einen Bildbereich von  $12 \times 12$  mit 64 Merkmalen. Darauf wird eine Dropout-Schicht zur Regularisierung eingefügt, die zufällig 25 % der Elemente weglässt. Das Ergebnis wird in einen Vektor mit 9216 Merkmalen umgewandelt. Hierauf wird eine lineare Transformation mit ReLU-Aktivierung und anschließend eine weitere Dropout-Schicht eingefügt, die 50 % der Elemente weglässt. Zum Schluss gibt es ein logistisches Regressionsmodell mit Softmax-Aktivierung, welche als Ausgabe den Vektor der Wahrscheinlichkeiten für die einzelnen Klassen (d. h. 10 Ziffern) liefert.

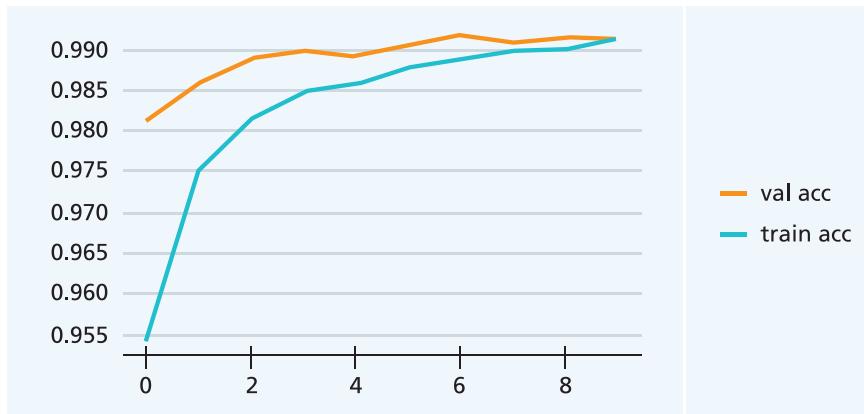
Wir haben in Abb. 5.8 gesehen, dass eine Convolution-Schicht einer linearen Transformation entspricht, in der die Parameter eines Kernels sehr häufig wiederholt werden. Dieses Parameter-Sharing ist der wesentliche Unterschied, der beim Training mit CNN beachtet werden muss. Ansonsten verläuft das Training eines solchen Netzes im Prinzip ab wie bei einem mehrschichtigen Feedforward-Netz. Als Verlustfunktion wird wieder die



**Abb. 5.11** Schichten des CNN zur MNIST-Ziffernerkennung. Unten links ist als Eingabe die  $28 \times 28$ -Pixelmatrix abgebildet. Als Ausgabe jeder Schicht ist der Bildbereich und die Anzahl der Merkmale aufgeführt. Ausgabe der letzten Schicht ist der Wahrscheinlichkeitsvektor der möglichen Ziffern

negative und logarithmierte Wahrscheinlichkeit eines Minibatches verwendet (Abb. 3.28). Das Modell wird mit dem stochastischen Gradientenabstieg trainiert.

Wie Abb. 5.12 zeigt, steigt beim Training die Genauigkeit auf der Validationsmenge schnell an und erreicht nach 10 Epochen einen Wert von 99,13 %. Damit entspricht der Fehler nur noch etwa einem Drittel des Fehlers des zweischichtigen neuronalen Netzes in Abb. 4.8 mit 97,5 %. Dies demonstriert die Effektivität der Convolution-Schichten für die Klassifikation von Bildern. Wichtig ist, dass bei der Anwendung des trainierten CNN die Dropout-Wahrscheinlichkeit auf 0,0 gesetzt wird, d. h. es werden keine Gewichte mehr auf null gesetzt.



**Abb. 5.12** Verlauf der Genauigkeit für die MNIST-Ziffernklassifikation mit dem CNN auf der Trainings- und der Validationsmenge



**Abb. 5.13** Für diese Eingaben haben bei MNIST die Kernel der ersten Schicht (links) und der zweiten Schicht (rechts) die höchsten Werte. Es werden dabei die jeweils 24 ersten Kernel betrachtet. Diese Merkmale sind offenbar bei der Klassifikation der Ziffern besonders effektiv

Abb. 5.13 zeigt die Mittelwerte derjenigen Eingaben, für welche die Kernel der ersten und zweiten Schicht die höchsten Werte erreicht. Dabei haben die Kernel der ersten Schicht ein rezeptives Feld der Größe  $3 \times 3$  und erzeugen nach dem Training 32 Merkmale. Es handelt sich hierbei meist um horizontale oder vertikale Kanten. Die Kernel der zweiten Schicht haben ein rezeptives Feld der Größe  $5 \times 5$  und generieren nach dem Training 64 Merkmale, auf denen die endgültige Klassifikation der Ziffern beruht. Wir beobachten hier vertikale, horizontale und diagonale Linien und Kanten sowie anders geartete Teilmuster, welche offenbar für die Ziffernklassifikation besonders aussagekräftig sind. Yann LeCun in Abb. 5.14 hatte entscheidenden Anteil an der Entwicklung der CNN.

## 5.4 Der ImageNet Wettbewerb befeuert die Methodenentwicklung

Wie schon in Abschn. 2.1 erwähnt, wird seit 2010 jährlich ein Wettbewerb veranstaltet, in dem unterschiedliche Forschungsgruppen ihre Bilderkennungssoftware auf die ImageNet-Daten (Russakovsky et al. 2015) anwenden können. Es gibt 1000 Objektkategorien sowie



**Yann LeCun**

studierte Informatik in Paris. 1987 war er als PostDoc bei Geoffrey Hinton in Toronto. Anschließend ging er zu den Bell-Labs und entwickelte dort Lernverfahren für Convolutional Neural Networks (CNN). Als Regulierungsmethode schlug er „optimal brain damage“ vor. Er baute probabilistische Netze, welche er auf Handschrifterkennung und Optical Character Recognition (OCR) anwendete. Das von ihm mitentwickelte Bankscheck-Lesegerät las zeitweise über 10 % der Schecks in den USA. Er entwickelte stochastische Versionen von Gradientenverfahren. Sein LeNet Modell zur Erkennung von Ziffern wurde Prototyp für alle weiteren CNN zur Bilderkennung. 2013 wurde er der erste Direktor von Facebook AI Research. 2019 erhielt er zusammen mit Yoshua Bengio und Geoffrey Hinton den Turing Award.

**Abb. 5.14** Yann LeCun \* 08.07.1960 Soisy-sous-Montmorency, Frankreich. Bildnachweis im Anhang



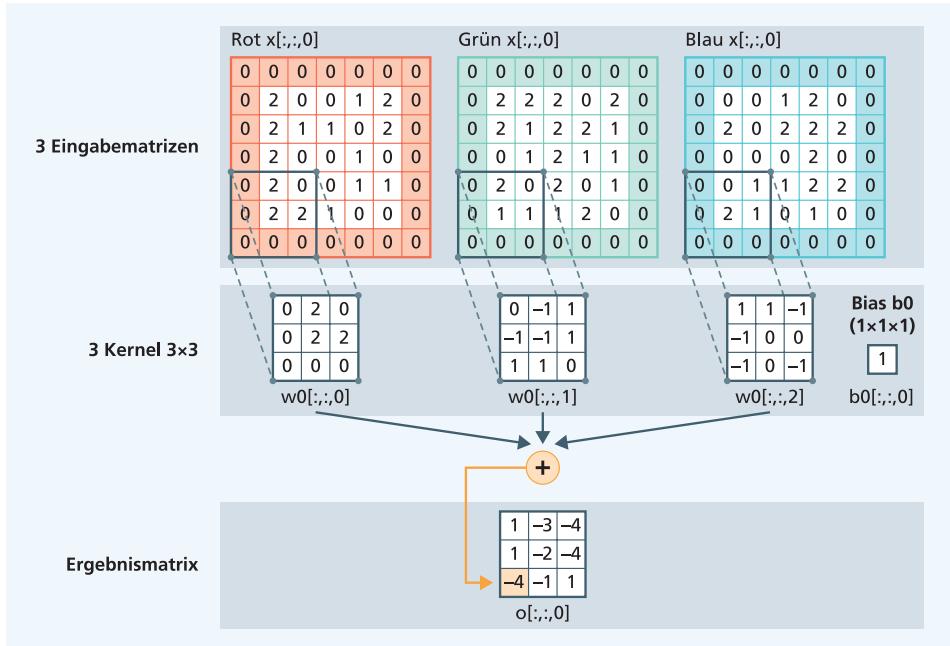
**Abb. 5.15** Die Bilder der ImageNet Sammlung sind mit 1000 verschiedenen Klassen annotiert. Wu et al. (2018) haben für vier Klassen Bilder gemäß den Klassifikationsergebnissen als „einfach“ oder „schwierig“ zu klassifizieren eingeordnet. Bildnachweis im Anhang

eine Trainingsmenge von etwa 1,2 Millionen Bildern und eine Validationsmenge von fünfzigtausend Bildern. Abb. 5.15 zeigt Beispielbilder aus vier repräsentativen Klassen. Wu et al. (2018) haben die Bilder analysiert und sie nach den Klassifikationsergebnissen als „einfach“ oder „schwierig“ eingeordnet. Bei den schwierigen Bildern sind die Objekte zum Beispiel sehr klein, ungewöhnlich gefärbt oder in einer komplexen Beleuchtungssituation. Die Klassifizierung ist relativ fein, es gibt z. B. 120 Hundekategorien. Schließlich gibt es eine den Teilnehmern des Wettbewerbes nicht bekannte Testmenge von hunderttausend Bildern. Für jede Kategorie sind mindestens 500 Trainingsbilder vorhanden. Dadurch ist die Variation innerhalb der Klassen gut abgedeckt.

Aus Kostengründen wurden nicht immer alle Objekte in einem Bild manuell klassifiziert. Daher wurde die Qualität der Bildklassifikation mit der Top5-Fehlerrate bewertet, welche misst, ob in der Testmenge die annotierte „wahre“ Objektkategorie eine der ersten fünf prognostizierten Kategorien ist.

Die Bilder wurden auf eine Größe von  $256 \times 256$  Pixeln reduziert. Da es sich um Farbbilder handelt, gibt es für jedes Pixel 3 Werte: den Rot-, Grün- und Blauwert, die jeweils in einer Matrix zusammengefasst wurden. Diese drei Matrizen bilden die Eingabe für die Bildklassifikation.

Um die drei Farbwertmatrizen zueinander in Relation zu setzen, wird die erste Convolution-Schicht des CNN abgeändert. Es werden drei Kernel definiert, die separat auf die Eingabematrizen angewendet werden. Das Ergebnis wird summiert und in eine Ergebnismatrix gespeichert. In Abb. 5.16 wird diese Berechnung dargestellt. Hierbei ist zu be-



**Abb. 5.16** Wird diese Berechnung dargestellt. Hierbei ist zu berücksichtigen, dass die Kernel jeweils um zwei Positionen verschoben werden (Schrittweite 2). Zudem werden die Felder außerhalb der Eingabematrizen mit dem Wert 0 berücksichtigt (Padding)

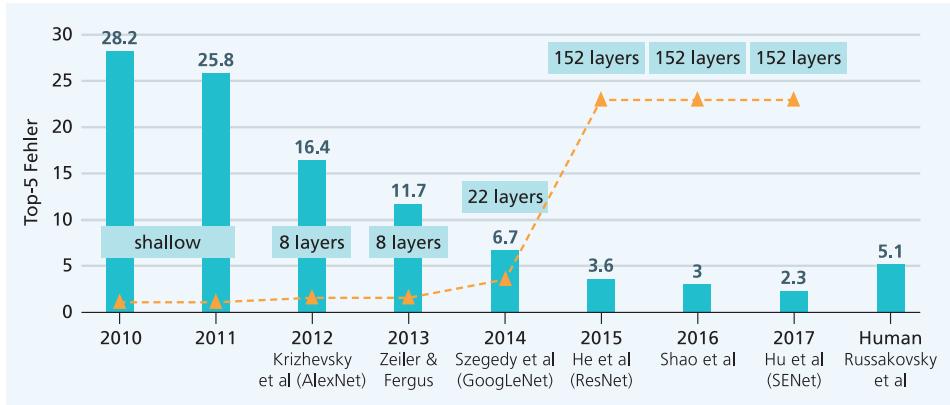
rücksichtigen, dass die Kernel jeweils um zwei Positionen verschoben werden (Schrittweite 2). Zudem werden die Felder außerhalb der Eingabematrizen mit dem Wert 0 berücksichtigt (Padding).

## 5.5 Fortschrittliche Convolutional Neural Networks

### 5.5.1 AlexNet nutzt erfolgreich GPUs zum Training

Das Jahr 2012 brachte den Durchbruch der TNN im ImageNet-Klassifikationswettbewerb. Das AlexNet (Krizhevsky et al. 2012) nutzt insgesamt fünf CNN-Schichten, drei MaxPool-Schichten und drei vollverbundene Schichten mit insgesamt 60 Millionen Parametern. Erstmals wurde die abgeschnittene lineare Funktion ReLU als Aktivierungsfunktion und Dropout zur effektiven Regularisierung verwendet. Das Netz wurde auf zwei GPUs (Graphical Processing Units) trainiert. Es konnte die Top-5-Fehlerrate im Vergleich zum Vorjahr um 36 % auf 16,4 % reduzieren.

Die Sieger des Jahres 2013 (Zeiler und Fergus 2014) verwendeten AlexNet und modifizierten nur einige Hyperparameter. Die Gewinner von 2014 (Szegedy et al. 2015) kon-



**Abb. 5.17** Top-5 Klassifikationsfehler des ImageNet Klassifikationswettbewerbs. Für jedes Jahr ist die Publikation des Gewinners angegeben, die Kurzbezeichnung des Siegernetzwerkes (in Klammern) sowie die Anzahl der Schichten. Ganz rechts findet man den Klassifikationsfehler menschlicher Kategorisierer. Bildnachweis im Anhang

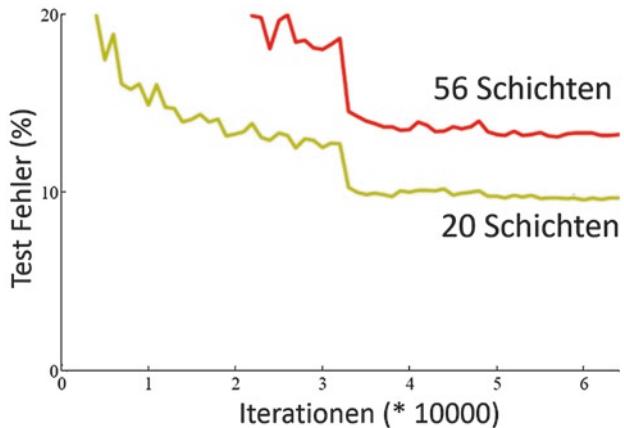
struierten ein CNN mit 22 Schichten, genannt GoogLeNet. Sie verwendeten ein Modul mit mehreren parallelen Convolution-Schichten und rezeptiven Feldern unterschiedlicher Größe (Inception-Modul). Dieses Modul wurde in dem Gesamtnetz mehrfach übereinander angeordnet. Die besten Ergebnisse jeden Jahres sind in Abb. 5.17 dargestellt (Li et al. 2018).

### 5.5.2 ResNet erleichtert die Optimierung durch Umgehungspfade

Wenn man einem dieser Netze weitere Convolution-Schichten hinzufügte, so ergab sich keine weitere Verbesserung. Wie in Abb. 5.18 dargestellt, hatte ein CNN mit 56 Schichten sowohl auf der Trainings- als auch der Testmenge einen höheren Fehler als das CNN mit 20 Schichten. Daraus folgt, dass es sich hierbei nicht um ein Problem der Überanpassung handelt, sondern dass das Optimierungsverfahren das Optimum bei sehr vielen Schichten nur schwer finden kann.

Eine mögliche Lösung ist, die verdeckten Vektoren der unteren Schichten mit einer sogenannten residualen Verbindung (von lat. Residuum „Der Rest“) nach oben zu kopieren, sodass das Modell während der Optimierung mit wenigen „aktiven“ Schichten beginnen und dann die zusätzlichen Schichten im Verlauf der Optimierung für Verbesserungen nutzen kann (Abb. 5.19). Man konstruiert also einen Umgehungspfad oder Bypass, der Aktivierungen an einer Convolution-Schicht vorbeischleust. Sind die Parameter einer Convolution-Schicht nahe Null, so werden die verdeckten Vektoren durch den Umgehungspfad einfach nach oben kopiert und das Netz hat effektiv weniger Schichten. Damit lässt es sich auch zu Beginn leichter optimieren. Diese Netzstruktur heißt Residualer Block, da in der Mathematik eine Restgröße auch Residuum genannt wird.

**Abb. 5.18** Vergleich der Testfehler auf dem CIFAR 10 Datensatz mit 10 Bildklassen bei erhöhter Anzahl von Convolution-Schichten (He et al. 2016).

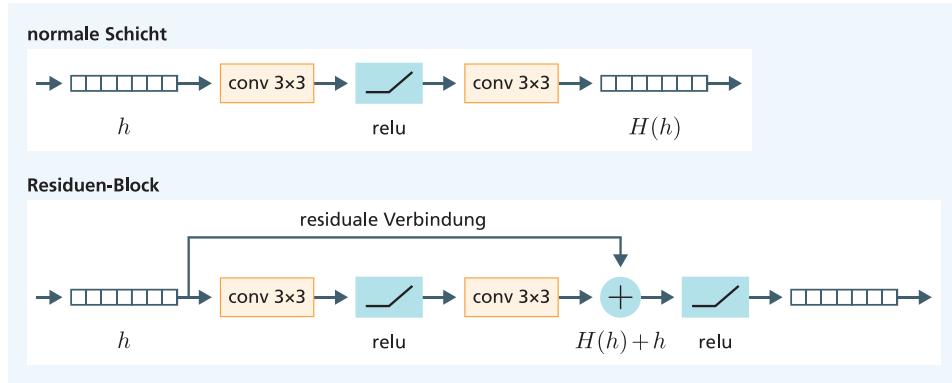


Das Gewinnernetz des Jahres 2015, ResNet (He et al. 2016), folgte diesem Schema. Als Eingabe dienten die drei Farbwert-Matrizen des Bildes. Jeder Residuale Block hat zwei  $3 \times 3$  Convolution-Schichten. Periodisch wird die Anzahl der Kernel verdoppelt. Gleichzeitig wird die Schrittweite der darauffolgenden Kernel verdoppelt, sodass die Gesamtgröße der Ergebnismatrizen gleichbleibt. Zu Vermeidung von Überanpassung wird Batch-Normalisierung und L2-Regularisierung verwendet (Abschn. 4.6.4). Derartige Netze können selbst bei sehr vielen Schichten erfolgreich trainiert werden. Zur Klassifikation von ImageNet wurden z. B. bis zu 152 Schichten verwendet. Bei zusätzlichen Schichten wurde eine immer höhere Genauigkeit erreicht.

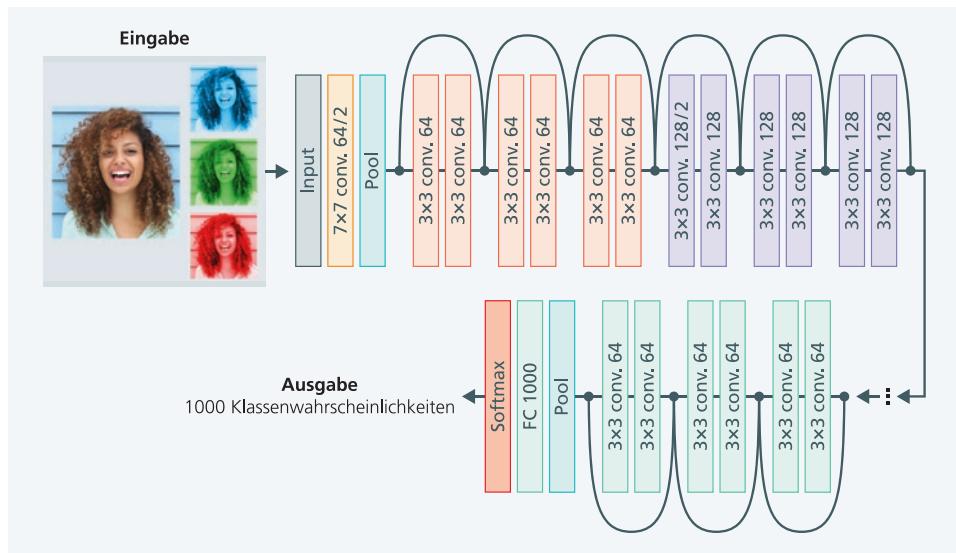
Das ResNet (Abb. 5.20) mit 152 Schichten erzielte in dem ImageNet-Wettbewerb von 2015 einen Top-5-Fehler von 3,6 %. Dies ist besser als die Genauigkeit von 5,1 % bei menschlichen Klassifikatoren. Das Netz hat 60,2 Millionen Parameter (Zagoruyko und Komodakis 2016). Das sind in etwa gleich viele wie beim AlexNet.

### ResNet benötigt eine enorme Rechenpower

Die Autoren nutzten den stochastischen Gradientenabstieg mit einer Minibatchgröße von 256. Die Lernrate beginnt mit 0,1 und wird durch 10 dividiert, wenn der Fehler nicht mehr sinkt. Eine Prognose für ein Bild benötigt etwa  $10^{10}$  FLOPS (Gleitkommaoperationen) (He et al. 2016). Diese 10 Milliarden FLOPS sind etwas weniger, als GoogLeNet mit 22 Schichten benötigt. Insgesamt werden 90 Epochen mit jeweils 1,28 Millionen Bildern trainiert, die in 1000 Klassen eingeteilt wurden. Der Trainingszeitbedarf ist relativ hoch. Um ein ResNet-Netz mit 50 Schichten für 90 Epochen auf einer Nvidia M40 GPU zu trainieren benötigt man 14 Tage, was etwa  $10^{18}$  FLOPS entspricht. Dies kann durch die parallele Verwendung von vielen GPUs stark reduziert werden. Mit 2048 GPU-Chips schaffen es (You et al. 2018) mittlerweile, ImageNet in 14 Minuten zu trainieren.



**Abb. 5.19** Normale Convolution-Schichten (oben) und entsprechender residualer Block (unten). Die residualen Blöcke können extrem oft hintereinander angeordnet werden, ohne die Parameteroptimierung zu beeinträchtigen

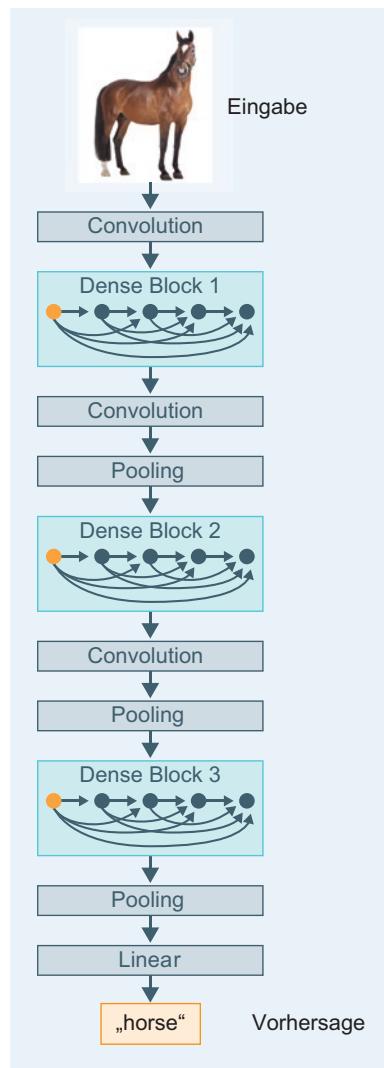


**Abb. 5.20** ResNet mit bis zu 152 Schichten und residualen Verbindungen zur Umgehung der Convolution-Schichten. Die Schichten sind folgendermaßen abgekürzt:  $7 \times 7$  conv. 64/2 ist eine Convolution-Schicht mit  $7 \times 7$  rezeptivem Feld, 64 Resultatmatrizen und Schrittweite 2. Pool ist eine Maxpooling-Schicht, FC 1000 eine vollverbundene Schicht mit einem Ausgabevektor der Länge 1000. Bildnachweis im Anhang

### 5.5.3 DenseNet verwendet zusätzliche Umgehungspfade

Im DenseNet (Huang et al. 2017) wurde das Muster der residualen Verbindungen weiter ausgeweitet. Hier erhält jede Schicht Eingaben von allen vorherigen Schichten (Abb. 5.21). Dabei werden die Eingaben nicht addiert, sondern aneinandergehängt, um eine Überlagerung der Merkmale zu vermeiden. Allerdings funktioniert diese Verknüpfung nicht mehr, wenn die Merkmalsauflösung durch Pooling geändert wird. Daher werden die Schichten in unterschiedliche Dense-Blocks aufgeteilt. Jede Schicht enthält als Teil-Operator eine

**Abb. 5.21** DenseNet mit drei Dense-Blocks. Innerhalb eines Dense-Blocks gibt jede Schicht ihre Ausgabe an alle nachfolgenden Schichten weiter. Bildnachweis im Anhang (Huang et al. 2017)



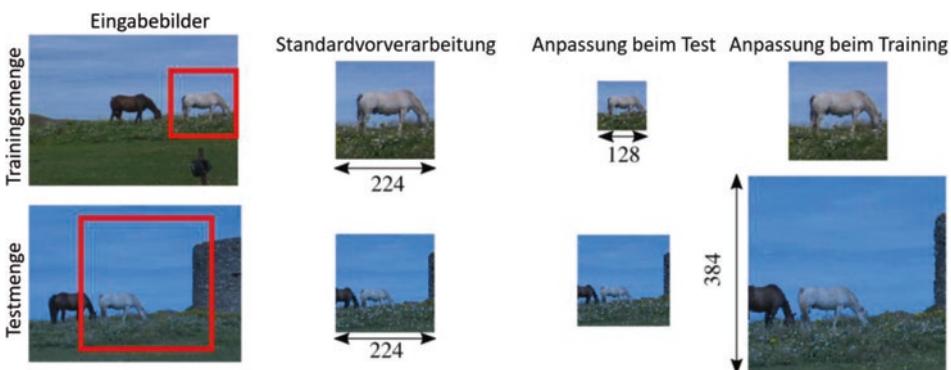
Batch-Normalisierung, eine ReLU-Nichtlinearität und eine  $3 \times 3$  Convolution-Schicht. Insgesamt hat DenseNet (Huang et al. 2017) bis zu 250 einzelne Schichten.

Auf ImageNet erreichte das ursprüngliche DenseNet einen Top5-Fehler von 6,15 %. Die Autoren stellen heraus, dass DenseNet weniger Parameter als ResNet benötigt, um einen vorgegebenen Fehler zu erreichen.

### 5.5.4 ResNeXt nutzt transformierte Bilder zum Training

Inzwischen wurde der Top-5-Fehler auf 2,0 % gesenkt (Touvron et al. 2019). Hierzu wurde das ResNeXt-101-Netz mit 101 Schichten und 829 Millionen Parametern trainiert. Das Modell verwendet Eingabebilder unterschiedlicher Pixelgrößen. Als besonders wirksam zur Steigerung der Genauigkeit erwies sich die Anreicherung der Trainingsdaten (Data Augmentation) durch unterschiedliche Transformationen, wie etwa unterschiedliche Bildausschnitte, horizontales Spiegeln und Farbvariationen.

Mittlerweile werden TNN zur Bilderkennung nicht mehr manuell konstruiert, sondern ihre Architektur wird durch automatische Verfahren bestimmt. Eine neuere Entwicklung definiert hierzu ein umfassendes differenzierbares Suchproblem, welches in Gänze durch Gradientenabstieg gelöst werden kann (Noy et al. 2019). Dieses Verfahren zu Hyperparameter-Optimierung benötigt wesentlich weniger Rechenzeit als die bisherigen Ansätze zur Architekturoptimierung. Abb. 4.33 zeigt die Suchdauer und erzielte Genauigkeit für die CIFAR-10 Bildklassifikation mit 10 Klassen. Der Klassifikationsfehler kann von 2,55 % auf 1,99 % verbessert werden mit einem Rechenaufwand von nur 0,2 GPU-Tagen (Abb. 5.22).



**Abb. 5.22** Das ResNeXt-Modell nutzt Modifikationen der Trainingsdaten, um verbesserte Genauigkeiten zu erzielen. Beispielsweise werden Teilbilder (rot umrandet) ausgewählt und zum Training genutzt. Standardmäßig werden die Ausschnitte auf eine Größe normalisiert. Alternativ werden aber auch verkleinerte oder vergrößerte Bilder als Trainingseingaben genutzt (Touvron et al. 2019). Bildnachweis im Anhang

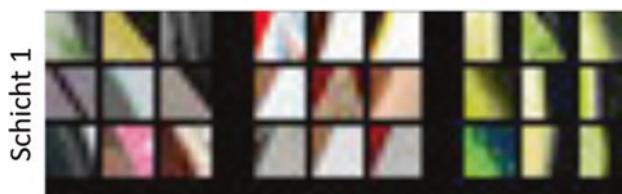
## 5.6 Analyse der CNN Ergebnisse

### 5.6.1 Einzelne Kernel reagieren auf Merkmale unterschiedlicher Art und Größe

Es ist aufschlussreich, zu untersuchen, auf welche Bildmerkmale die unterschiedlichen Kernel der Convolution-Schichten ansprechen. Zeiler und Fergus (2014) wählten zunächst Kernel der jeweiligen Convolution-Schichten des CNN aus. Sie entwickelten eine Rückprojektionstechnik, mit der die Eingabemuster gefunden werden können, die die hohe Aktivität dieser Kernel verursacht haben. In Abb. 5.23 und 5.24 sind die Ergebnisse dieser Untersuchung für die ImageNet-Daten mit einer Variante des AlexNet (Abschn. 5.5.1) zu sehen. In den Schichten 1–5 wurden zufällig einzelne Kernel ausgewählt. Anschließend wurden für die Bilder der Validationsmenge, die ja nicht beim Training verwendet wurde, die jeweils höchsten Ausgabewerte jedes Kernels bestimmt und durch Rückprojektion die zugehörigen Eingabemuster konstruiert.

Abb. 5.23 und 5.24 enthalten diese Rückprojektionen für die Schichten von 1–5. Die Komplexität der Abbildungen steigt mit der Tiefe der Schicht. Für Schicht 1 gibt es lediglich Kanten in verschiedenen Richtungen für unterschiedliche Farben und Farbflächen. Schicht 2 reagiert auf Ecken und Kombinationen von Kanten und Farben. Schicht 3 enthält komplexere Formen, z. B. Kreisstrukturen und ähnliche Texturen (kleinteilige Muster). Schicht 4 enthält schon größere Objekte, etwa Hundegesichter und runde Volumina. Schicht 5 zeigt komplettete Objekte in unterschiedlichen Perspektiven, z. B. Tastatur-Objekte, aber auch Grasflächen im Hintergrund. Interessant ist die höhere Invarianz der Merkmale in höheren Schichten sowie die Betonung der diskriminierenden Bestandteile der Bilder (z. B. Nasen und Augen von Hunden).

Szegedy et al. (2014) untersuchten die semantische Bedeutung einzelner Komponenten der verdeckten Vektoren. Oft wird implizit davon ausgegangen, dass die Komponenten des verdeckten Vektors der letzten Schicht  $h = (h_1, \dots, h_k)$  eine eindeutige Basis bilden, also die Komponenten dieses Vektors inhaltlich gut interpretierbar sind. Stattdessen zeigten die Autoren, dass zufällige Projektionen  $q(h) = (q_1, \dots, q_k)$  von  $h$  semantisch nicht von den



**Abb. 5.23** Visualisierung von einzelnen Kernen eines trainierten CNNs in der Schicht 1. Für jeden Kernel werden rückprojizierte Muster aus der Validationsmenge gezeigt, die hohe Ausgabewerte der Kernel verursachen. Bildnachweis im Anhang



**Abb. 5.24** Visualisierung von einzelnen Kernen eines trainierten CNNs in den Schichten 2–5. Für jeden der Kernel werden rückprojizierte Muster aus der Validationsmenge gezeigt, die hohe Ausgabewerte der Kernel verursachen. Bildnachweis im Anhang

Koordinaten  $(h_1, \dots, h_k)$  zu unterscheiden sind. Offenbar haben die einzelnen Komponenten  $h_i$  oder  $q_j$  keine spezielle Bedeutung. Vielmehr enthalten nur alle Komponenten zusammen die benötigte Information zu einer Klassifikation.

### 5.6.2 Ähnlichen Bildern entsprechen benachbarte verdeckte Vektoren

Die berechneten verdeckten Vektoren kann man als Punkte in einem hochdimensionalen Raum betrachten. Zwischen diesen Punkten lässt sich die Distanz ausrechnen, um Nachbarschaften zwischen Elementen festzustellen. Krizhevsky et al. (2012) haben dies für den Eingabevektor  $h$  der Softmax-Schicht des AlexNet (Abschn. 5.5.1) durchgeführt. Dieser Vektor hat die Länge 4096 und kodiert sämtliche für die Klassifikation relevanten Eigenchaften.

Abb. 5.25 enthält in der linken Spalte fünf Bilder aus der ImageNet-Testmenge, zu denen der jeweilige Vektor  $h$  berechnet wurde. In den Spalten 2–5 finden sich Bilder aus der ImageNet-Testmenge, deren  $h$ -Vektoren den geringsten Abstand zu den  $h$ -Vektoren der linken Spalte besitzen. All diese Bilder wurden also nicht zum Training verwendet. Es zeigt sich, dass die  $h$ -Vektoren die Bilder gut charakterisieren und dass sie für eine Ähnlichkeitssuche geeignet sind. Es gibt eine Reihe von Retrieval-Verfahren für Bilder, welche auf diesen berechneten Vektoren der CNN aufbauen.



**Abb. 5.25** Der verdeckte Vektor  $h$  der letzten CNN-Schicht kann zur Ähnlichkeitsberechnung genutzt werden. Die linke Spalte enthält neue Bilder, zu denen ähnliche Bilder gesucht werden. Die rechten vier Spalten enthalten gefundenen ähnliche Bilder. Bildnachweis im Anhang

## 5.7 Transferlernen reduziert den Bedarf an Trainingsdaten

ImageNet (Deng et al. 2009) umfasst mehr als 14 Millionen Bilder für die 1000 Kategorien (Abschn. 2.1). Zu jeder Klasse gibt es Hunderte bis Tausende von Bildern. Für diesen großen Trainingsdatenbestand waren die TNN erstmals in der Lage, die Klassen der Bildobjekte mit hoher Sicherheit zuzuordnen. Ein wichtiger Kritikpunkt lautet (Marcus 2018), dass Menschen in der Lage sind, eine neue Art von Objekten schon zu erkennen, wenn sie nur wenige Beispiele gesehen haben. In gleicher Weise gilt dies auch für die Spracherken-

nung oder das Training von Übersetzungsmodellen, welche sehr große Textdatenbestände zum Training benötigen.

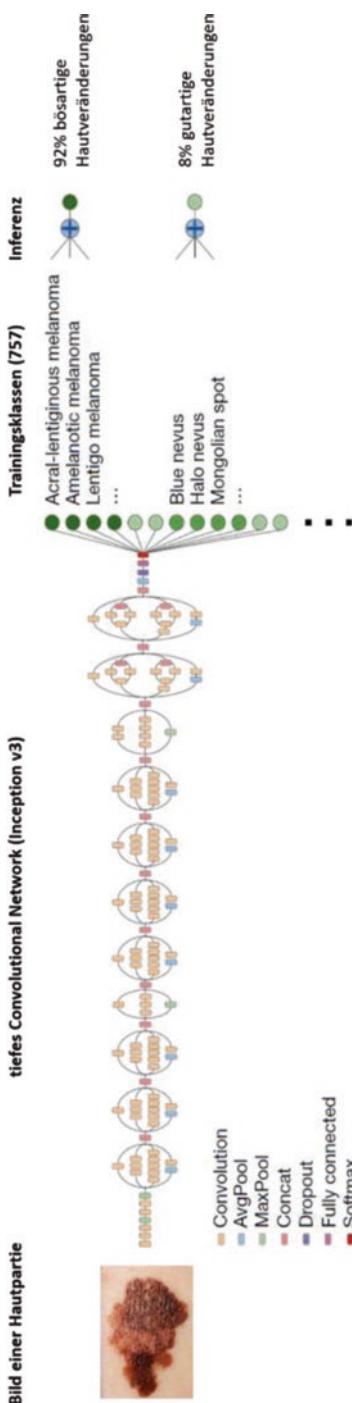
Diese Kritik spricht ein großes Problem der Künstlichen Intelligenz an, und ist in der Praxis oft ein starkes Hindernis für die Anwendung der KI. Ein Gegenargument ist die Feststellung, dass auch Menschen in ihrer Kindheit sehr viel Zeit und Beobachtungen benötigen, bis sie eine große Anzahl von Objekten unterscheiden können.

Als Antwort auf diese Kritik wurde das Konzept des Transferlernens entwickelt. Hierbei wird zunächst ein Modell mit ausreichenden Trainingsdaten für ein Lernproblem trainiert. Anschließend wird das Modell für ein leicht verändertes Problem weiter trainiert, wobei idealerweise nur noch wenige zusätzliche Daten erforderlich sind. Die Ergebnisse aus dem Training mit ImageNet konnten Cui et al. (2018) nutzen und auf dem Datenbestand iNaturalists mit über 5000 Kategorien aus den Bereichen (Pflanzen, Vögel, etc.) mit wenig Trainingsaufwand einen Top5-Fehler von 5,4 % erreichen (Abb. 5.26). Lee et al. (2019) zeigen, dass es auch möglich ist, bei einem Training mit nur fünf Trainingsbeispielen einer Klasse eine Genauigkeit von 80 % zu erzielen. Offenbar sind Ansätze zum Transferlernen in vielen Fällen geeignet, die Anzahl der benötigten Trainingsbeispiele zu reduzieren.

Beispielsweise war es möglich Transferlernen für die Diagnose von Hautkrebs zu nutzen. Hierbei wurde ein tiefes CNN auf ImageNet vortrainiert (Abb. 5.27). Hierdurch hatte das Netzwerk schon viele Merkmale für die Interpretation und Klassifikation von Bildern gelernt. Anschließend wurde es auf einer relativ kleinen Trainingsmenge für die Klassifizierung möglicher Arten von Hautkrebs angepasst. Durch dieses Vorgehen war das TNN in der Lage, Hautkrebs mit der gleichen Genauigkeit zu klassifizieren wie erfahrene Dermatologen (Abschn. 2.1.1). Die Verwendung von vortrainierten CNN für spezielle Aufgaben der Bilderkennung ist mittlerweile Standard.

**Abb. 5.26** Unterschiedliche Trainingsbeispiele bei der feingranularen Bildklassifikation iNaturalists mit über 5000 Kategorien (Van Horn et al. 2018). Bildnachweis im Anhang





**Abb. 5.27** Transferlernen bei der Erkennung von Hautkrebs. Hierbei wird ein vortrainiertes tiefes CNN (Google Inception v3) genutzt, welches auf ImageNet mit 1.3 Millionen Bildern für 1000 Klassen vortrainiert wurde. Anschließend wurde es auf 129.450 Bilder von Hautveränderungen trainiert mit 757 Ausgabeklassen. Zusätzlich wird die Wahrscheinlichkeit einer bösartigen bzw. gutartigen Veränderung für den jeweiligen Fall ausgegeben (Esteva et al. 2017). Bildnachweis im Anhang

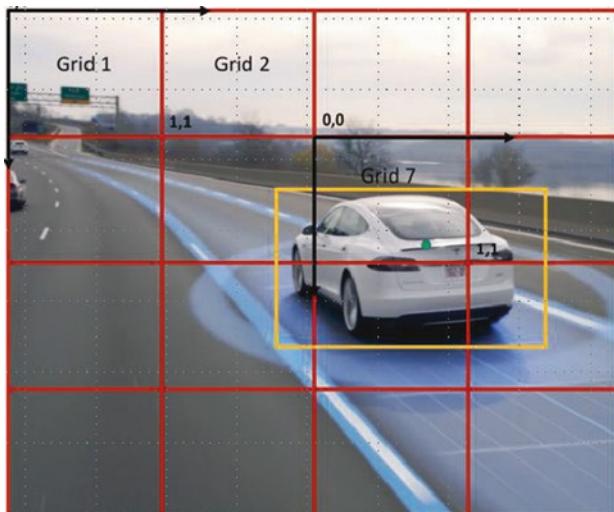
Es ist möglich, mit halbüberwachtem Lernen die Ergebnisse der Bildklassifikation zu verbessern. Bei dieser Art von Lernen werden manuell annotierte, aber auch nicht-annotierte Trainingsbeispiele verwendet. Beispielsweise trainieren Yalniz et al. (2019) ein erstes „Teacher Model“ auf annotierten Trainingsdaten (ImageNet). Mit diesem Teacher Modell klassifizieren sie eine sehr große Menge (z. B. 1 Mrd.) von ungelabelten Bildern. Für jede Klasse suchen sie mehrere tausend der mit höchster Sicherheit klassifizierten Bilder jeder Klasse aus und trainieren damit ein „Student Model“. Zum Schluss erfolgt ein Feintuning des Student Models mit dem ursprünglichen annotierten Datensatz. Für den ImageNet Datensatz mit 1000 Klassen nutzen Yalniz et al. (2019) zum Beispiel einen Datensatz von einer Milliarde Bilder aus einer Social Media Webseite, die mit unterschiedlichen Hashtags versehen waren. Die Autoren können mit ihrem Ansatz die Top-1 Genauigkeit für mehrere der besten Netzarchitekturen um jeweils mehr als 3 % verbessern.

---

## 5.8 Lokalisierung eines Objektes im Bild

### 5.8.1 Objektlokalisierung durch Rechtecke

Häufig möchte man nicht nur wissen, ob in einem Bild ein Auto auftaucht, sondern auch, an welcher Position das Auto im Bild ist. Bei der Objektlokalisierung (Abschn. 5.1.1) wird die Lage eines Objektes mit einer Bounding-Box angegeben. Enthält ein Bild mehrere Objekte, so hat die Objektlokalisierung die Aufgabe, die Klasse dieser Objekte zusammen mit deren Bounding-Boxen zu bestimmen. Eine bekannte Anwendung ist das autonome Fahren. Damit

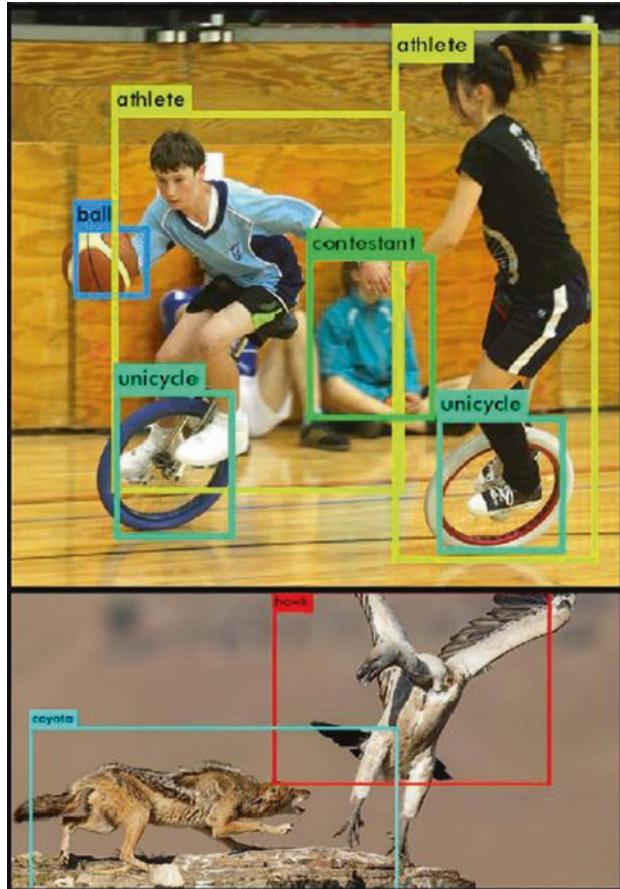


**Abb. 5.28** Beim YOLO-Verfahren zur Lokalisation von Objekten in Bildern wird das Bild in Gitterzellen aufgeteilt. Ist das Zentrum eines Objektes in einer Gitterzelle, so wird die Position des Zentrums sowie die Breite und Höhe der Bounding Box prognostiziert. Zudem wird die Wahrscheinlichkeit für die Bildklasse für jede Gitterzelle geschätzt. Bildnachweis im Anhang

autonome Fahrzeuge sicher navigieren können, müssen sie andere Autos, Fußgänger, Fahrräder, Bäume usw. erkennen und auf dem Videobild und damit in der Umgebung lokalisieren.

Ein für diese Aufgabe entwickeltes Verfahren ist YOLO (Redmon et al. 2016). Es teilt ein Bild in ein regelmäßiges Gitter auf. In Abb. 5.28 ist dies der Einfachheit halber ein  $4 \times 4$ -Gitter, normalerweise aber ein feineres Gitter.

In jeder Gitterzelle kann ein anderes Objekt erkannt werden. Wenn  $k$  verschiedene Objektklassen möglich sind, so wird die Position des Zentrums jedes Objektes ausgegeben ( $Bx, By$ ) sowie die Breite und Höhe ( $Bw, Bh$ ) der Bounding-Box relativ zur Breite/Höhe der Gitterzelle. In Abb. 5.29 ist das Zentrum des Autos innerhalb von Gitterzelle „Grid 7“.  
Damit kann das Verfahren die exakten Koordinaten der Bounding-Box (gelb) berechnen. Zusätzlich wird noch zu jeder Klasse die Wahrscheinlichkeit ausgegeben, mit der sich in der Gitterzelle ein Objekt der Klasse befindet. Zudem wird noch eine Wahrscheinlichkeit geschätzt, ob sich überhaupt ein Objekt in der Klasse befindet.



**Abb. 5.29** Yolo9000 kann Bounding-Boxen für 9000 verschiedene Objektklassen erstellen. Bildnachweis im Anhang

Insgesamt prognostiziert das Verfahren für jede Zelle die Klasse eines darin enthaltenen Objektes zusammen mit dessen Bounding-Box. Dies geschieht mit einem einzigen tiefen CNN, welches die Ausgaben für jede Gitterzelle in einem einzigen Prognoseschritt generiert.

Nachteil des Verfahrens ist, dass in jeder Gitterzelle nur ein Objekt erkannt werden kann. Hierfür wurden mittlerweile Varianten entwickelt, z. B. Yolo9000 (Redmon und Farhadi 2017), welches 9000 Objektkategorien unterscheiden kann und trotzdem in der Lage ist, in Realzeit Videos zu annotieren. Abb. 5.29 zeigt die von Yolo9000 gefundenen Bounding-Boxen für zwei verschiedene Bilder.

### 5.8.2 Bildsegmentierung zur pixelgenaue Bestimmung von Klassen

Bildsegmentierung ist eine Bilderkennungsaufgabe, in der die Regionen der einzelnen Objekte eines Bildes genauer gekennzeichnet werden als mit Bounding-Boxen. Das Ziel der Bildsegmentierung ist die Klassifikation jedes Pixels des Bildes mit der Klasse des Objektes, welches es darstellt (Abb. 5.30). Geht es um die Zuordnung von Objekten einzelner Klassen (z. B. einzelner Personen) zu den Bildpixeln, so spricht man von Objektsegmentierung. Hierbei werden also z. B. die Pixel unterschiedlicher Personen unterschiedlich dargestellt.

In einigen Anwendungen sind Segmentierungen besonders relevant. Bei selbstfahrenden Autos z. B. ist es wichtig, die Position von Straßen, Personen und anderen Fahrzeugen zuverlässig zu erkennen. In der medizinischen Diagnostik von Röntgenbildern muss die Lage der einzelnen Organe und Krankheitsherde in einem Bild genau erfasst werden.



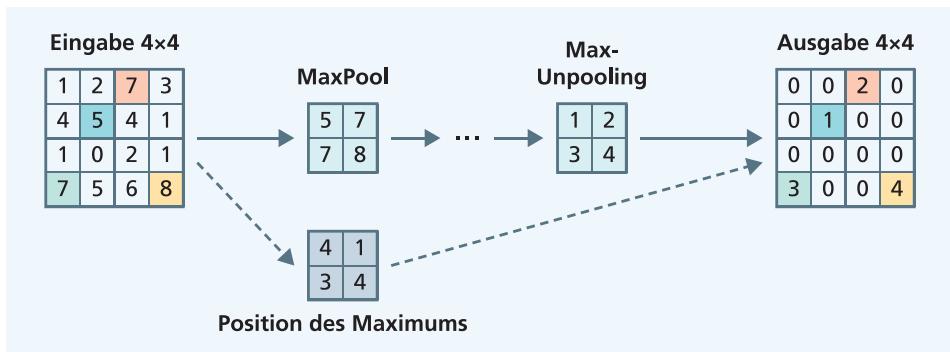
**Abb. 5.30** Originalbild (links) und segmentiertes Bild (rechts) mit den Objektklassen Person, Fahrrad und Hintergrund. Bildnachweis im Anhang

Ein naiver Ansatz für ein neuronales Netz zur Bildsegmentierung könnte ein Modell aus mehreren Convolution-Schichten vorsehen, bei denen die dazwischenliegenden Ergebnismatrizen genauso groß sind wie die Pixelmatrix des Eingabebildes. Jedem Ausgabepixel könnte dann die Wahrscheinlichkeit einer Klasse zugeordnet werden (Li et al. 2017, S. 22). Da dieses Vorgehen einen zu hohen Rechenaufwand erfordert, ist man bestrebt, die Ausdehnung der dazwischenliegenden Schichten zu verringern.

Ein verbreiteter Ansatz zur Bildsegmentierung besteht aus einem Encoder und einem Decoder. Der Encoder reduziert die räumliche Auflösung der Eingabe (Li et al. 2017, S. 24), z. B. durch Pooling oder Convolution-Schichten mit hoher Schrittweite (stride). Dabei kann man Merkmale trainieren, welche bei der Unterscheidung zwischen den Klassen hocheffizient sind. Anschließend kann man durch den Decoder die Auflösung wieder erhöhen und die zuvor gefundenen Klassifizierungen auf die Pixel des Eingabebildes verteilen.

### 5.8.3 Max-Unpooling belegt ein vergrößertes Feld mit Werten

Für die Erhöhung der Auflösung (Upsampling) wurden unterschiedliche Ansätze entwickelt. Beim Max-Unpooling wird der Effekt einer vorherigen Maxpool-Schicht umgekehrt (s. Abb. 5.31). Beim Maxpooling wird nicht nur der Wert des Maximums, z. B. für jedes  $2 \times 2$ -Feld, extrahiert, sondern auch die Position des Maximums für jedes Feld gespeichert (Zeiler und Fergus 2014). Beim Max-Unpooling, welches einige Schichten weiter stattfindet, wird das Ergebnis der Netzberechnungen auf eine höhere Auflösung übertragen. Der berechnete Wert wird an derjenigen Position in das Feld eingetragen, an welchem zuvor das Maximum in der Maxpooling-Schicht auftrat. Damit ist die Lokalisierung der Merkmale aus der Pooling-Schicht übertragen worden und kann für die pixelgenaue Markierung der Klassen verwendet werden (Noh et al. 2015).

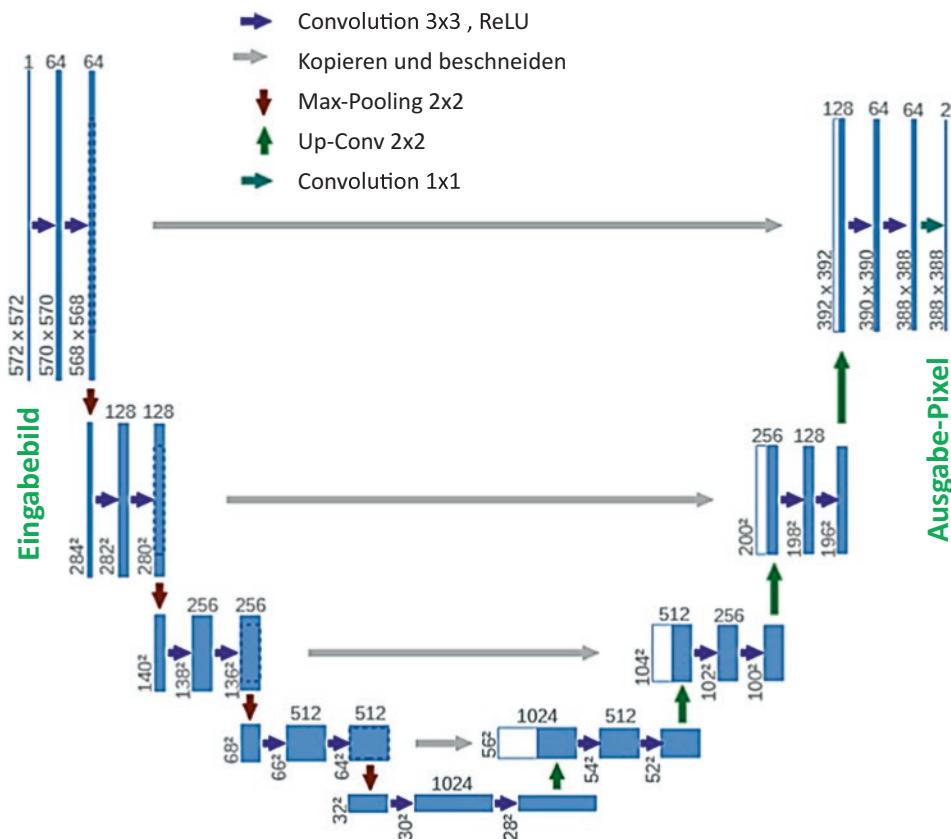


**Abb. 5.31** Erhöhung der Auflösung eines CNN durch Max-Unpooling. Beim Pooling (links) wird für receptive Felder der Größe  $2 \times 2$  sowohl das Maximum als auch die Position des Maximums berechnet. Beim Unpooling wird die Position des Maximums verwendet, um den Unpooling-Wert in das „richtige“ Eingabefeld zu setzen, an dem zuvor das Maximum auftrat

### 5.8.4 Das U-Net erkennt zunächst Objekte und findet dann die zugehörigen Pixel

Ronneberger et al. (2015) haben das U-Net zur Objektsegmentierung entwickelt, welches Objekte einer Klasse in einem Bild zunächst erkennt und anschließend die zugehörigen Pixel identifiziert. Das Netz besteht aus einem kontrahierenden Teilnetz, welches die Bildteile zueinander in Beziehung setzt, und einem nachfolgenden symmetrischen expandierenden Teilnetz, welches die präzise Verortung der Klasseninformationen berechnet.

Die U-Net-Architektur ist in Abb. 5.32 dargestellt. Die Eingabegröße ist  $572 \times 572$  Pixel. Hierauf werden im kontrahierenden Teilnetz mehrfach  $3 \times 3$ -Convolution-Schichten angewendet gefolgt von einer abgeschnittenen linearen Funktion (ReLU) und einem  $2 \times 2$  Max-Pooling mit Schrittweite 2 zur Reduktion der Eingabegröße. In jedem Schritt des



**Abb. 5.32** Architektur des U-Net mit dem kontrahierenden Teilnetz (links) und dem expandierenden Teilnetz (rechts). Jede blaue Box entspricht einer Merkmalsmatrix. Die Anzahl der Merkmale steht über der Box, die Auflösung an der linken Seite. Weiße Boxen repräsentieren die kopierten Werte aus dem kontrahierenden Teilnetz. Bildnachweis im Anhang

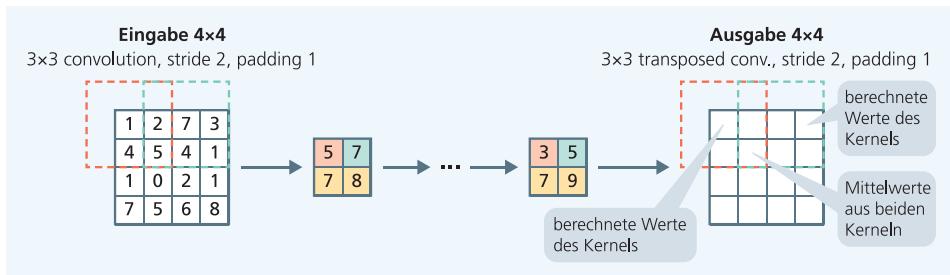
kontrahierenden Netzes wird gleichzeitig die Anzahl der Merkmale verdoppelt. Die Ausgabe des kontrahierenden Teilnetzes hat eine Feldgröße von  $28 \times 28$  und 1024 Merkmale.

Das expandierende Teilnetz besteht aus Transposed-Convolutions, welche die Feldgröße erhöhen. Bei der Transposed-Convolution wird für jede vorherige Convolution-Schicht des kontrahierenden Teilnetzes eine zugehörige Transposed-Convolution definiert. Die Convolution-Schicht hat z. B. einen trainierbaren  $3 \times 3$ -Kernel mit Schrittweite 2 und ragt um einen Schritt über das Eingabefeld hinaus (padding 1). Der Kernel wird um jeweils 2 Positionen verschoben, sodass das Ausgabefeld nur halb so groß ist wie das Eingabefeld. Die zugehörige Transposed-Convolution bildet jedes Eingabepixel auf ein  $3 \times 3$ -Feld ab und verwendet einen trainierbaren  $3 \times 3$ -Kernel, wie in Abb. 5.33 dargestellt. In den Bereichen, in denen sich die Kernel überlappen, werden die berechneten Werte aufsummiert.

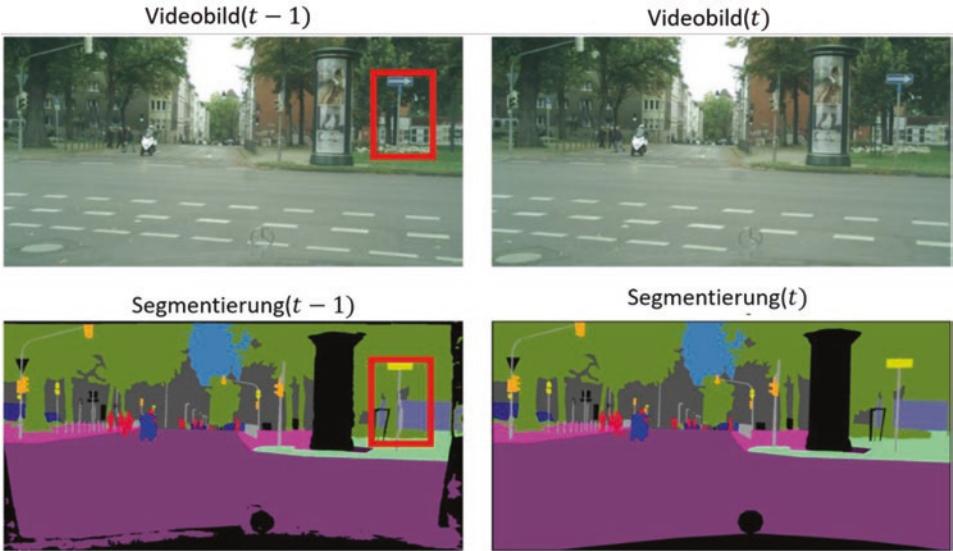
Aus der entsprechenden Schicht des kontrahierenden Netzes werden die dortigen Merkmale kopiert und nach entsprechender Beschneidung am Rand dem Ergebnis der Transposed-Convolution angehängt. Es folgen zwei  $3 \times 3$ -Convolution-Schichten und eine abgeschnittene lineare Funktion (ReLU). Dieses Vorgehen wird dreimal wiederholt und ergibt ein Ausgabefeld von  $388 \times 388$  mit 64 Merkmalen. Zum Schluss wird eine  $1 \times 1$  Convolution verwendet, um die Klassenausgabe für jedes Pixel zu berechnen.

Das Netz wird mit Beispielen trainiert, welche wie bei Abb. 5.30 aus dem Originalbild und dessen Segmentierung bestehen. Das Netz ist sehr populär und wurde von einer Reihe von Autoren erweitert (Jordan 2018). Zur Regularisierung werden heute zusätzlich BatchNorm-Schichten eingefügt (Tutani 2018).

Zhu et al. (2019) wenden Segmentierungstechniken auf Videos von Straßenszenen an (Abb. 5.34). Dabei nutzen sie aus, dass die Label von Pixeln zwischen den verschiedenen Videobildern sich meist nicht ändern. Sie prognostizieren aus den bisherigen Videobildern die nächsten Bilder und die damit zusammenhängende Segmentierung. Hiermit können sie die Genauigkeit der Segmentierung steigern und erreichen die besten Resultate (83,5 % in einer speziellen Metrik für Bildsegmentierung) auf dem Cityscapes-Datensatz.



**Abb. 5.33** Die kontrahierende Schicht besteht aus einer  $3 \times 3$ -Convolution-Schicht mit Schrittweite 2 (stride) und Padding 1. Sie bewirkt eine Verkleinerung des Eingabefeldes. Die zugehörige expandierende Transposed-Convolution bewirkt eine Vergrößerung des Eingabefeldes. Sie bildet jedes Eingabepixel auf ein  $3 \times 3$ -Feld ab. Die Parameter des Kernels sind lernbar und im Überlappungsbereich werden die Ausgabewerte summiert

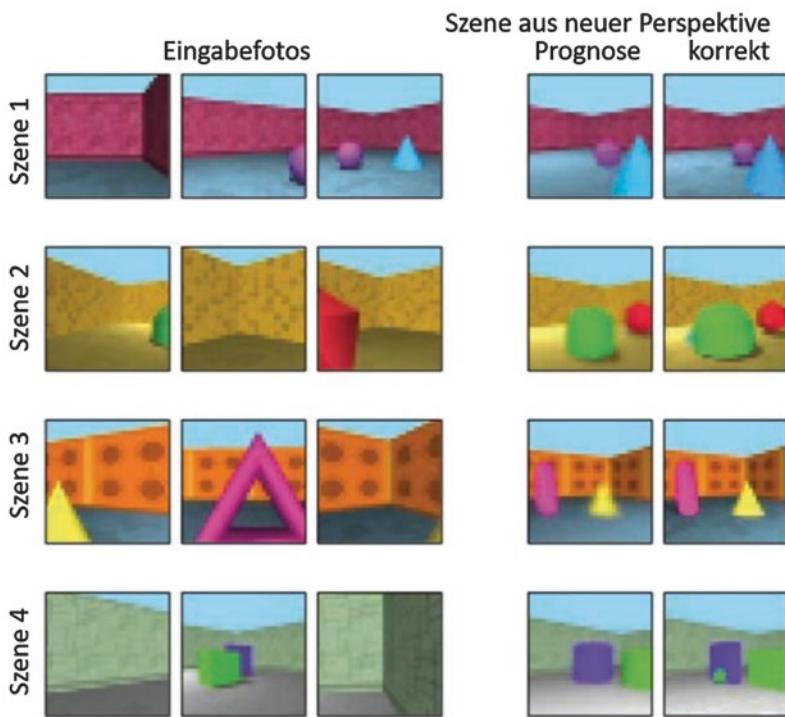


**Abb. 5.34** Bildsegmentierung für den Cityscapes-Datensatz für aufeinanderfolgende Videobilder. In der oberen Reihe sind die Originalbilder und darunter die vorhergesagte Segmentierung. Bemerkenswert ist die genaue Erkennung und Vorhersage des Motorrads und der Person, die die Straße überquert (Zhu et al. 2019). Bildnachweis im Anhang

## 5.9 Die 3D-Rekonstruktion einer Szene

Das Ziel von Eslami et al. (2018) ist es, bildliche Szenen zu beschreiben, ohne das manuell Bilder annotiert werden müssen. Hierzu definieren sie das generative Query Network, welches die Beschreibung von Szenen nur unter Verwendung von Sensoreingaben lernt. Dieses TNN verwendet als Eingabebilder einer Szene Fotos aus verschiedenen Blickwinkeln und erstellt eine interne Darstellung. Diese Repräsentation wird genutzt, um das Aussehen der Szene aus zuvor nicht beobachteten Blickwinkeln vorherzusagen. Damit kann es Repräsentationen für Szenen ohne menschliche Annotationen erstellen und daraus neue Ansichten erzeugen.

In ihren Experimenten trainieren Eslami et al. (2018) ein TNN, welches aus Eingabebildern aus verschiedenen Aspekten ein Embedding ableitet (linke Spalten in Abb. 5.35). Für eine neue Blickrichtung prognostiziert dann eine weitere Modellkomponente eine Szenenansicht (4. Spalte in Abb. 5.35), welche gut mit der wahren Ansicht übereinstimmt (5. Spalte in Abb. 5.35). Die erzeugten Prognosen beachten die Gesetze der Perspektive, der Verdeckung, der Beleuchtung und des Schattens. Lediglich in Szene 4 kann das Modell nicht die grüne Kugel prognostizieren, weil sie auf den Eingabebildern verdeckt ist.



**Abb. 5.35** Trainingsbilder für einzelne Szenen (links), prognostiziertes Bild für eine neue Blickrichtung (4. Spalte) und wahres Bild für die neue Blickrichtung. Bildnachweis im Anhang

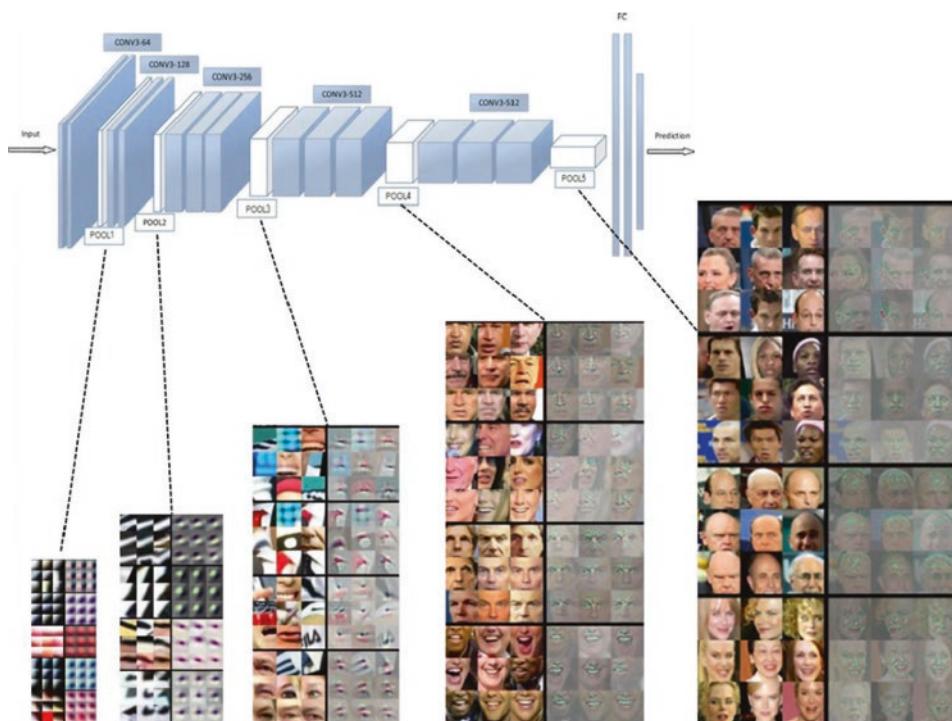
## 5.10 Gesichter können mit sehr hoher Genauigkeit zugeordnet werden

Eine wichtige Anwendung der Bildklassifikation ist die Gesichtserkennung. Abb. 5.36 zeigt die Schwierigkeit der Erkennung verschiedener Personen. Zum einen können die Fotos sehr verschiedenartige Lichtverhältnisse aufweisen. Außerdem können gewollte Gesichtsveränderungen wie Make-up, Brillen oder Frisuren eine Identifikation erschweren. Wang et al. (2019) geben einen umfassenden Überblick über Gesichtserkennung.

In einem ersten Schritt werden zunächst die Gesichter in einem Bild mit einer robusten Technik lokalisiert und auf die gleiche Größe transformiert (Ranjan et al. 2018). Anschließend wird ein tiefes CNN, z. B. ResNet, mit diesem Datensatz von Gesichtern trainiert (Abb. 5.37). Ein bekannter Trainingsdatensatz zur Gesichtserkennung ist LFW (Labeled Faces in the Wild). Er besteht aus den Gesichtern berühmter Personen, welche



**Abb. 5.36** Zwei Beispiele zur Gesichtserkennung. Das linke Bilderpaar zeigt die gleiche Person, das rechte Bilderpaar verschiedene Personen. Bildnachweis im Anhang



**Abb. 5.37** Architektur eines TNN zur Gesichtserkennung. Wie bei ResNet werden in unterschiedlichen Schichten Merkmale wachsender Größe und Komplexität extrahiert. Dabei werden aus kleinteiligen unteren Merkmalen größere Gesichtsattribute gebildet. Die Ausgabe ist ein verdeckter Vektor, welcher die Eigenschaften des Gesichtes repräsentiert. Bei der Anwendung auf ein unbekanntes Gesicht wird dessen verdeckter Vektor mit den verdeckten Vektoren anderer Gesichter in einer Datenbank verglichen, z. B. über den euklidischen Abstand, und das am besten passende Gesicht ausgewählt (Wang et al. 2019). Bildnachweis im Anhang

ohne Datenschutzprobleme verwendet werden können. Die meisten Bilder zeigen die Personen von vorne in einer großen Variation der Beleuchtungsverhältnisse. Um den Aufwand für die Sammlung von Fotos zu verringern, werden unterschiedliche Methoden der „Datenvermehrung“ verwendet, um Bilder in etwas geänderten Ansichten künstlich zu erzeugen. Eine Variante des Datensatzes umfasst 4 Millionen Bilder von 80.000 Personen.

Wie in Abb. 5.25 werden Bilder durch den verdeckten Vektor der letzten CNN-Schicht repräsentiert. Anschließend wird ein Ähnlichkeitsmaß zwischen der Repräsentation des Testbildes und der Repräsentation der Bilder der bekannten Personen berechnet. Phillips et al. (2018) haben die Performanz eines Gesichtserkennungs-Verfahrens mit einem Merkmalsvektor der Länge 512 mit der Erkennungsleistung einer Reihe menschlicher Experten verglichen. Die Genauigkeit des Algorithmus (0,96 AUC) war besser als bei 73 % der „Examiner“, welche die besten verfügbaren menschlichen Experten sind und im Durchschnitt einen AUC-Wert von 0,93 erreichten. Der AUC-Wert wird aus Präzision und Recall berechnet und sollte möglichst hoch sein. Wang et al. (2019, S. 6) geben für den LFW-Datensatz bei einem auf ResNet mit 100 Schichten beruhenden Netz eine Genauigkeit von 99,83 % an. Das heißt, von 1000 Personen aus dem Testdatensatz werden weniger als 2 falsch zugeordnet.

Viele potentielle Anwendungen, z. B. die Suche nach Personen auf einer Fahndungsliste, erfordern eine hohe Treffsicherheit bei einer sehr niedrigen Fehlalarmrate, z. B.  $10^{-6}$ . Dies ist immer noch eine große Herausforderung selbst bei extrem großen Trainingsdaten. Zudem sinkt die Genauigkeit der Erkennung bei der Anwendung auf nicht-weiße Menschen stark von über 99 % auf 80–90 % (Wang et al. 2019, S. 14), da diese in den Trainingsdaten meist unterrepräsentiert sind. Dies hat in jüngster Zeit zu vermehrten Diskussionen in der Öffentlichkeit über die Zuverlässigkeit von Gesichtserkennungsverfahren geführt.

---

## 5.11 Abschätzung der Genauigkeit von Modellprognosen

TNN werden heute in vielen relevanten Anwendungen genutzt, bei denen oft schwerwiegende Entscheidungen getroffen werden, z. B. zur Diagnose von Tumoren (Abb. 2.2) oder bei selbstfahrenden Autos (Abschn. 2.8). Hier wird dringend eine Abschätzung gebraucht, wann man sich auf die Ergebnisse eines TNN verlassen kann und wann die Resultate unsicher sind. Standardverfahren zum Training liefern für eine Eingabe lediglich einen plausiblen Prognosewert und geben keine Aufschlüsse über die Genauigkeit der Prognose. Die hier diskutierten Techniken lassen sich nicht nur auf die Bildverarbeitung, sondern auf beliebige Modelle des maschinellen Lernens anwenden.

### 5.11.1 Unsicherheit der Modellprognosen

Es gibt verschiedene Arten von Unsicherheiten der Modellprognose (Gal 2016):

- Unsicherheit wegen **fehlender Trainingsabdeckung**: Ein Modell zur Erkennung von Hunderassen wird mit Daten unterschiedlicher Hunderassen trainiert. Es erhält die Aufgabe, das Bild einer Katze zu klassifizieren (Abb. 5.38). Die resultierende Modellprognose ist unsicher, weil Katzen nicht in den Trainingsdaten auftauchen. Analoge Effekte ergeben sich, wenn Teile der Population nur mit wenigen Beispielen in der Trainingsmenge vertreten sind.
- Unsicherheit wegen **Messfehlern**: Ein Modell wird zur Unterscheidung von Hunden und Katzen mit Bildern von Hunden und Katzen trainiert. Es erhält die Aufgabe, ein verschwommenes Katzenbild zu klassifizieren (Abb. 5.39). Die resultierende Modellprognose ist unsicher.
- **Modellunsicherheit**: Es werden mehrere Modelle trainiert mit unterschiedlich vielen Schichten und unterschiedlich langen verdeckten Vektoren (Abb. 5.40). Auch wegen unterschiedlicher Anfangswerte und anderer Zufallseinflüsse (Minibatches) werden sich die Parameterschätzungen unterscheiden. Alle Modelle haben leicht unterschiedliche Werte der Verlustfunktion, sind also alle plausibel. Die zugehörigen Modellprognosen unterscheiden sich, sind also unsicher.

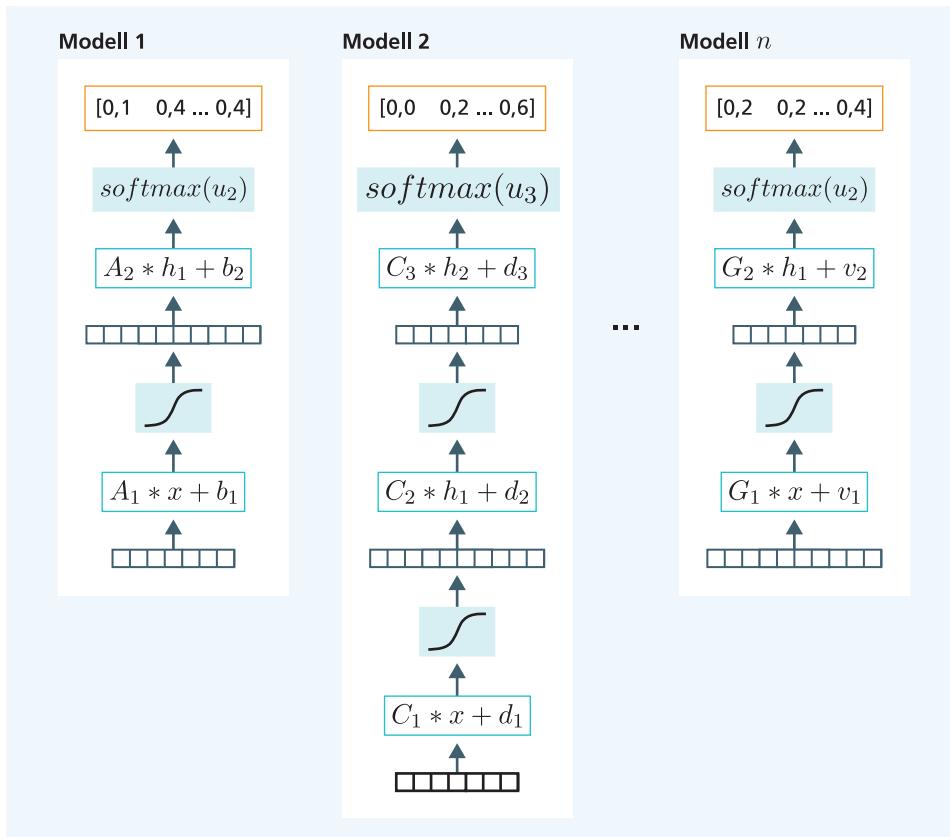
Bisher wurde für ein Modell mithilfe eines Optimierungsverfahrens ein einziger „optimaler“ Parametervektor  $\hat{w}$  bestimmt und damit eine Prognose durchgeführt. Offenbar sind aber auch Parameterwerte in der Nähe von  $\hat{w}$  ebenfalls plausibel und könnten für eine Prognose verwendet werden. Es gibt zwei unterschiedliche Ansätze, Nachbarwerte des optimalen Parameterwertes zu berücksichtigen, um das Ausmaß dieser Unsicherheiten zu bestimmen.



**Abb. 5.38** Ein Klassifikationsmodell für Hunde wurde mit Bildern von Hunden (links) trainiert. Die Klassifikation eines Katzenbildes (rechts) ist unsicher, weil Katzen nicht durch die Trainingsmenge abgedeckt sind. Bildnachweis im Anhang



**Abb. 5.39** Ein Klassifikationsmodell für Bilder wurde mit Bildern von Hunden und Katzen trainiert (links). Die Klassifikation eines unscharfen Eingabebildes (rechts) ist unsicher. Bildnachweis im Anhang



**Abb. 5.40** Modelle mit unterschiedlicher Modellstruktur erreichen unterschiedliche Werte der Verlustfunktion, können aber alle noch plausibel sein. Die zugehörigen Modellprognosen sind unterschiedlich. Die Modellunsicherheit entsteht, da nicht klar ist, welches der Modelle für eine Prognose verwendet werden soll

### 5.11.2 Der Bootstrap erzeugt eine Menge plausibler Modelle

Eine wichtige Ursache der Unsicherheit der Modellprognose ist, dass wir niemals **alle** möglichen Daten zum Training zur Verfügung haben, sondern nur eine Stichprobe dieser potentiellen Trainingsdaten. Der Bootstrap (auch Bagging) ist ein Ansatz, den Effekt dieser Trainingsdatenauswahl abzuschätzen. Er zieht nach einem bestimmten Verfahren neue zufällige Stichproben aus den vorhandenen Trainingsdaten und trainiert eine Reihe von Modellen. Es lässt sich zeigen, dass die zufälligen Stichproben ebenfalls plausible Darstellungen der Realität sind und dass daher diese Modelle die Variation in den Trainingsdaten erfassen (Hastie et al. 2009, S. 261 ff.). Führt man mit jedem dieser Modelle eine Prognose durch, so erhält man eine Verteilung von Prognosen, welche die Modellunsicherheit abbilden.

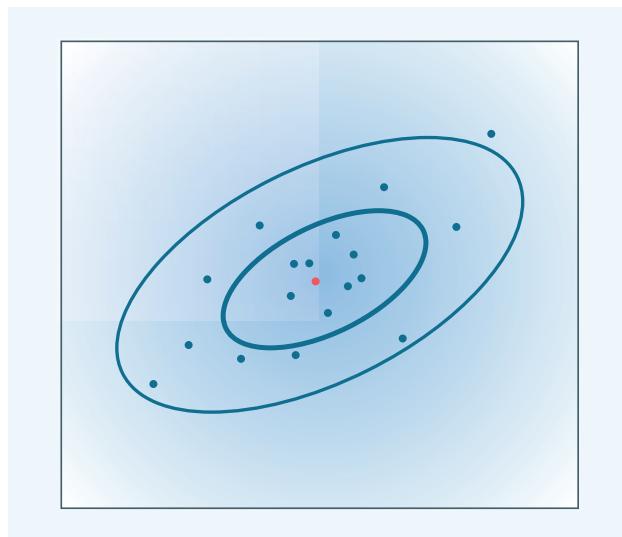
Das Verfahren ist oft erprobt worden und recht zuverlässig. Es hat den Nebeneffekt, dass der Mittelwert der Prognosen diese „Ensembles“ von Modellen genauer ist, als die Prognose mit einem einzelnen Modell. Aus diesem Grunde verbessert ein Ensemble von unterschiedlichen Modellen oft die Zuverlässigkeit (Abb. 5.40) und erlaubt zudem die Abschätzung der Prognoseunsicherheit. Nachteil des Verfahrens ist, dass man eine große Zahl ( $>30$ ) von Modellen trainieren muss, um Aussagen über die Verteilung treffen zu können. Ein Bootstrap-Prognoseintervall ist größer für Eingaben aus Bereichen, für die nur wenige oder gar keine Trainingsdaten verfügbar sind. Damit kann das Verfahren die zusätzliche Unsicherheit abschätzen, wenn zu einer Eingabe nur wenige Trainingsbeispiele vorhanden sind.

### 5.11.3 Bayessche neuronale Netze

Bayessche neuronale Netze betrachten den Modellparameter  $w$  als eine zusätzliche Variable, für den sich eine Wahrscheinlichkeitsverteilung  $p(w|Daten)$  ableiten lässt (Goodfellow et al. 2016, S. 132). Sie wird A-posteriori-Wahrscheinlichkeit genannt und lässt sich als die Wahrscheinlichkeit interpretieren, dass der jeweilige Parameter derjenige des „korrekten Modells“ ist, welches die Daten erzeugt hat. In Abb. 5.41 ist diese Verteilung durch Linien gleicher Wahrscheinlichkeit dargestellt. In der Ellipse in der Mitte liegen besonders viele plausible Modellparameter. Der rote Punkt in der Mitte ist das Ergebnis  $\hat{w}$  der Optimierung, welches meist die höchste Wahrscheinlichkeit besitzt. Die Parameter in der Nähe haben eine kleinere Wahrscheinlichkeit, müssen aber auch in Betracht gezogen werden.

Es ist in der Praxis unmöglich, diese Verteilung analytisch zu bestimmen. Es gibt unterschiedliche Verfahren, sie näherungsweise zu rekonstruieren. Markov-Chain-Monte-Carlo-Verfahren (MCMC-Verfahren) erzeugen eine Reihe aufeinanderfolgender Parametervektoren  $w^{(1)}, w^{(2)}, \dots$ , die diese Verteilung durchwandern und als eine Stichprobe der Verteilung angesehen werden können. Ergebnis ist also eine Menge von Parametern,

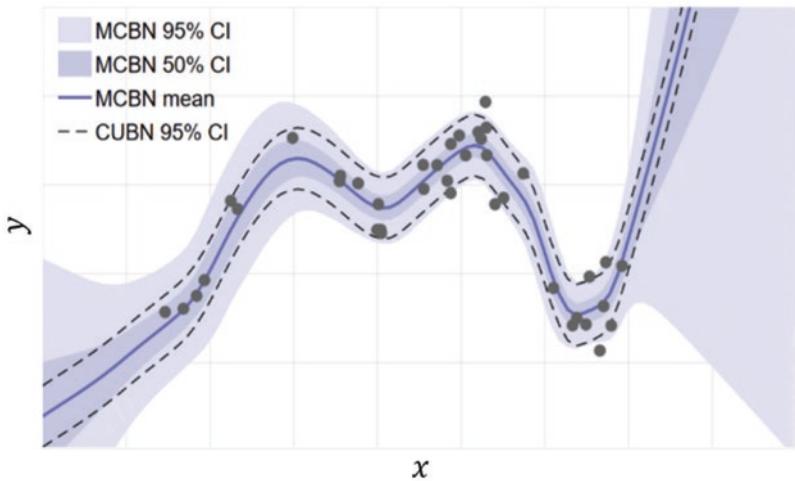
**Abb. 5.41** Bayessche a-posteriori-Verteilung der Modellparameter  $w$  (Punkte) zur Repräsentation der Modellsicherheit. Jeder Punkt ist ein Parameter eines möglichen Modells. Der rote Punkt hat die höchste Wahrscheinlichkeit und ist oft das Ergebnis  $\hat{w}$  der Optimierung



welche die Verteilung der plausiblen Parameter repräsentieren. Welling und Teh (2011) stellen ein Verfahren vor, welches diese Verteilung unter Verwendung von Minibatch-Gradienten effizient berechnet. Aus dieser Verteilung lassen sich Modellprognosen berechnen und deren Unsicherheit abschätzen. Die Autoren zeigen, dass dieses Vorgehen Überanpassung vermeidet.

Gal (2016) zeigt, dass ein mit Dropout (Abschn. 4.6.4) zur Regularisierung trainiertes TNN näherungsweise ein Bayessches neuronales Netz ist. Schätzwerte für die Unsicherheit kann man erhalten, wenn man Prognosen bei unterschiedlichen Dropout-Masken berechnet. Dies verursacht keinen zusätzlichen Aufwand bei der Optimierung. In jüngerer Zeit wird häufig Batchnormalisierung (Abschn. 4.6.4) zur Regularisierung von TNN verwendet. Teye et al. (2018) zeigen, dass das Training eines TNN mit Batchnormalisierung äquivalent ist zu einer näherungsweisen Inferenz in einem Bayesschen neuronalen Netz. Daher lassen sich auch für diese Regularisierungstechnik Abschätzungen der Prognoseunsicherheit ohne großen Rechenaufwand gewinnen (Abb. 5.42). Man sieht hier deutlich, dass die Prognoseunsicherheit außerhalb des von Daten überdeckten Bereiches stark ansteigt.

In Abb. 5.43 wird diese Technik auf ein TNN zur Bildsegmentierung von Fotos angewendet. In der unteren Reihe sind die Unsicherheitsschätzungen dargestellt. Die Unsicherheit ist am höchsten an den Rändern der Objektbereiche. Die Unsicherheitsschätzung ist auf der rechten Seite genauer, weil dort größere Minibatches (36 statt 10 Trainingsbeispiele) genutzt wurden.



**Abb. 5.42** Regressionsaufgabe, bei der der Wert von  $y$  aus den Werten von  $x$  mit einem 3-Schichten-Modell prognostiziert wird. Die beobachteten Daten sind durch Punkte markiert. Mit Hilfe des Batch-Normalisierungs-Verfahrens MCBN wurde der Mittelwert der Prognose (blaue Linie) und die 50 %- und 95%-Prognoseintervalle berechnet (schattiert). Die gestrichelten Linien sind Prognoseintervalle einer unzureichenden Technik. Bildnachweis im Anhang

## 5.12 Zuverlässigkeit der Bilderkennung

### 5.12.1 Der Einfluss von Bildverzerrungen

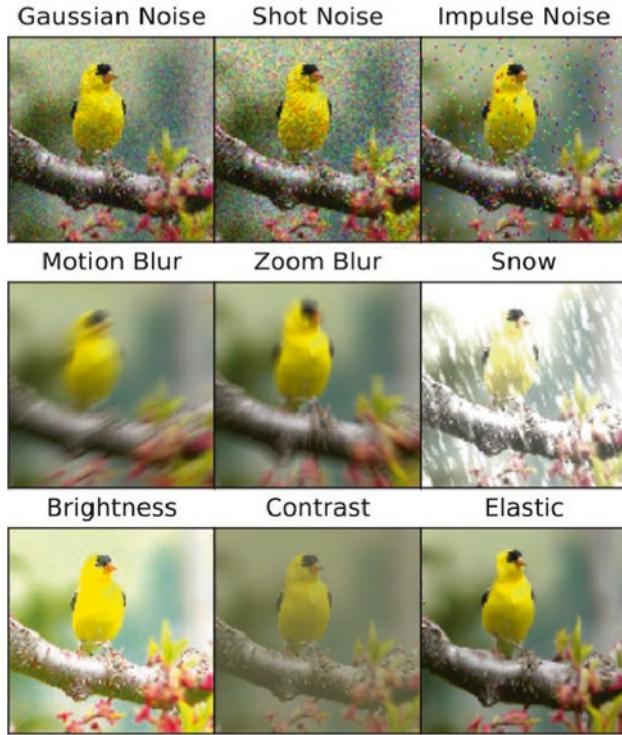
TNN sind sehr erfolgreich bei der Erkennung von Objekten in den verfügbaren Bilddatenbanken. Die Netze werden auf die Erkennungsleistung hin optimiert. Hierbei wird aber nicht auf Robustheit der Erkennung gegenüber im Alltag vorkommenden Modifikationen von Bildern geachtet. Dies ist aber insbesondere in sicherheitsrelevanten Anwendungsberichen – z. B. beim selbstfahrenden Auto – extrem wichtig.

In Abb. 5.44 sind eine Reihe derartiger Modifikationen der Bilder zu sehen, bei denen ein menschlicher Betrachter das gesuchte Objekt trotzdem noch gut erkennen kann (Hendrycks und Dietterich 2019). Andere Arten von Transformationen sind Verschiebungen, Rotation, Kippen und unterschiedliche Skalierungen (Vergrößerungen und Verkleinerungen). Diese wurden auf ImageNet angewendet und teilweise kombiniert und als Benchmark-Daten ImageNet-P und ImageNet-C zur Verfügung gestellt (Hendrycks und Dietterich 2019).

Um die Anfälligkeit der Bildklassifikation gegen solche Verzerrungen zu bestimmen, haben Hendrycks und Dietterich (2019) eine Reihe verschiedenartiger Algorithmen auf ImageNet trainiert. Anschließend wurden die Verfahren auf die Testbeispiele mit unterschiedlichen Verzerrungen angewendet und der mittlere Verzerrungsfehler berechnet



**Abb. 5.43** Die Fotos (oben) werden durch ein Bayessches SegNET in Objektbereiche segmentiert (Mitte). Die untere Reihe zeigt die zugehörigen Unsicherheiten bei der Segmentierung. Bildnachweis im Anhang



**Abb. 5.44** Algorithmisch erzeugte Verfälschungen von Bildern. 9 von 15 Verzerrungen im ImageNet-C Datenbestand. Jede Verzerrung hat fünf Intensitätsstufen (Hendrycks und Dietterich 2019). Bildnachweis im Anhang

(Abb. 5.45). AlexNet (Abschn. 5.5.1) erreicht auf den unverfälschten Daten einen Fehler von 43,5 %. Dieser Wert verdoppelt sich bei den unterschiedlichen Verzerrungen auf Werte um 80 %. ResNet-50 (Abschn. 5.5.2) hat auf den Originaldaten einen Fehler von 23,9 %, welcher sich bei Verzerrungen auf Werte um 65 % erhöht. Obwohl der absolute Fehler bei Verzerrungen für ResNet-50 geringer ausfällt als bei AlexNet, ist die relative Erhöhung von 23,9 % auf 65 % stärker.

Dies zeigt, dass es noch viel zu tun gibt, um die Bilderkennungsverfahren für die Praxis brauchbar zu machen. Ein Ansatz hierzu ist, das TNN auf unverfälschten und zusätzlich auf verzerrten ImageNet-Daten zu trainieren. Allerdings hat das TNN dann weiter hohe Fehlerraten für nicht berücksichtigte Arten von Verzerrungen. Hossain et al. (2018) entwickeln mit dem DCT-Net ein TNN, welches relativ robust gegen viele Arten von Verzerrungen ist, bei nur geringen Einbußen der Genauigkeit auf den unverfälschten Daten. Während bei dem zu Grunde liegenden VGG16-TNN der Originaldatenfehler 7,4 % auf 67,5 % bei Gaussian Noise ansteigt, erhöht sich der Originaldatenfehler des DCT-Net von 12,5 % auf 47,0 % bei Gaussian Noise. Trotzdem bleibt die Verschlechterung der Bildklassifikation durch Verzerrungen ein wichtiges Problem.

**Abb. 5.45** Größe der Fehler bei der Klassifikation von Bildern für zwei Netzwerk-Architekturen bei unterschiedlichen Arten von Verzerrungen auf dem ImageNet-C Benchmark (Hendrycks, et al., 2019)

Verzerrung	AlexNet	ResNet 50
keine	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

Man muss aber auch damit rechnen, dass bei der Anwendung eines fertig trainierten Modells in den Anwendungsdaten Ausreißer auftreten, die das Modell nicht behandeln kann und daher unzuverlässige Ergebnisse liefert. Daher ist in diesem Fall ein Anomalie- oder Ausreißer-Detektor erforderlich, der der Modellanwendung vorgeschaltet wird. Bei der Verwendung einer Einklassen-Support Vektor Maschine ist möglich, diese als neuronales Netz zu interpretieren und damit Ausreißer-Entscheidung erklärbar machen (Kauffmann et al. 2020). Man kann auch mit Hilfe eines generativen adversarialen Netzes (GAN) (Abschn. 9.1.2) einen Diskriminatoren trainieren, der von der Trainingsmenge abweichende Elemente identifiziert (Di Mattia et al. 2019). Derartige Verfahren haben potentiell eine höhere Treffsicherheit, weil der Diskriminatoren beispielsweise trennscharfe CNNs für Bilder verwenden kann.

### 5.12.2 Gezielte Konstruktion von falschklassifizierten Bildern

Das Training von TNN ist nur möglich mit ausgefeilten Optimierungsverfahren, welche Millionen von Parametern so anpassen können, dass das TNN die relevanten Eigenschaften der Trainingsbeispiele extrahieren kann. Umgekehrt kann man aber auch für ein fertig trainiertes TNN einzelne Bilder durch eine gezielte Modifikation der Bildpixel so verändern, dass das Bild durch das Netz in eine falsche Kategorie eingeordnet wird. Die Parameter  $w$  des TNN werden dabei nicht verändert. Ein solches Vorgehen nennt man *adversariale Attacke* (von englisch adversarial: gegnerisch).

Eykholt et al. (2018) haben dieses Vorgehen auf die Erkennung von Verkehrszeichen angewendet. Als Bilderkennungsnetz haben sie das LISA-CNN verwendet, welches drei Convolution-Schichten und eine vollverbundene Schicht aufweist. Es hat eine Genauigkeit von 91 % auf der Testmenge. Um realistisch zu bleiben, haben die Autoren nur Änderungen auf den Verkehrszeichen selbst zugelassen, z. B. in Form von Aufklebern oder Verschmutzung.

hohe Wahrscheinlichkeit von "45mph"
kleine Bildänderung in Sticker-Form

$$\max_d p(y = 45\text{mph} | x + d; w) - \lambda * \|d\|^2$$

maximiere bezüglich  $d$ 



**Abb. 5.46** Bei einer adversarialen Attacke wird ein Bild  $x$  durch eine kleine Änderung  $d$  modifiziert.  $d$  wird so gewählt, dass die Wahrscheinlichkeit der falschen Klasse hoch ist und die Absolutwerte von  $d$  klein sind.  $\lambda$  ist ein Gewichtungsfaktor. Bei der Verkehrszeichenveränderung wurden nur Modifikationen an den Verkehrszeichen zugelassen. Bildnachweis im Anhang

Ziel war es, Stoppschilder so zu verändern, dass sie von dem Bilderkennungsnetz als Geschwindigkeitsbegrenzung auf 45 mph klassifiziert werden. In Abb. 5.46 wird dieses Optimierungsproblem formuliert, wobei die Größe der Änderungen durch die Flächensumme der Quadrate gemessen wird. Dabei wird sichergestellt, dass die Änderungen nur auf dem Verkehrsschild stattfinden und die Form von Aufklebern haben.

Abb. 5.47 zeigt einige Beispiel-Verkehrsschilder, die vom LISA-CNN alle als 45 mph-Geschwindigkeitsbeschränkungen klassifiziert wurden. Dabei war die Falschklassifikation weitgehend unabhängig von dem Blickwinkel auf die Schilder und von der Entfernung. In zusätzlichen Experimenten wurde auch gezeigt, dass es bei Videoaufnahmen aus einem vorbeifahrenden Auto an den modifizierten Verkehrsschildern konsistent zu

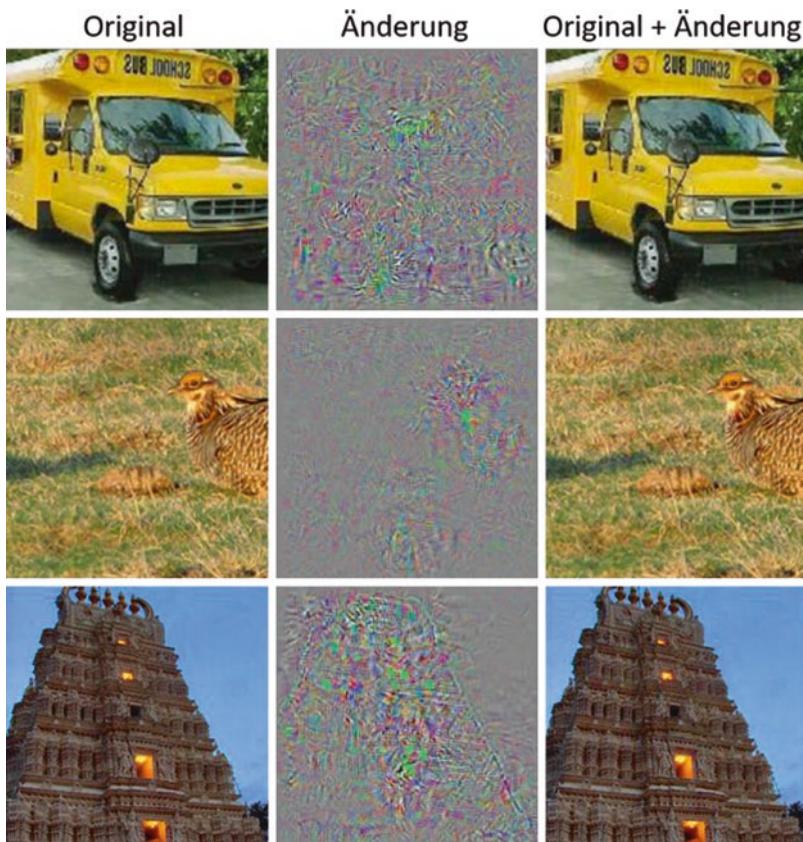


**Abb. 5.47** Diese Grafik zeigt Stoppschilder, die alle vom LISA-CNN als 45mph-Geschwindigkeitsbeschränkung klassifiziert wurden. In der oberen Zeile sind Änderungen auf einem realen Verkehrsschild durch Aufkleber zu sehen, die vom adversarialen Algorithmus berechnet wurden. In der unteren Zeile sind berechnete Änderungen, die wie Verschmutzungen aussehen. Es wird deutlich, dass die adversarialen Änderungen auch bei unterschiedlichen Perspektiven und Entfernungen von Verkehrszeichen funktionieren. Bildnachweis im Anhang

Falschklassifikationen kam. Hingegen wurde das unmodifizierte Stoppschild bei Vorbeifahrten korrekt erkannt.

Die meisten Ansätze dieser Art unterstellen, dass der Angreifer Zugang zu dem trainierten Klassifikationsmodell und den Trainingsdaten hat (white box-Attacke). Es gibt aber auch erfolgreiche Verfahren, die ohne dieses Wissen auskommen, also keinen Zugang zu den Trainingsdaten und dem trainierten Modell haben und auch die Modellarchitektur nicht kennen (black box-Attacke). Dann wird zunächst ein neues Klassifikationsmodell auf ähnlichen Daten trainiert (Xiao et al. 2019). Mit diesem näherungsweisen Modell werden Modifikationen der Eingabebilder erzeugt, welche wieder mit einer hohen Erfolgsrate zu Falschklassifikationen durch das Originalmodell führen.

Kann man sämtliche Bildpixel modifizieren, so sind die für eine adversariale Attacke notwendigen Verzerrungen der Bilder so gering (Abb. 5.48), dass menschliche Betrachter



**Abb. 5.48** In der linken Spalte finden sich die originalen ImageNet-Bilder. Die mittlere Spalte zeigt die subtilen Änderungen, mit denen das linke Bild überlagert wird. In der rechten Spalte findet man die modifizierten Bilder, welche von AlexNet alle als „Strauß“ klassifiziert wurden. Bildnachweis im Anhang

sie als „Rauschen“ ignorieren (Szegedy et al. 2014). Das TNN extrahiert aber hieraus die relevanten Merkmale der neu vergebenen Objektklasse und kommt zu der Falschklassifikation. Offenbar verwendet das TNN oft sehr kleinteilige Details der Bilder zur Klassifikation, während sich Menschen eher an den Konturen eines Objektes orientieren. Modifikationen dieser Art könnten auch in der Praxis missbraucht werden, z. B. für die Täuschung eines Gesichtserkennungssystems zur Zugangskontrolle.

Das Problem sind offenbar Bilder, bei denen durch kleine Variationen sehr unterschiedliche Ausgaben erzeugt werden können. Sokolic et al. (2016) stellen ein Verfahren vor, welches das Ausmaß der Ausgabeänderungen bei kleinen Eingabeänderungen reduziert. Jakubovitz und Giryes (2018) zeigen, dass dieses Vorgehen die Robustheit des Netzes stark erhöht und die Genauigkeit des Ausgangsnetzes nur minimal verringert.

---

## 5.13 Zusammenfassung und Trends

Die Bilderkennung hat die Aufgabe, Objekte in Bildern zu finden und Bilder zu interpretieren. Im einfachsten Fall wird dabei nur festgestellt, ob ein Objekt einer Klasse (z. B. Katze) in einem Bild vorhanden ist. Weitergehende Methoden klassifizieren mehrere Objekte im Bild und lokalisieren die Positionen der Objekte.

Die hierzu verwendete Convolution-Schicht wurde nach biologischen Vorbildern entwickelt. Sie besteht aus einem kleinen rezeptiven Feld (z. B.  $3 \times 3$  Pixel), welches mit einer Parametermatrix linear transformiert wird. Dieses rezeptive Feld wird über die Eingabematrix verschoben und berechnet für jede Position einen Wert, der in der zugehörigen Merkmals- oder Ergebnismatrix gespeichert wird. In jeder Convolution-Schicht werden viele unterschiedliche Merkmalsmatrizen berechnet. Durch Pooling-Schichten wird die Anzahl der Merkmale reduziert, z. B. durch Bildung eines Maximums in einem kleinen rezeptiven Feld. In nachfolgenden Convolution-Schichten werden diese Merkmale weiter transformiert. Ein solches Netz wird Convolutional Neural Network (CNN) genannt.

Auf dem ImageNet-Datensatz wurden derartige Modelle so trainiert, dass sie Bilder gut nach Objektklassen einteilen können. Mittlerweile gibt es fortgeschrittene Netze mit Hunderten von Schichten, die bei der Klassifikation besser sind als Menschen. Die Analyse der CNN-Berechnungen ergibt, dass auf den unteren Schichten einfache Merkmale extrahiert werden, die mit steigender Höhe der Schicht immer komplexer werden.

Die Lokalisierung von Objekten in Bildern kann entweder mit umschließenden Rechtecken durchgeführt werden oder durch Markierung der Pixel eines Objektes. Die Verfahren sind mittlerweile sehr gut in ihrer Erkennungsleistung. Dies ist aber auch für hochentwickelte Verwendungen notwendig, wie z. B. autonome Autos. Gesichter können heute mit hoher Sicherheit erkannt und zugeordnet werden. Für eine Massenüberwachung der Bevölkerung ergeben sich aber immer noch zu viele Fehlzuordnungen.

Ein Problem ist die Anfälligkeit der Verfahren gegenüber Bildverzerrungen. Sie können die Erkennungsleistung drastisch reduzieren. Eine weitere Gefahrenquelle sind adversariale Angriffe, bei denen ein Eindringling versucht, das Erkennungsergebnis zu verfälschen.

schen. Oft bewirkt schon eine kleine Änderung des Bildes, dass das Modell das Bild einer anderen Klasse zuordnet. Allerdings muss der Angreifer dazu meist Zugang zu dem Modell haben. Hier besteht noch ein großer Bedarf an verbesserten Abwehrmaßnahmen.

### Trends

- Beim Beantworten von Fragen zu Bildern (Visual Question Answering) werden textuelle Fragen zu einem Bild gestellt, die dann in Textform beantwortet werden müssen, z. B.: Worauf sitzt das Kind? Es ist zu erwarten, dass die Beantwortung derartiger Fragen im Zentrum der Entwicklung stehen wird. Dabei wird es vermutlich einen integrierten Ansatz unter Verwendung von Modellen für Text, Bild und Sprache geben.
- Selbstüberwachtes Lernen (S. 42) hat im Bereich der Sprachverarbeitung zu nachhaltigen Erfolgen geführt (S. 148). Dabei wird z. B. aus den bisherigen Wörtern eines Textes jeweils das nächste Wort prognostiziert. Es ist zu erwarten, dass dieser Ansatz auch vermehrt im Bild- und Videobereich angewendet wird.
- In der Industrie werden verstärkt KI-Systeme zur visuellen Qualitätskontrolle von medizinischen Geräten, der Produktion von Pharmazeutika, Nahrungsmitteln und in der Automobilindustrie eingesetzt.

---

## Literatur

- Cui, Y., et al. 2018.** Large scale fine-grained categorization and domain-specific transfer learning. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4109–4118). 2018.
- Deng, J., et al. 2009.** ImageNet: A large-scale hierarchical image database. *CVPR 2009 – IEEE Conference on Computer Vision and Pattern Recognition*. S. 248–255. 2009.
- Di Mattia, F., et al. 2019.** A survey on gans for anomaly detection. [Online] 2019. [Zitat vom: 20. 04. 2020.] arXiv preprint arXiv:1906.11632.
- Eslami, S. A., et al. 2018.** Neural scene representation and rendering. *Science*, 360(6394), 1204–1210. 2018.
- 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. 2017.
- Eykholz, K., et al. 2018.** Robust physical-world attacks on deep learning visual classification. *Proc. ICPR pp. 1625–1634*. 2018.
- Fukushima, K. 1980.** Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 93–202. 1980.
- Gal, Yarin. 2016.** *Uncertainty in Deep Learning – Thesis*. University of Cambridge : s.n., 2016.
- Goodfellow, Ian, Bengio, Yoshua und Courville, Aaron. 2016.** *Deep Learning*. s.l. : MIT Press, 2016.
- Hastie, Trevor, Tibshirani, Robert und Friedman, Jerome. 2009.** *The Elements of Statistical Learning*, 2. Ausgabe. s.l. : Springer, 2009.

- He, K., et al. 2016.** Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778). 2016.
- Hendrycks, D. und Dietterich, T. 2019.** Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *ICLR 2019*. 2019.
- Hossain, M. T., et al. 2018.** Distortion robust image classification with deep convolutional neural network based on discrete cosine transform. [Online] 2018. [Zitat vom: 30. 05. 2019.] arXiv preprint arXiv:1811.05819.
- Huang, G., et al. 2017.** Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700–4708). 2017.
- Hubel, D. H. und Wiesel, T. N. 1962.** Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1), 106–154. 1962.
- Jakubovitz, D. und Giryes, R. 2018.** Improving DNN Robustness to Adversarial Attacks using Jacobian Regularization. *ECCV 2018*. 2018.
- Jordan, J. 2018.** An overview of semantic image segmentation. [Online] 2018. <https://www.jeremy-jordan.me/semantic-segmentation/>.
- Kauffmann, J., Müller, K. R. und Montavon, G. 2020.** Towards explaining anomalies: a deep Taylor decomposition of one-class models. *Pattern Recognition*, 107198. 2020.
- Keras. 2019.** *Keras CNN for MNIST*. [Online] 2019. [https://github.com/keras-team/keras/blob/master/examples/mnist\\_cnn.py](https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py).
- Krämer, T. 2016.** *Von Sehstrahlen und schwebenden Bildern*. [Online] 2016. <https://www.dasgehirn.info/wahrnehmen/sehen/von-sehstrahlen-und-schwebenden-bildern>.
- Krizhevsky, A., Sutskever, I. und Hinton, G. E. 2012.** Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105). 2012.
- LeCun, Y., et al. 1998.** Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. 1998.
- Lee, J., et al. 2019.** Biobert: pre-trained biomedical language representation model for biomedical text mining. [Online] 2019. [Zitat vom: 16. 07. 2019.] arXiv preprint arXiv:1901.08746.
- Li, F., Johnson, J. und Yeung, S. 2017.** Lecture 11: Detection and Segmentation. [Online] 2017. [http://cs231n.stanford.edu/slides/2017/cs231n\\_2017\\_lecture11.pdf](http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf).
- Li, Fei-Fei, Johnson, Justin und Yeung, Serena. 2018.** *CNNs for Visual Recognition. Lecture 2: Image Classification pipeline*. s.l. : Stanford University, 2018.
- Marcus, G. 2018.** Deep learning: A critical appraisal. [Online] 2018. [Zitat vom: 21. 05. 2019.] arXiv preprint arXiv:1801.00631.
- Noh, H., S. Hong und Han, B. 2015.** Learning Deconvolution Network for Semantic Segmentation. *Proceedings of the IEEE international conference on computer vision* (pp. 1520–1528). 2015.
- Noy, A., et al. 2019.** ASAP: Architecture search, anneal and prune. [Online] 25. 09. 2019. arXiv preprint arXiv:1904.04123.
- Phillips, P. J., 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. 2018.
- Ranjan, R., et al. 2018.** Deep learning for understanding faces: Machines may be just as good, or better, than humans. *IEEE Signal Processing Magazine*, 35(1), 66–83. 2018.
- Redmon, J. und Farhadi, A. 2017.** YOLO9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263–7271). 2017.
- Redmon, J., et al. 2016.** You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788). 2016.
- Ronneberger, O., Fischer, P. und Brox, T. 2015.** U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention* (pp. 234–241). Springer, Cham. 2015.

- Russakovsky, O., et al. 2015.** *International Journal of Computer Vision*, 115(3), 211–252. 2015.
- Sokolic, J., et al. 2016.** Margin Preservation of Deep Neural Networks. [Online] 2016. arXiv, abs/1605.08254, 2016.
- Szegedy, C., et al. 2015.** Going deeper with convolutions. *IEEE conference on computer vision and pattern recognition* (pp. 1–9). 2015.
- Szegedy, C., et al. 2014.** Intriguing properties of neural networks. [Online] 2014. [Zitat vom: 30. 05. 2019.] arXiv preprint arXiv:1312.6199.
- Teye, M., Azizpour, H. und Smith, K. 2018.** Bayesian uncertainty estimation for batch normalized deep networks. [Online] 2018. [Zitat vom: 09. 06. 2019.] arXiv preprint arXiv:1802.06455.
- Touvron, H., et al. 2019.** Fixing the train-test resolution discrepancy. [Online] 19. 07. 2019. [Zitat vom: 06. 10. 2019.] <https://arxiv.org/pdf/1906.06423v2.pdf>.
- Tutani, G. 2018.** Practical image segmentation with Unet. [Online] 2018. <https://tuatini.me/practical-image-segmentation-with-unet/>.
- Van Horn, G., et al. 2018.** The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8769–8778). 2018.
- Visual. 2020.** Visual cortex. *Wikipedia*. [Online] 2020. [Zitat vom: 25. 01. 2020.] [https://en.wikipedia.org/wiki/Visual\\_cortex](https://en.wikipedia.org/wiki/Visual_cortex).
- Wang, C., Li, M. und Smola, A. J. 2019.** Language Models with Transformers. [Online] 2019. [Zitat vom: 27. 05. 2019.] arXiv preprint arXiv:1904.09408.
- Welling, M. und Teh, Y. W. 2011.** Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 681–688). 2011.
- Wu, L. Y., et al. 2018.** Starspace: Embed all the things! *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- Xiao, C., et al. 2019.** Generating adversarial examples with adversarial networks. [Online] 14. 02. 2019. [Zitat vom: 15. 06. 2019.] arXiv preprint arXiv:1801.02610.
- Yalniz, I. Z., et al. 2019.** Billion-scale semi-supervised learning for image classification. [Online] 2019. [Zitat vom: 21. 10. 2019.] arXiv preprint arXiv:1905.00546.
- You, Y., et al. 2018.** Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing* (p. 1). ACM. 2018.
- Zagoruyko, Sergey, und Komodakis, Nikos. 2016.** Wide residual networks. [Online] 2016. arXiv preprint arXiv:1605.07146.
- Zeiler, M. D. und Fergus, R. 2014.** Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer, Cham. 2014.
- Zhu, Y., et al. 2019.** Improving Semantic Segmentation via Video Propagation and Label Relaxation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8856–8865). 2019.



# Erfassung der Bedeutung von geschriebenem Text

6

Sprache ist das häufigste Kommunikationsmedium zwischen Menschen (Abb. 6.1). Die Verarbeitung natürlicher Sprache (Natural Language Processing, NLP) beschäftigt sich mit der Entwicklung von Algorithmen, welche automatisch natürliche menschliche Sprache verstehen und erzeugen. Dazu gibt es eine sehr breite Palette von Anwendungen, etwa Suchmaschinen für Webseiten, maschinelle Übersetzung von einer Sprache in eine andere, Erkennung von gesprochener Sprache oder das Führen von Dialogen. In diesem Kapitel werden wir uns mit der Verarbeitung geschriebener Sprache beschäftigen, im nächsten Kap. 7 mit der Analyse gesprochener Sprache.

Geschriebene Texte liegen immer als Sequenzen vor, z. B. der Sequenz von Wörtern in einem Satz oder Dokument. Verfahren zur Textverarbeitung müssen derartige Sequenzen gut auswerten können. Ein anderes Charakteristikum ist die Vielzahl von Wörtern, die in einer Sprache existieren. Das „Deutsche Wörterbuch“ der Brüder Grimm umfasst einen Wortschatz von mindestens 320.000 Wörtern. Zudem erlaubt das Deutsche die kreative Bildung neuer Wörter, z. B. als Komposita. Die Berücksichtigung so vieler Merkmale stellt ganz neue Anforderungen an die verwendeten Analysemethoden und maschinellen Lernverfahren.

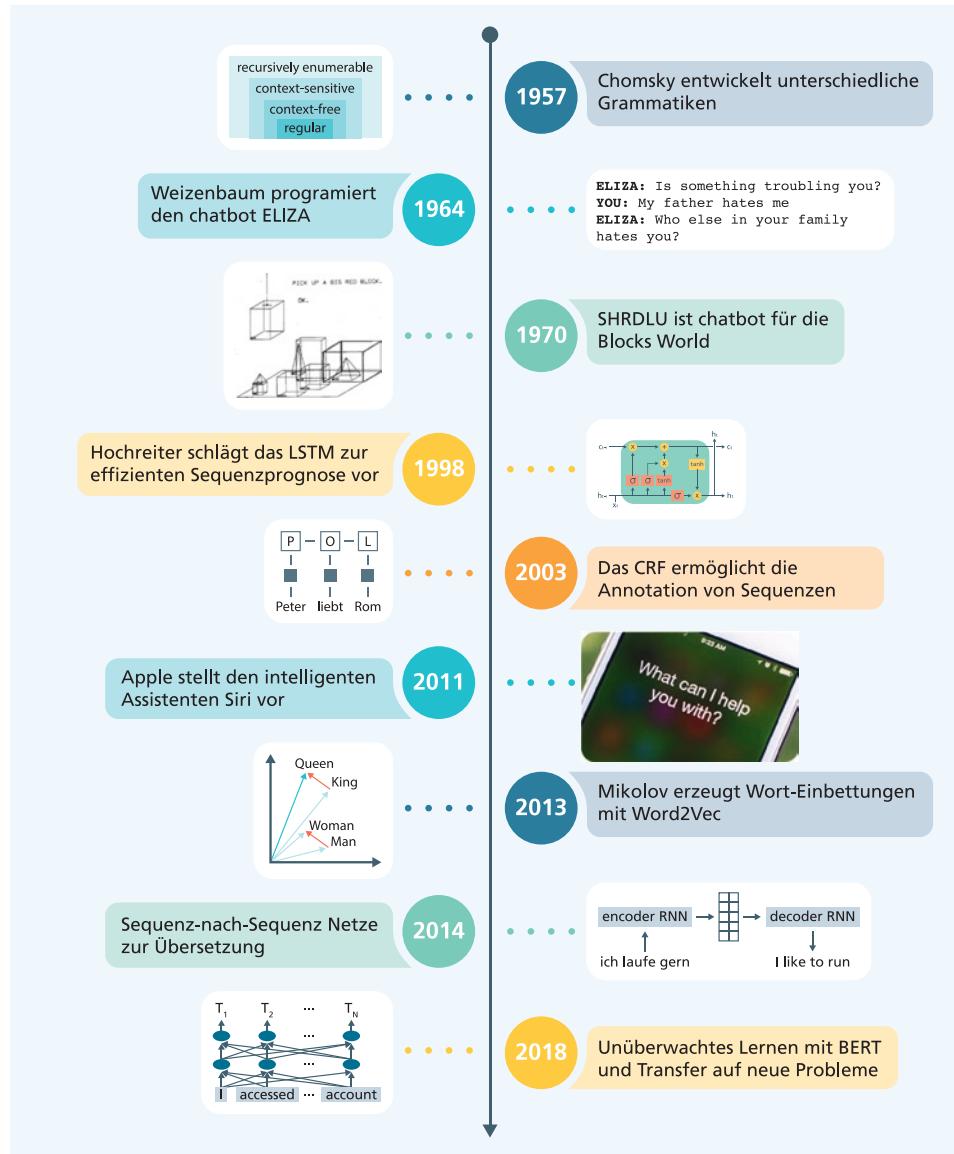
Die Entwicklung der automatischen Verarbeitung von Texten hat eine lange Geschichte (Abb. 6.2). Ab der Mitte des letzten Jahrhunderts entwickelte Chomsky Grammatiken, um die formale grammatische Struktur der Sprache zu erfassen. Es werden erste Chatbots für Dialoge zu kleinen Themenbereichen entwickelt, welche Regeln verwenden. Als spezielle tiefe neuronale Netze für Sequenzen werden rekurrente neuronale Netze vorgeschlagen, wobei das von Hochreiter und Schmidhuber 1998 entwickelte LSTM besonders lange

**Abb. 6.1** Mit natürlicher Sprache können Menschen mühelos miteinander kommunizieren. Dabei können die unterschiedlichsten Medien und Kommunikationskanäle genutzt werden. Bildnachweis im Anhang



Zusammenhänge erfassen kann. Eine wichtige Idee zur Sprachverarbeitung ist die Repräsentation eines Wortes durch einen Vektor von Zahlen, der Embedding genannt wird. Dabei sollen Wörter mit ähnlicher Bedeutung auch ähnliche Embeddings besitzen. Mikolov stellt 2013 ein einfaches Verfahren zur Berechnung von Embeddings vor. Mit dem Aufkommen der Grafikprozessoren (GPU) können immer komplexere Probleme mit LSTMs gelöst werden, z. B. die automatische Übersetzung im Jahre 2014. In diesem Zusammenhang wird der Attention-Mechanismus entwickelt, welcher eine inhaltliche Beziehung zwischen Wörtern herstellt. Darauf beruhen Transformer und BERT, welche ab 2018 die Lösung neuer Probleme in der semantischen Analyse von Sprache ermöglichen.

In jüngster Zeit wurden Ansätze für die Verarbeitung natürlicher Sprache entwickelt, die aus großen Mengen nicht-annotierter Texte die innere Struktur von Sprache lernen. Sie erwerben profunde Kenntnisse über die Grammatik sowie über inhaltliche Zusammenhänge, einschließlich einem begrenzten Weltwissen. Diese Modelle, z. B. BERT, lassen sich für eine Anzahl von Aufgaben ohne speziell annotierte Trainingsdaten verwenden. Wir werden diese Modelle am Schluss dieses Kapitels (Abschn. 6.7) darstellen. Zuvor werden wir eine Reihe von überwachten und unüberwachten Modellen vorstellen, die unterschiedliche Probleme des NLP lösen.



**Abb. 6.2** Die Geschichte der automatischen Verarbeitung geschriebener natürlicher Sprache. Bildnachweis im Anhang

## 6.1 Wie kann man die Bedeutung von Wörtern durch Vektoren darstellen?

Ein Symbol ist ein „Erkennungszeichen“, also ein Schrift- oder Bildzeichen für eine Idee, einen Gegenstand, einen Vorgang usw. Die klassische Verarbeitung natürlicher Sprache beruhte auf der Verwendung von Symbolen für Wörter und Relationen:

- Wissen wird explizit dargestellt durch Symbole für Wörter (Begriffe) und Relationen. Dabei stellt eine Relation eine Beziehung zwischen Begriffen her. Einige Aspekte eines Restaurants können beispielweise durch die Wort- und Relationssymbole in Abb. 6.3 beschrieben werden. Die letzte Angabe beschreibt die Regel: wenn Person X eine Speise Y bestellt und Z Kellnerin im Restaurant ist, so serviert Z die Speise Y für Person X.
- Ergebnisse werden durch den Vergleich der Wortsymbole und die Aneinanderkettung von Regeln zu einer Schlussfolgerung mit Hilfe der Logik berechnet. Aus der „Wissensdatenbank“ in Abb. 6.3 kann man beispielweise schließen: „serviert(Carol, Bob, Pizza)“ und „serviert(Carol, Alice, Sushi)“.
- Die Sprachverarbeitung zielte darauf, natürlichsprachige Texte in Wissensdatenbanken zu überführen, um dann mit Hilfe der Logik daraus Schlussfolgerungen zu ziehen.

Ein grundlegendes Problem dieses Ansatzes ist aber, dass die eigentliche Bedeutung der Wörter (z. B. „im“, „arbeitet\_im“, „Restaurant“, „Carol“, ...) nicht bekannt ist. Es sind einfach nur Symbole, die die Reichhaltigkeit des natürlichen Sprachgebrauchs nicht erfassen. Vielmehr ist es erforderlich, dass die Wortsymbole in der realen Welt verankert werden müssen, das sogenannte Symbol-Grounding. Man muss also zum Beispiel das Symbol „Restaurant“ mit realen Restaurants und deren Eigenschaften in Verbindung bringen. Es gilt also, die Bedeutung alternativer Formulierungen inhaltlich in Beziehung zu setzen.

Bei der Arbeit mit Wissensbasen wie in Abb. 6.3 hat sich herausgestellt, dass man sehr viele Fakten und Regeln benötigt (sogenanntes Weltwissen), um eine brauchbare Beschreibung von realen Anwendungsbereichen zu erhalten. Beispielsweise muss man berücksichtigen, dass Carol nur innerhalb ihrer Schicht im Restaurant serviert. Andererseits stellte sich heraus, dass zusätzliche Regeln und Fakten oft zu inneren Widersprüchen in



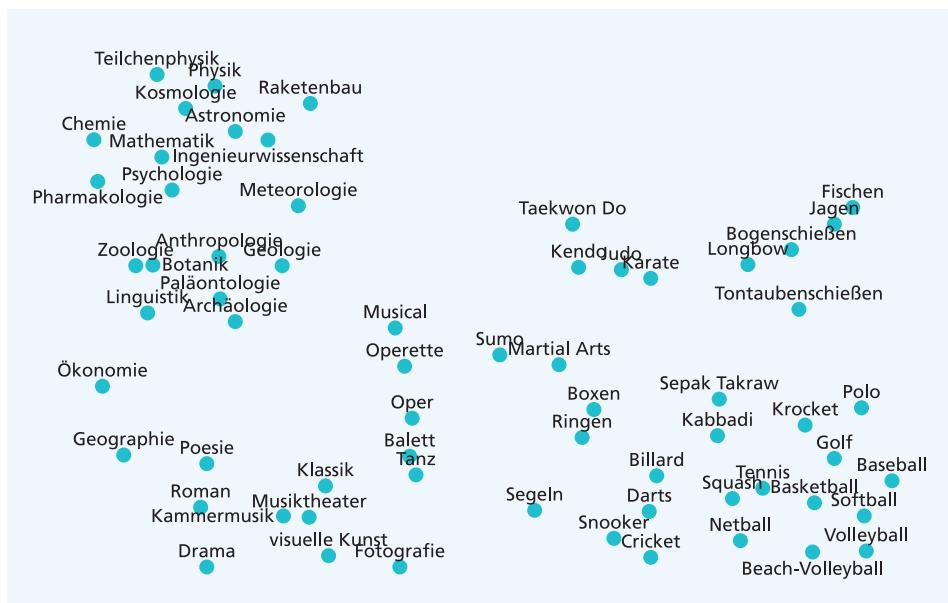
**Abb. 6.3** In einer symbolischen Wissensdatenbank wird Wissen durch Wortsymbole gekennzeichnet und diese sind durch Relationen verbunden

der Wissensdatenbank führen. Daher ist die Verwendung von symbolischen Wissensdatenbanken nicht besonders erfolgreich.

Um die Bedeutung von Wörtern besser beschreiben zu können, hat sich in den letzten Jahren eine neue Art der Informationsverarbeitung etabliert:

- Wissen (Wörter, Sätze) wird implizit dargestellt durch große Vektoren von reellen Zahlen. Wörter mit ähnlicher Bedeutung werden auf Vektoren abgebildet, die einen geringen Vektorabstand im mathematischen Sinne haben. Der semantische Gehalt von Wörtern und Sätzen wird dadurch vergleichbar.
- Diese Bedeutungsvektoren werden durch tiefe neuronale Netze abgeleitet, die die Vektoren als Eingaben verwenden. Sie basieren auf der Vorstellung, dass bedeutungsähnliche Wörter meist in ähnlichen Satzumgebungen auftauchen.
- Die notwendigen Modelle können mit einfachen Optimierungsverfahren trainiert werden, das Zustandekommen eines Ergebnisses ist aber nur schwer erklärbar.

In Abb. 6.4 wurden solche hochdimensionalen Repräsentationen von Wörtern auf zwei Dimensionen so projiziert, dass die Nähe von Vektoren weitgehend erhalten bleibt. Dabei sind nur die Nähe und Distanz der Begriffe relevant, die Dimensionen selbst haben keine direkte inhaltliche Bedeutung. Rechts sind Sportarten, in der Mitte Theaterformen und



**Abb. 6.4** Wörter lassen sich durch Vektoren repräsentieren, wobei die Vektoren bedeutungsähnlicher Wörter einen geringen Vektorabstand haben. Diese Abbildung enthält eine Projektion dieser hochdimensionalen Embeddingvektoren auf Vektoren der Länge 2, die einem Punkt in der 2-dimensionalen Ebene entsprechen. Rechts oben befinden sich unterschiedliche Sportarten, links oben Wissenschaftsdisziplinen und links unten Kunstbegriffe

links Wissenschaftsbezeichnungen genannt. Stehen zwei Begriffe nahe beieinander, so sind sie sehr ähnlich in ihren Bedeutungen.

### 6.1.1 Das Konzept der Embeddingvektoren

Die Schreibweise von Wörtern sagt fast nichts über ihre Bedeutung aus. Die Wörter „Apfelsine“ und „Orange“ haben eine vollständig unterschiedliche Schreibweise, drücken aber den selben Begriff aus. Solche Wörter mit gleicher oder sehr ähnlicher Bedeutung werden Synonyme genannt. Da die Verarbeitung natürlicher Sprache danach strebt, den Sinn der Wörter zu erfassen, wird eine Repräsentation für Wörter gesucht, welche für ähnliche Wörter nahezu gleich ist und für semantisch unterschiedliche Wörter verschieden ist.

Ein Ansatz zur automatischen Bestimmung einer solchen Repräsentation wurde schon früh von Firth (1957) entwickelt. Er stellte fest, dass „ein Wort charakterisiert wird durch die Gesellschaft, die es hat“. Das bedeutet, dass der Sinn eines Wortes durch seine Nachbarwörter gekennzeichnet wird. Abb. 6.5 enthält Beispielsätze, in denen die Begriffe Apfelsine, Orange und Zitrone ausgetauscht werden können. Dies ist ein Hinweis darauf, dass diese Begriffe semantisch ähnlich sind.

Wir haben in Abschn. 4.3.1 gesehen, dass tiefe neuronale Netze die Eingabemerkmale in neue Vektoren oder Tensoren transformieren, die eine geeigneter Repräsentation der Eingaben für die Lernaufgabe bilden. Daher liegt es nahe, dies auch mit Wörtern zu versuchen, also einen Vektor als Repräsentation eines Wortes zu verwenden.

Im Folgenden wird jedes Wort  $v$  durch einen Vektor  $emb(v)$  von reellen Zahlen repräsentiert, den Embeddingvektor (Einbettungsvektor) oder kurz Embedding. Die Dimension  $k$  des Vektors wird zuvor gewählt, z. B.  $k = 100$ . Das Ziel ist nun, für jedes Wort  $v$  die Werte des zugehörigen Vektors  $emb(v)$  so festzulegen, dass die Vektoren von **ähnlichen** Begriffen im  $k$ -dimensionalen Raum **nahe** beieinanderliegen.

In Abb. 6.6 ist dies in drei Dimensionen für die Begriffe „Orange“ und „Apfelsine“ symbolisiert. Der Begriff „Apfel“ ist zwar eine Obstsorte, aber verschieden von „Orange“ und hat ein Embedding, welches weiter entfernt ist als „Apfelsine“. Die Richtung der Verbindungsgeraden zwischen zwei Begriffen, z. B. „Orange“ und „Orangenbaum“, kann in-

**Abb. 6.5** Die Begriffe Apfelsine, Orange und Zitrone können in diesen drei Sätzen ausgetauscht werden. Dies ist ein Anhaltspunkt für ihre semantische Ähnlichkeit. Bildnachweis im Anhang

Die **Apfelsine** ist sehr saftig.  
Karl pflückte die **Apfelsine** vom Baum.  
Eine **Apfelsine** enthält viel Vitamin C.

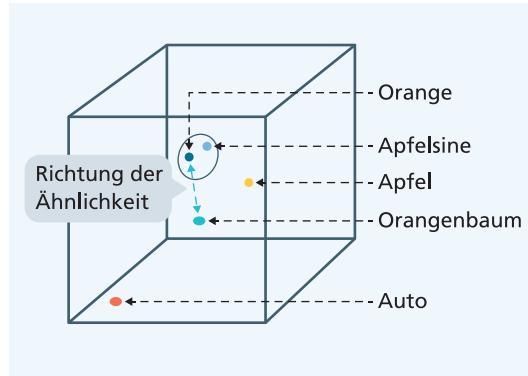


Die **Orange** ist sehr saftig.  
Karl pflückte die **Orange** vom Baum.  
Eine **Orange** enthält viel Vitamin C.



Die **Zitrone** ist sehr saftig.  
Karl pflückte die **Zitrone** vom Baum.  
Eine **Zitrone** enthält viel Vitamin C.





**Abb. 6.6** Dreidimensionale Darstellung der Einbettungen von Wörtern. Bedeutungsähnliche Begriffe liegen nahe beieinander, andere sind weiter entfernt, je nach dem Grad der Bedeutungsähnlichkeit. Die Richtung der Verbindungslinie zwischen zwei Einbettungen sagt etwas über die inhaltliche Beziehung der Begriffe aus und ist eine „Richtung der Ähnlichkeit“

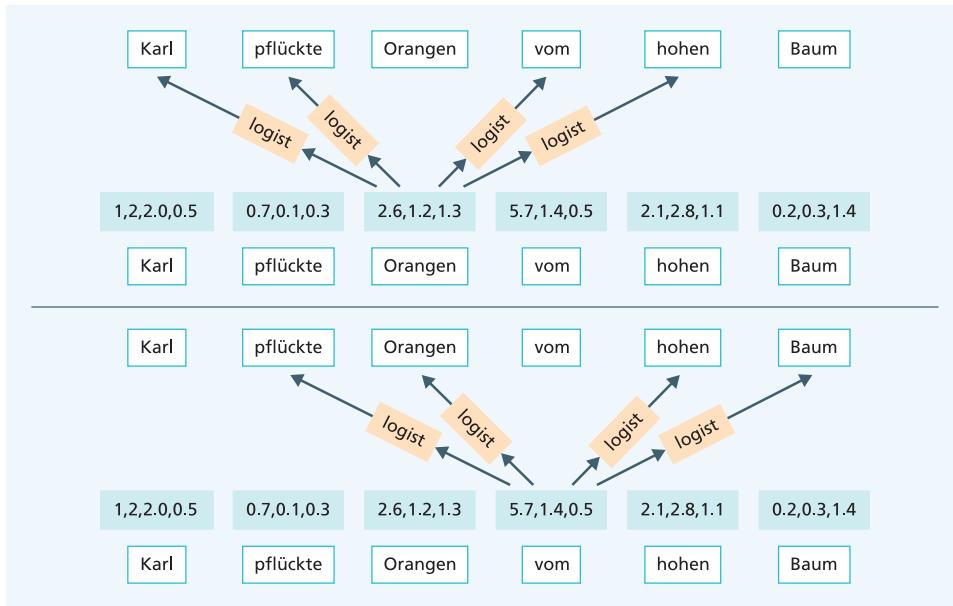
haltlich interpretiert werden als eine „Richtung der Ähnlichkeit“ und symbolisiert die Relation zwischen Obstbaum und seiner Frucht. Noch weiter entfernt sind die Embeddings sehr verschiedener Begriffe wie etwa von „Auto“ (Abb. 6.6 und 6.7).

Insgesamt sind Embeddings eine Möglichkeit, Wörter so zu kodieren, dass sie in TNN verarbeitet werden können. Alle folgenden neuronalen Modelle gehen nach diesem Muster vor. Yoshua Bengio hat wichtige Grundlagen für die Ableitung von Embeddings durch neuronale Netze gelegt (Abb. 6.8).

### 6.1.2 Berechnung von Embeddingvektoren mit Word2vec

Word2Vec ist ein Embedding-Netz, ein einfacher, aber sehr effizienter Ansatz zur Bestimmung von Embeddings. Das Verfahren verwendet einen großen Trainingsdatenbestand von Texten. Für jedes Wort  $v$  im Vokabular  $V$  dieser Trainingsdaten wird ein eigener Embeddingvektor  $\text{emb}(v)$  der Länge  $k$  definiert, welcher zu Beginn zufällig mit reellen Zahlen zwischen  $-1,0$  und  $1,0$  gefüllt wird. In der Regel wird eine Embeddinglänge  $k$  zwischen 100 und 1000 gewählt.

Dazu muss zunächst die Menge der Wörter, das Vokabular einer Textsammlung, zusammengestellt werden. Hierzu wird eine Liste der unterschiedlichen Wörter in dem Text gebildet und nach ihrer Häufigkeit sortiert, wobei Wörter mit unterschiedlichen Endungen als unterschiedlich zählen. Das können mehrere hunderttausend Wörter sein, wobei sehr viele Wörter selten sind, d. h. nur ein- oder zweimal auftauchen. Diese Beobachtung gilt für fast alle Sprachen. Einen Embeddingvektor kann man nur dann zuverlässig bestimmen, wenn das Wort mehrfach auftaucht. Daher lässt man seltene Wörter aus dem Vokabular weg und ersetzt sie in dem Text durch einen speziellen Platzhalter, z. B. „UNK“. Trotzdem enthält eine Textsammlung dann immer noch sehr viele unterschiedliche Wörter, z. B.  $N = 50.000$ .



**Abb. 6.7** Auf den Embeddingvektor des zentralen Wortes  $v^{(t)} = \text{„Orangen“}$  wird das logistische Regressionsmodell angewendet (oben). Hiermit werden die Wahrscheinlichkeiten der Wörter in der Nachbarschaft von „Orangen“ prognostiziert. Auf dem unteren Bild wird die Prognose der Nachbarwörter für das nächste Wort  $v^{(t+1)} = \text{„vom“}$  durchgeführt. Analog wird dies für alle Wörter des Trainingsdatensatzes wiederholt. Die unbekannten Parameter  $A$ ,  $b$ ,  $\text{emb}(v)$  werden durch Optimierung so verändert, dass die Wahrscheinlichkeiten der Nachbarwörter insgesamt möglichst hoch werden



**Yoshua Bengio** studierte Informatik und war als PostDoc am MIT unter Michael Jordan. In den 90er-Jahren kombinierte er TNN mit probabilistischen Modellen, die später zur Handschrifterkennung genutzt wurden. 1993 wechselte er an die Uni Montreal. Danach schlug er hochdimensionale Wort-Embeddings vor, welche heute überall verwendet wurden. Er war an der Entwicklung von Attention-Mechanismen beteiligt, die bei der Sprachübersetzung eine große Rolle spielen. Er war auch an der Entwicklung der generativen adversarialen Netze (GAN) beteiligt, mit welchem sich z. B. aus den verfügbaren Trainingsbildern neuartige Bilder erzeugen lassen. Er nimmt an, dass in Zukunft KI-Systeme erforderlich sind, welche Ursachen und Wirkungen erkennen können. Im Jahr 2019 erhielt er den Turing-Preis, den renommieritesten Wissenschaftspris der Informatik.

**Abb. 6.8** Yoshua Bengio, \*1964 in Paris, hat die Entwicklung neuronaler Netze und der Repräsentation von Objekten maßgeblich vorangebracht. Bildnachweis im Anhang

Nach den Überlegungen von Firth (1957) sind benachbarte Wörter für die Bedeutung eines Wortes relevant. Für den Beispielsatz „Karl pflückte Orangen vom hohen Baum“ besteht die 2er-Nachbarschaft des Wortes „Orangen“ aus den Wörtern „Karl“, „pflückte“, „vom“ und „hohen“ (Abb. 6.7). Offenbar charakterisieren diese Wörter das Wort Orange in gewisser Weise, z. B. deutet „pflückte“ an, dass „Orange“ ein Ding ist, welches geerntet und gepflückt werden kann.

Das Word2vec-Modell prognostiziert nun die Wahrscheinlichkeiten der Wörter  $v^{(t+i)}$  in der Nachbarschaft des **zentralen Wortes**  $v^{(t)}$  (z. B. „Orangen“) mithilfe eines logistischen Regressionsmodells (Abschn. 3.4):

$$p(v^{(t+i)}|v^{(t)}) = \text{softmax}\left(A * \text{emb}(v^{(t)}) + b\right)$$

Dabei bezeichnet  $v^{(t)}$  das Wort an der Position  $t$ . Wählt man als Vokabulargröße  $N = 50.000$ , so erhält man ein Klassifikationsmodell mit einer riesigen Anzahl von 50.000 Klassen, und das Modell berechnet für jedes mögliche Nachbarwort eine Wahrscheinlichkeit.

Typischerweise verwendet man 5-er bis 10-er Nachbarschaften. Dieses Modell enthält als unbekannte Parameter die Matrix  $A$  und den Vektor  $b$  sowie für jedes Wort  $v$  den Embeddingvektor  $\text{emb}(v)$ . Die Parameter  $A$  und  $b$  sind gleich für die Prognose aller Nachbarschaftswörter. Die Dimensionen dieser Parameter sind für  $\text{emb}(v) : k \times N$ , für  $A : N \times k$ , und für  $b : N$ . Wählt man  $k = 100$ , so hat das Modell insgesamt mehr als 10 Millionen freie Parameter, kann also potentiell viele Eigenschaften der Wörter darstellen. Die Kostenfunktion verlangt wie in Abschn. 3.5.2, dass die Wahrscheinlichkeit der prognostizierten Nachbarwörter möglichst hoch werden soll (Mikolov et al. 2013).

Word2vec ist ein Beispiel für selbstüberwachtes Lernen (Abschn. 3.1.2). Hierbei wird eine Komponente der beobachteten Daten aus den restlichen Komponenten prognostiziert. Es werden keine expliziten Annotationen benötigt. Das Modell ignoriert die genaue Reihenfolge der Wörter, es wird lediglich die Nachbarschaft der Wörter berücksichtigt. Trotzdem kann das Modell viele Informationen über die innere Struktur eines Textes extrahieren und für die geplanten Anwendungen verwenden. Dieses Prinzip wird später für viele Anwendungen in der Sprachverarbeitung verwendet.

### 6.1.3 Die Approximation der Softmaxfunktion reduziert den Rechenaufwand

Die Parameter des Verfahrens werden wie üblich mithilfe des stochastischen Gradientenabstiegs (SGD) optimiert. Die Parameter werden daher so verändert, dass die Wahrscheinlichkeit der Wörter, die in der Umgebung des zentralen Wortes auftauchen, möglichst hoch wird. Dabei ist zu berücksichtigen, dass meist viele verschiedene Wörter in der Nachbarschaft beobachtet werden, die Wahrscheinlichkeiten also nicht hoch sein können. Allerdings wird die Wahrscheinlichkeit von Wörtern sehr klein, wenn sie niemals in der Umgebung eines zentralen Wortes erscheinen.

Dabei gibt es eine Schwierigkeit, welche durch die hohe Anzahl alternativer Wörter verursacht wird. Die lineare Transformation des logistischen Modells

$$u^{(t)} = A * \text{emb}(v^{(t)}) + b$$

erzeugt einen Punktewert  $u_i^{(t)}$  für **jedes** der  $N$ , z. B.  $N = 50.000$ , möglichen Wörter. Um z. B. die Wahrscheinlichkeit des Wortes  $v^{(t-2)} = \text{Karl}$  in der Nachbarschaft durch die Softmaxfunktion auszurechnen, müssen im Nenner des Bruches  $N = 50.000$  Terme berechnet und addiert werden, was einen sehr hohen Rechenaufwand verursacht. Als Alternative kann man einen Näherungswert für die Softmaxfunktion berechnen, bei dessen Bestimmung zunächst eine Stichprobe  $S$  von wenigen (5–20) Wörtern zufällig ausgewählt, und die Summe über die zugehörigen Terme im Nenner bestimmt wird.

$$\frac{\exp(u_{i(\text{Karl})}^{(t)})}{\exp(u_1^{(t)}) + \dots + \exp(u_{50000}^{(t)})} \approx \frac{\exp(u_{i(\text{Karl})}^{(t)})}{\exp(u_{i(\text{Karl})}^{(t)}) + \sum_{m \in S} \exp(u_m^{(t)})}$$

Index von Karl

Summe der Punktewerte aller Worte

Summe über eine zufällige Stichprobe der Worte

Dieses Verfahren heißt „Noise Contrastive Estimation“ (Dyer 2014) und wird sehr häufig bei der Berechnung der Softmaxfunktion für viele Klassen verwendet. Der Rechenaufwand ist hierbei um Größenordnungen geringer. Die zusätzliche Ungenauigkeit bei der Gradientenbestimmung ist nicht schwerwiegend, da für die Optimierung ein stochastischer Gradientenabstieg genutzt wird (Abschn. 3.6.6), welcher selbst schon ungenaue Gradienten liefert.

## 6.2 Eigenschaften der Embedding-Vektoren

Das Word2vec-Verfahren wurde von uns für die deutsche Wikipedia erprobt. Diese umfasst aktuell etwa 2,2 Millionen Artikel mit 971 Millionen Wörtern und erfordert 6,2 GB Speicher. Es wurde ein Vokabular von etwa 2 Millionen Wörtern gebildet und die Länge der Embeddings auf 100 festgelegt. Das Training des Modells erforderte etwa 6 Stunden. Wichtig ist, dass das Verfahren keinerlei Vorgaben verwendet und für beliebige Sprachen und Alphabete funktioniert.

Als Ergebnis des Trainings erhält man für jedes Wort des Vokabulars einen Embedding-Vektor. Die Komponenten dieses Vektors haben keine Bedeutung an sich, sondern sind nur im Vergleich mit den Embeddings anderer Wörter zu interpretieren.

### 6.2.1 Nächste Nachbarn von Embeddings haben ähnliche Bedeutungen

Als eine erste Evaluierung kann man zu einem Wort das zugehörige Embedding nehmen und die nächsten Nachbarn im 100-dimensionalen Raum suchen. Als Abstandsmaß wählt

man meist die Kosinus-Distanz, welche den Winkel zwischen zwei Vektoren  $u = (u_1, \dots, u_k)$  und  $s = (s_1, \dots, s_k)$  berechnet. Hierzu werden beide Vektoren zunächst auf die Länge 1 normiert (d. h. durch ihre euklidische Länge dividiert). Anschließend ist das Skalarprodukt zwischen den normierten Vektoren  $\tilde{u}$  und  $\tilde{s}$

$$d(u, s) = \sum_i \tilde{u}_i * \tilde{s}_i$$

der Winkel zwischen den Vektoren  $u$  und  $s$ . Die möglicherweise unterschiedlichen Längen der Vektoren spielen wegen der Normierung keine Rolle.

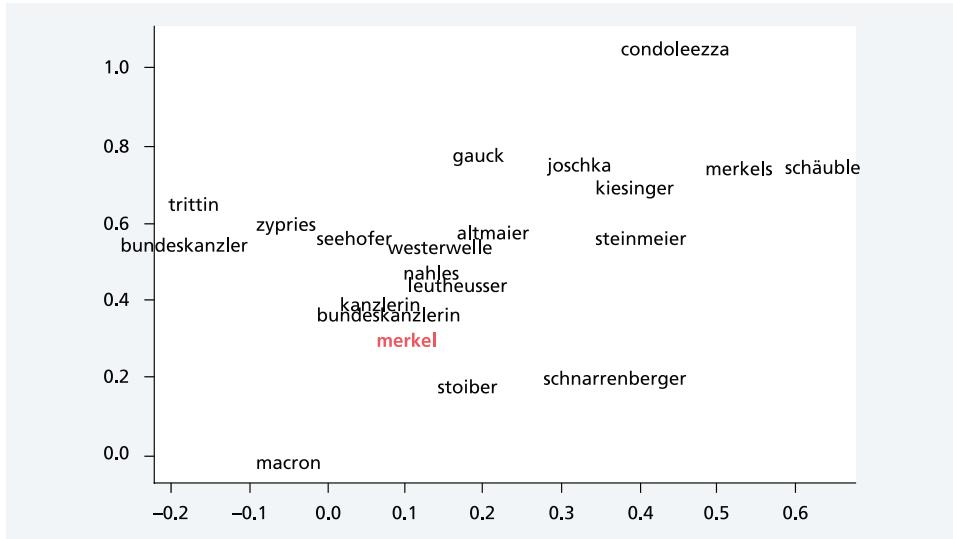
Abb. 6.9 zeigt die nächsten Nachbarn der Wörter in der ersten Zeile in absteigender Reihenfolge. Danach hat z. B. das Embedding von „bundeskanzlerin“ den geringsten Abstand zum Embedding von „merkel“. Dies besagt, dass die Wörter „merkel“ und „bundeskanzlerin“ in sehr ähnlichen Umgebungen vorkommen. Daher hat auch das Embedding des ehemaligen Vizekanzlers Steinmeier einen geringen Abstand zum Embedding von „merkel“, obwohl Steinmeier nie Bundeskanzler war. Die Ähnlichkeit der Wörter stimmt also nicht immer mit den üblichen Begriffsbildungen überein.

In Abb. 6.10 sind die Embeddings der 20 nächsten Nachbarn von „merkel“ im zweidimensionalen Raum symbolisiert. Hierbei sind z. B. „macron“ und „condoleezza“ sehr weit voneinander entfernt, weil die betreffenden Personen in meist anderen Dokumenten (zu unterschiedlichen Zeiten) zusammen mit „merkel“ erwähnt wurden. Hierbei muss man berücksichtigen, dass die gegenseitigen Abstandsbeziehungen im 100-dimensionalen Raum wesentlich differenzierter sind.

Abb. 6.10 wurde mit t-SNE (van der Maaten und Hinton 2008) erstellt, einem Verfahren zur Visualisierung von hochdimensionalen Daten. Es bildet jeden hochdimensionalen Punkt in einen Punkt im zweidimensionalen Raum ab, sodass im hochdimensionalen Raum benachbarte Punkte auch im niedrig-dimensionalen Raum nahe beieinanderliegen. Entsprechend haben im hochdimensionalen Raum weiter entfernte Punkte auch im niedrigdimensionalen Raum einen hohen Abstand.

cdu	merkel	trump	doof	iran	brexit
spd	bundeskanzlerin	trumps	doofe	irans	brexits
fdp	steinmeier	obama	doofy	irak	grexit
pds	westerwelle	barack	dumm	iranis	greixts
landesliste	merkels	obamas	doofs	teheran	frexit
bundestagswahl	seehofer	reagan	blöd	afghanistan	euroskeptischen
bundestag	schröder	mccain	lümmel	turkmenistan	europawahlkampf
afd	stoiber	bush	bumst	pakistan	ukip
csu	leutheusser	clinton	dofer	iranshar	eurokrise
landtagswahl	kanzleramtschef	clintons	blubberbumm	iranische	euroskepsis

**Abb. 6.9** Für den Embeddingvektor der Wörter in der obersten Zeile wurden die am nächsten gelegenen Embeddingvektoren berechnet und die zugehörigen Wörter ausgedruckt. Die Embeddingvektoren der in den unteren Zeilen aufgeführte Wörter sind weiter entfernt, diese Wörter also unähnlicher

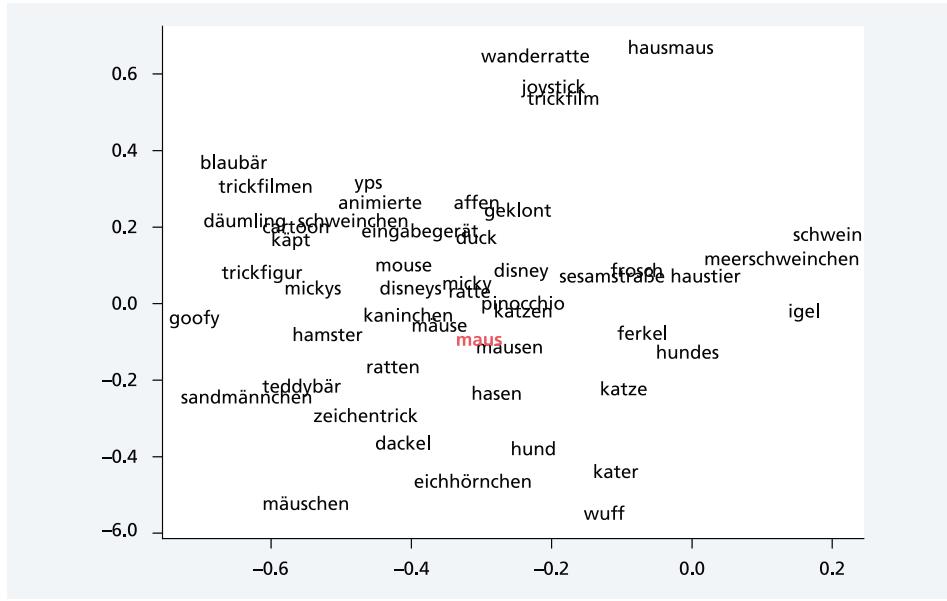


**Abb. 6.10** Die Embeddings der zwanzig nächsten Nachbarn von „merkel“ projiziert auf zwei Dimensionen. Das Skipgram-Modell wurde für Embeddings der Länge 100 für die deutsche Wikipedia berechnet

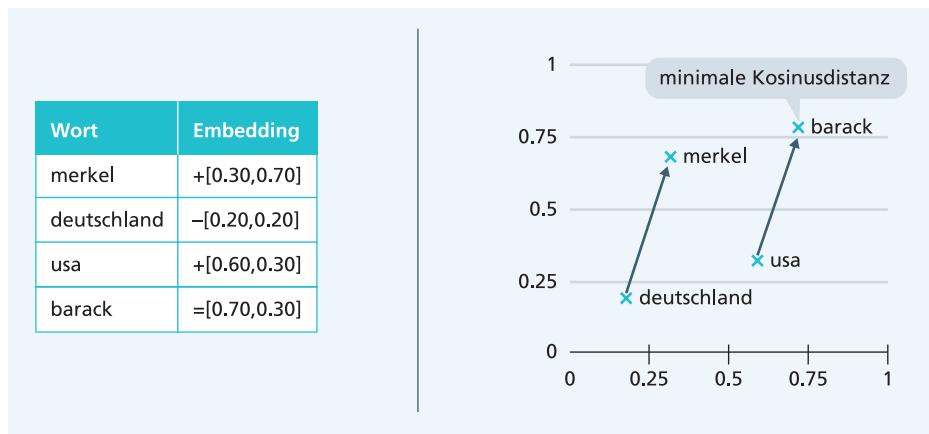
In Abb. 6.11 sind die Embeddings der 50 nächsten Nachbarn von „maus“ dargestellt. In der linken oberen Ecke finden sich meist Begriffe aus dem Zeichentrickbereich, rechts unten die Bezeichnungen von kleinen Tieren und in der Mitte auch Begriffe wie „eingabegerät“ und „joystick“. Aus den gezeigten Beispielen wird deutlich, dass man mithilfe der Embeddings die Bedeutung von Wörtern im Text lernen kann. Welche Begriffe als Nachbarn auftauchen, hängt natürlich von der verarbeiteten Sammlung von Dokumenten ab. Eine Kollektion von medizinischen Fachartikeln wird völlig andere Begriffsfelder bilden als sie sich bei der Wikipedia ergeben.

## 6.2.2 Differenzen zwischen Embeddings lassen sich als Relationen interpretieren

Die Embeddings haben die überraschende Eigenschaft, dass man Distanzvektoren zwischen Embeddings inhaltlich interpretieren kann. In Abb. 6.12 ist eine solche Beziehung in zwei Dimensionen dargestellt. Subtrahiert man das Embedding von „deutschland“ von dem Embedding von „merkel“, so erhält man einen Differenzvektor (in der Abbindung durch einen Pfeil angedeutet), der die Beziehung zwischen „merkel“ und „deutschland“ erfasst. Addiert man diesen Differenzvektor auf „usa“, so ergibt sich ein neuer Embeddingvektor. Zu diesem Embeddingvektor kann man das Wort mit dem nächsten Embedding suchen und erhält „barack“. Damit kann man den Differenzvektor als die Beziehung zwischen einem Land und dessen Regierungschef interpretieren. Eine solche Beziehung zwischen Begriffen wird auch Analogie genannt, bei der die Relation zwischen zwei Kon-



**Abb. 6.11** Zweidimensionale Projektion der Embeddingvektoren der 50 nächsten Nachbarn des Wortes „maus“. Das Skipgram-Modell wurde für Embeddings der Länge 100 für die deutsche Wikipedia berechnet



**Abb. 6.12** Subtrahiert man von dem Embedding von „merkel“ das Embedding von „deutschland“, so ergibt sich ein Differenzvektor, angedeutet durch einen Pfeil. Addiert man diesen Differenzvektor zu dem Embedding von „usa“, so ist das Wort mit dem nächstgelegenen Embedding „barack“

berlin - deutschland + russland	dollar - usa + russland	merkel - deutschland + usa	rot - rose + gras	klein - frau + mann
moskau	rubel	barack	grün	groß
petersburg	juganskneftegas	obama	gelb	bajohren
kolokolnikow	rubeln	reagan	blau	gross
tschurikow	surgutneftegas	obamas	grasbulten	kleinendorf
zekaterinowski	nefegas	clinton	grasbüscheln	schmal
sokolnikow	milliarden	bush	mittelrot	gansen
leningrad	milliarde	condoleezza	bodenfarben	krotzen
chwostow	millionow	trumps	grasbüschel	gonschorowen
zekaterinodar	beresowsk	merkels	auferichtetem	trampken

**Abb. 6.13** Analogien lassen sich mit Embeddings berechnen. In der Titelzeile stehen die Wörter, deren Embeddings subtrahiert und addiert wurden. In den Spalten stehen in absteigender Reihenfolge die Wörter, deren Embeddings dem Ergebnis am nächsten kommen

zepten (hier „usa“ und „barack“) durch die Relation zwischen zwei anderen Konzepten („deutschland“ und „merkel“) erklärt wird.

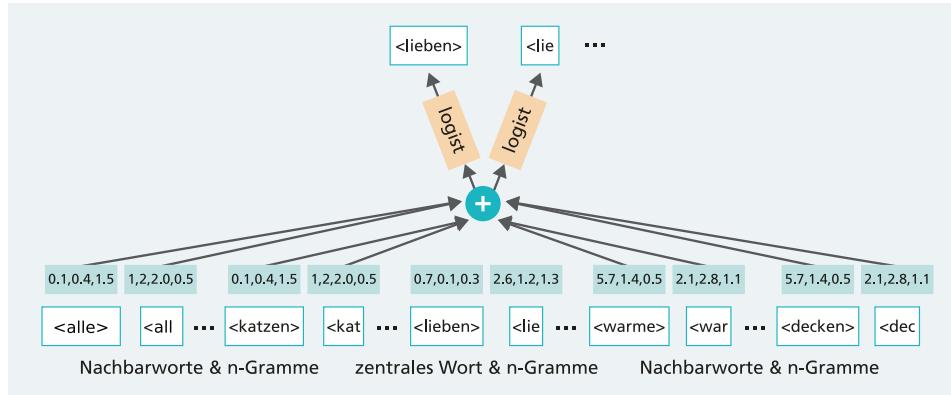
In Abb. 6.13 finden sich weitere Analogien, welche in den Embeddings gelten. Die Überschriften zeigen, welche Embeddings abgezogen und addiert wurden. Darunter sind die nächsten Nachbarn zum Ergebnis der resultierenden Embeddings in absteigender Reihenfolge aufgeführt. Die Analogien gelten häufig für die Beziehung Land – Hauptstadt, Land – Währung, Land – Regierungschef usw. In der letzten Spalte sieht man ein Beispiel, in dem sich ein mögliches Vorurteil im Sprachgebrauch zeigt. Während eine „frau“ das Größenattribut „klein“ erhält, so wird dem „mann“ das Größenattribut „groß“ zugeordnet.

Solche Analogien lassen sich nur für die Relationen extrahieren, die im Text häufig vorkommen. Zu Testzwecken wurde eine Menge von 18.000 solcher Relationen aufgestellt und geprüft, wie oft sie reproduziert wurden. Das Word2vec-Verfahren war in der Lage, 69 % dieser Analogien nachzuvollziehen (Levy et al. 2015).

Embeddings für Wörter lassen sich nutzen, um die Bedeutung von Wörtern zu repräsentieren. Daher werden auch in Websuchmaschinen Embeddings verwendet, um nach bedeutungsähnlichen Wörtern zu suchen und nicht nur nach übereinstimmenden Zeichenfolgen (Diaz et al. 2016). Dies führt zu einer wesentlichen Verbesserung der Suchergebnisse.

### 6.2.3 FastText nutzt n-Gramme von Buchstaben

Ein Problem des Word2vec-Verfahrens ist, dass jedes Wort ein eigenes Embedding erhält und sehr seltene Wörter nicht berücksichtigt werden. Aus diesem Grunde wurde eine Variante vorgeschlagen, welche nicht nur Wörter, sondern auch Wortteile in Betracht zieht (Bojanowski et al. 2017). Damit kann das Verfahren auch Silben und Endungen als Eingaben verwenden und seltene Wortteile aus seinen Bestandteilen zusammensetzen (Abb. 6.14).



**Abb. 6.14** Training des FastText-Modells mit dem Text „alle Katzen lieben warme Decken ...“. Jedes Wort wird zusätzlich durch seine Buchstabenfolgen (n-Gramme) dargestellt. Für ein zentrales Wort („lieben“) wird eine Nachbarschaft von Wörtern berücksichtigt (hier zwei Nachbarn). Es wird die Summe aller Embeddings der Buchstabenfolgen der Nachbarwörter gebildet. Hiermit wird das zentrale Wort und seine n-Gramme prognostiziert

Wie in Abb. 6.15 gezeigt, wird ein Wort durch das gesamte Wort und Buchstabenfolgen repräsentiert. Derartige Folgen von  $n$  Buchstaben nennt man auch  $n$ -Gramme. Es werden meist alle überlappenden 3-Gramme bis 6-Gramme genutzt. Abb. 6.15 zeigt die Darstellung des Wortes „mutprobe“ durch das Wort selbst und seine  $n$ -Gramme.

Für jeden Teil dieser Darstellung wird ein eigenes Embedding verwendet. Wie bei Word2vec wird ein Modell genutzt, welches die Relation eines zentralen Wortes mit der Nachbarschaft beschreiben will. Eingabe in das Modell ist die Summe der Embeddings der Nachbarwörter und deren Buchstabenfolgen. Mit dieser Eingabe soll das zentrale Wort und seine Buchstabenfolgen mit einem logistischen Modell vorhergesagt werden (Abb. 6.14).

Beim Training werden sowohl die Parameter des logistischen Modells als auch die Embeddings so modifiziert, dass die Prognose möglichst gut wird. In Abb. 6.16 werden Resultate von FastText bei der Anwendung auf deutschen Trainingsdaten gezeigt. Für zusammengesetzte Wörter werden die wichtigsten Buchstabenfolgen gezeigt, welche das größte Gewicht bei der Repräsentation des zusammengesetzten Wortes erhalten.

**Abb. 6.15** Zerlegung eines Wortes in Buchstabenfolgen (n-Gramme). < und > deuten den Beginn und das Ende des Wortes an

Zerlegung des Wortes "mutprobe":  
komplettes Wort: <mutprobe>  
3-Gramme: <mu, mut, utp, tpr, pro, rob, obe, be>  
4-Gramme: <mut, mutp, utpr, tpro, prob, robe, obe>  
5-Gramme: <utp, mutpr, utpro, tprob, probe, robe>

Wort	Wichtige n-Gramme		
autofahrer	fahr	fahrer	auto
freundeskreis	kreis	kreis>	<freun
grundwort	wort	wort>	grund
sprachschule	schul	hschul	sprach
tageslicht	licht	gesl	tages

**Abb. 6.16** Die wichtigsten Zeichenfolgen für einige Wörter. Für jedes Wort werden die n-Gramme gezeigt, die zu den stärksten Änderungen in der FastText-Repräsentation führen, wenn sie weggelassen werden

### 6.2.4 StarSpace erzeugen Embeddings für andere Objekte

Bisher haben wir Verfahren kennengelernt, die Wörter durch Embeddings beschreiben, welche die Nachbarschaft jeweils eines Paares von Wörtern auswerten. Dieses Vorgehen kann man von Wortpaaren auf andere Arten von Objektpaaren übertragen, z. B. auf die Nachbarschaft zwischen Wörtern und Dokumentklassen, zwischen einer Web-Anfrage und den Antwortdokumenten, zwischen einem Musikstück und den Zuhörern, zwischen den Personen eines Beziehungsgraphen (z. B. Netz der Facebook-Freunde) usw. All diese Beziehungen lassen sich mithilfe des StarSpace-Verfahrens (Wu et al. 2018)

Anfrage	Am besten passender Wikipedia Artikel
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...

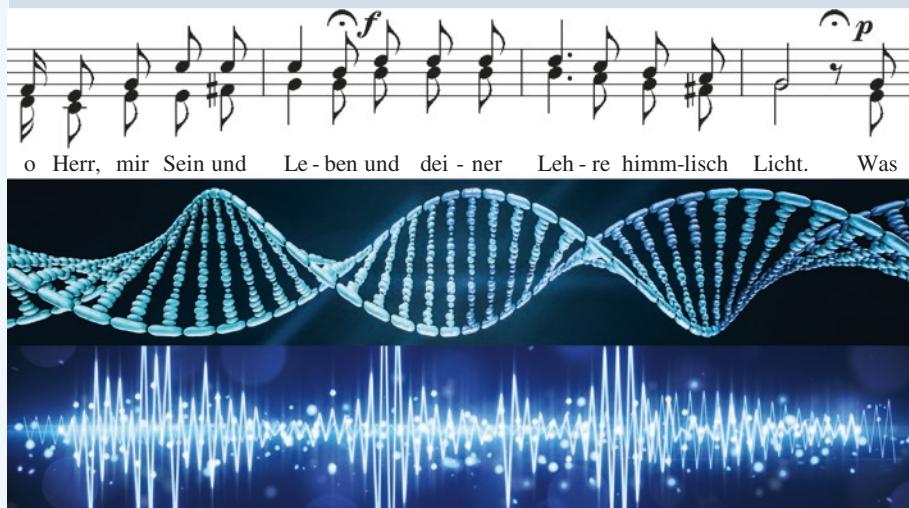
**Abb. 6.17** StarSpace berechnet mithilfe von Embeddings die am besten passenden Wikipedia-Artikel zu einer Anfrage

berücksichtigen. Es arbeitet nach dem gleichen Schema wie der FastText-Algorithmus. Die gewonnenen Embeddings lassen sich dann je nach Anwendung für die Klassifikation von Texten in sehr viele Objektklassen, für die Suche geeigneter Dokumente in einer Suchmaschine, für Empfehlung geeigneter Musikstücke, für die Embeddings von Sätzen oder Dokumenten usw. einsetzen. Abb. 6.17 zeigt die Ergebnisse einer Suche nach Wikipedia-Artikeln. Dabei wurden diejenigen Artikel ausgewählt, die bezüglich der StarSpace-Embeddings am besten zu einer Suchanfrage passten.

### 6.3 Rekurrente neuronale Netze zur Erfassung von Sequenzen

Rekurrente neuronale Netze (RNN) sind neuronale Netze, die auf die Verarbeitung von Sequenzen spezialisiert sind (Rumelhart et al. 1986). Sequenzen sind z. B. die aufeinanderfolgenden Wörter eines Satzes, die Noten eines Musikstückes, die Bausteine einer DNA-Sequenz oder aber die Töne einer Sprachaufnahme (Abb. 6.18). So wie die CNN auf die zweidimensionale Struktur von Pixelmatrizen in der Bildverarbeitung spezialisiert sind, so verarbeiten RNNs die Elemente einer Sequenz. Wir werden uns im Folgenden auf

Die Terroranschläge am 11. September 2001 waren vier koordinierte Flugzeugentführungen mit anschließenden Selbstmordattentaten auf wichtige zivile und militärische Gebäude in den Vereinigten Staaten von Amerika. Sie wurden vom islamistischen Terrornetzwerk al-Qaida geplant und von 19 seiner Mitglieder verübt, darunter 15 Bürgern Saudi-Arabiens. Die Ereignisse dieses Tages werden in den USA auch kurz als Nine-Eleven oder 9/11[1], auf Deutsch auch als 11. September bezeichnet.



**Abb. 6.18** Beispiele für Sequenzen: Text eines Dokuments, Noten und Text eines Musikstückes, DNA-Sequenz. und Schallschwingungen eines akustischen Ereignisses. Bildnachweis im Anhang

Sequenzen von Wörtern konzentrieren, die entsprechenden Modelle können aber auch auf andere Arten von Sequenzen übertragen werden.

TNN, die wir bisher kennengelernt haben, erhalten eine Eingabe fester Größe (Vektor, Matrix oder Tensor). Sequenzen hingegen haben eine variable Länge. Es muss also ein neues Schema gefunden werden, mit dem man Sequenzen mit neuronalen Netzen verarbeiten kann.

Eine wichtige Aufgabe ist die Prognose des nächsten Wortes in einem Satz. Ein solches Modell heißt auch **Sprachmodell**, weil es die grammatischen und inhaltlichen Zusammenhänge der Wörter in einem Text erkennen muss. Da in der Regel viele Wörter an der nächsten Position möglich sind, berechnet man für jedes Wort im Lexikon die Wahrscheinlichkeit, dass es das nächste Wort ist. Dazu wird die Information benötigt, welche Wörter bereits zuvor im Satz aufgetaucht sind. Sprachmodelle sind ein weiteres Beispiel für selbstüberwachtes Lernen (Abschn. 3.1.2), bei dem ein Teil der beobachteten Trainingsdaten aus einem anderen Teil prognostiziert wird. Wir werden sehen, dass sich mit Hilfe von Sprachmodellen grammatisch korrekte und inhaltlich konsistente Texte erzeugen lassen.

### 6.3.1 Rekurrente neuronale Netze als Sprachmodelle

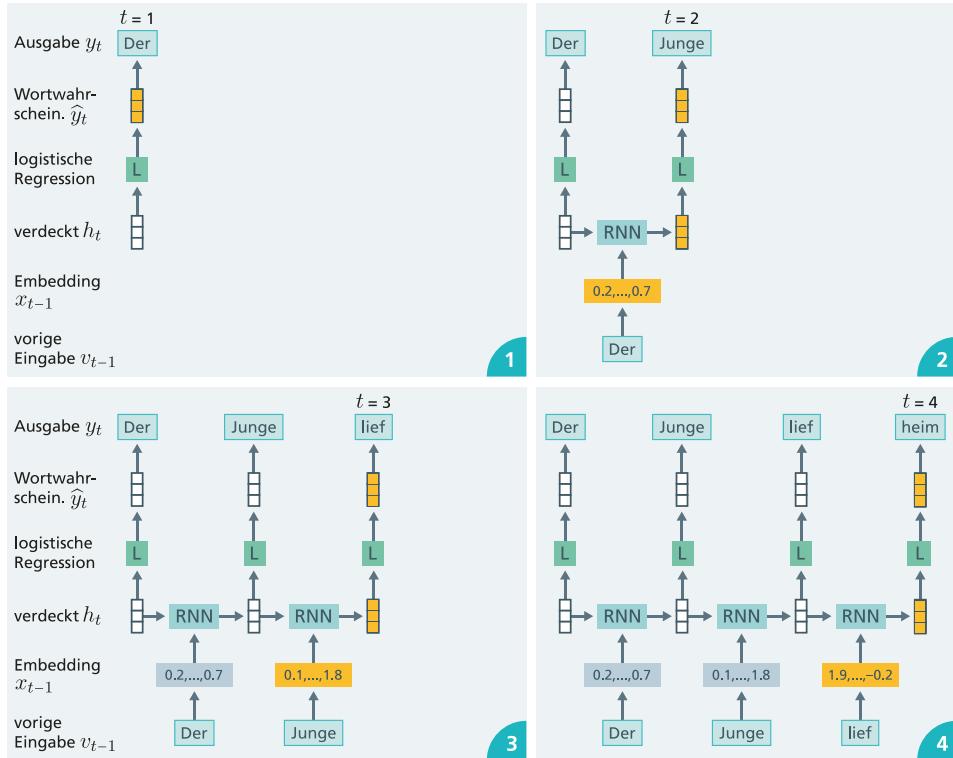
Wir haben gesehen, dass in TNN die Information aus dem Eingabevektor durch den Operator einer Schicht transformiert wird (Abschn. 4.3.1). Ergebnis ist ein verdeckter Vektor, der die Eingabe repräsentiert. In gleicher Weise kann man einen verdeckten Vektor nutzen, um den zurückliegenden Teil eines Satzes zu repräsentieren. Die Worte des Vokabulars können an nahezu allen Positionen in einem Satz auftauchen. Daher benötigt man ein Netz, welches an den unterschiedlichen Stellen der Sequenz die gleiche Verarbeitung durchführt. Da ein solches Netz immer wieder die gleichen Operatoren nutzt, hat man die Bezeichnung rekurrentes neuronales Netz gewählt.

Die Funktionsweise eines RNN ist in Abb. 6.19 für einzelne Arbeitsschritte dargestellt. Hier geht es darum, die Wahrscheinlichkeiten der Worte des Satzes „Der Junge lief heim“ nacheinander zu prognostizieren.

Zunächst werden für die Prognose des laufenden Ausgabewortes  $y_t$  das **vorherige** Wort  $v_{t-1}$  als Eingabe für das Netz kodiert. Dazu wird ein Embeddingvektor  $x_{t-1} = emb(v_{t-1})$  der Länge  $n_x$  (z. B.  $n_x = 100$ ) verwendet, der ja die Bedeutung eines Wortes  $v_{t-1}$  an der Position  $t - 1$  darstellen kann (Abschn. 6.2.1). Für das nicht vorhandene Wort  $v_0$  wird dabei das Embedding  $x_0 = (0, \dots, 0)$  verwendet.

Dann wird ein verdeckter Vektor  $h_{t-1}$  mit einer zuvor gewählten Länge (z. B.  $n_h = 200$ ) definiert, um den Inhalt der bisherigen Worte des Satzes vor der Position  $t$  zu speichern. Zu Beginn ( $t = 1$ ) wird  $h_t$  meist auf 0 gesetzt. Das RNN besteht im einfachsten Fall aus einer linearen Transformation und einer nichtlinearen Aktivierung, welche den aktuellen verdeckten Vektor  $h_t$  aus dem vorherigen verdeckten Vektor und dem Embedding  $x_{t-1}$  des vorherigen Wortes berechnet (Elman 1990):

$$h_t = \tanh(U * h_{t-1} + W * x_{t-1} + b)$$



**Abb. 6.19** Rekurrentes neuronales Netz zur Prognose der Wahrscheinlichkeit des nächsten Wortes. Die in jedem Schritt berechneten Vektoren sind orange eingefärbt. Zu Beginn wird aus den Startvektor  $h_1$  die Wahrscheinlichkeit des ersten Wortes (beobachtet wurde „Der“) mit dem logistischen Regressionsmodell  $L$  prognostiziert ( $t = 1$  oben links). Anschließend prognostiziert das RNN den nächsten verdeckten Vektor  $h_2$  aus dem Embedding  $x_1$  des vorherigen Wortes „Der“ und dem vorherigen verdeckten Vektor  $h_1$  ( $t = 2$  oben rechts). Das logistische Regressionsmodell  $L$  berechnet dann die Wahrscheinlichkeit des nächsten Wortes (beobachtet wurde „Junge“) aus dem verdeckten Vektor  $h_2$ . Anschließend werden analog die Wahrscheinlichkeiten der beobachteten Worte „lief“ ( $t = 3$ ) und „heim“ ( $t = 4$ ) prognostiziert

Die Matrizen  $U$  und  $W$  sowie der Vektor  $b$  sind unbekannte Parameter. Die Wahrscheinlichkeit des nächsten Wortes  $v_t$  wird mit einem logistischen Regressionsmodell prognostiziert:

$$\hat{p}(v_t | v_{t-1}, \dots, v_1) = \text{softmax}(A * h_t + c)$$

Die bedingte Wahrscheinlichkeit wird durch das RNN näherungsweise berechnet; daher wurde sie mit dem Schätzsymbol  $\hat{p}$  markiert. Die Matrix  $A$  und der Vektor  $c$  sind ebenfalls unbekannte Parameter. In Abb. 6.20 sind die Berechnungsschritte aufgeführt, die notwendig sind, um die bedingten Wahrscheinlichkeiten der Worte eines Satzes zu bestimmen. Die bedingte Wahrscheinlichkeit ist ein Zahlenvektor, der für jedes Wort des Vokabulars eine Wahrscheinlichkeit enthält. Hat das Vokabular einen Umfang  $n_v = 50.000$ , so umfasst der Wahrscheinlichkeitsvektor 50.000 Wahrscheinlichkeiten, die sich zu 1,0 aufaddieren.

Für  $t = 1, \dots, 4$

1. Bestimme das Embedding  
 $x_{t-1} = emb(v_{t-1})$  für das vorige Wort  $v_{t-1}$
2. Berechne den nächsten verdeckten Vektor  
 $h_t = \tanh(U * h_{t-1} + W * x_{t-1} + b)$
3. Berechne die Wahrscheinlichkeit des nächsten Wortes  $v_t$   
 $\hat{p}(v_t | v_{t-1}, \dots, v_1) = \text{softmax}(A * h_t + c)$

**Abb. 6.20** Berechnung der bedingten Wahrscheinlichkeiten der Worte eines Satzes

Zu Beginn des Trainings ist die prognostizierte Wahrscheinlichkeit des nächsten Wortes sehr niedrig. Würde nur diese Wahrscheinlichkeit für die Prognose des übernächsten Wortes genutzt, so wäre der Fehler in jedem Fall sehr hoch. Daher wird bei der Prognose des übernächsten Wortes nicht die prognostizierte Wahrscheinlichkeit des vorigen Wortes als Eingabe verwendet, sondern das Embedding des Wortes selbst. Durch die Nutzung des verzögerten Satzes als Eingabe wird die Konvergenz stark unterstützt. Wichtig ist, dass der zuvor berechnete verdeckte Vektor  $h_{t-1}$  zusammen mit dem Embedding  $emb(v_{t-1})$  in die Berechnung des nächsten verdeckten Vektors  $h_t$  eingeht. Damit kann der verdeckte Vektor potentiell den Inhalt aller vorherigen Worte speichern.

### 6.3.2 Training der rekurrenten neuronalen Netze

Das RNN berechnet näherungsweise die bedingte Wahrscheinlichkeit des nächsten Wortes, wenn die vorherigen Wörter bekannt sind. Damit kann man die Wahrscheinlichkeit des ganzen Satzes berechnen. In unserem Beispiel ergibt sich

$$p(\text{Der Junge lief heim}) = p(\text{heim} | \text{Der Junge lief}) * p(\text{lief} | \text{Der Junge}) * p(\text{Junge} | \text{Der}) * p(\text{Der})$$

Dies ist ein Zusammenhang, der sich direkt aus der Definition der bedingten Wahrscheinlichkeit ergibt. Dies bedeutet, dass man die Wahrscheinlichkeit eines kompletten Dokumentes mit  $T$  Wörtern durch die bedingten Wahrscheinlichkeiten der jeweils nächsten Wörter berechnen kann. Wird ein Sprachmodell anhand einer großen Trainingsmenge von Texten trainiert, so lässt sich anschließen sehr genau die Plausibilität eines neuen Textes bewerten. Damit kann man potentiell alle syntaktischen, aber auch alle inhaltlichen Bezüge zwischen Wörtern reproduzieren, die in der Trainingsmenge auftreten. Wir werden im weiteren Verlauf sehen, dass es mittlerweile Sprachmodelle gibt, welche die tatsächliche bedingte Wahrscheinlichkeit sehr gut approximieren und damit inhaltlich erstaunlich konsistente Texte generieren können.

Die Prognose der bedingten Wahrscheinlichkeiten hängt von den unbekannten Parametern ab, die am Anfang zufällig auf kleine Werte gesetzt werden. Unbekannte Parameter sind in unserem Fall die Matrizen  $U$ ,  $W$  und der Vektor  $b$  des RNN, die Matrix  $A$  und der Vektor  $c$  des logistischen Modells und die Embeddings  $emb(v_i)$  der Worte, d. h.  $w = (U, W, A, b, c, emb)$  ist der Vektor aller Parameter.  $w$  soll nun durch Optimierung so modifiziert werden, dass die Wahrscheinlichkeit der Trainingsdaten möglichst hoch wird.

Besteht die Trainingsmenge aus  $N$  Sätzen  $s_1, \dots, s_N$ , so werden Parameter so verändert, dass das Produkt der Wahrscheinlichkeiten aller Sätze möglichst hoch wird (Abschn. 3.5.1):  $\max_w \prod_{i=1}^N \hat{p}(s_i; w)$ . Um zu kleinen Zahlen zu vermeiden, wird dieser Ausdruck mit  $-\log$  transformiert und man erhält

$$L(w) = \sum_{i=1}^N -\log \hat{p}(s_i; w) = -\sum_{i=1}^N \sum_{t=1}^{n_i} \log \hat{p}(v_{i,t} | v_{i,t-1}, \dots, v_{i,1}; w)$$

wenn der  $i$ -te Satz aus den Wörtern  $v_{i,1}, \dots, v_{i,n_i}$  besteht. Dies definiert analog zu (Abschn. 3.5.2) die Verlustfunktion  $L(w)$  des RNNs. Zur Optimierung des RNN wird für ein Minibatch von Sätzen der Trainingsmenge der Gradient der Verlustfunktion  $\frac{\partial L(w)}{\partial w}$  benötigt. Dieser wird automatisch von modernen Toolkits geliefert (Abschn. 4.5.2), und man kann mit dem stochastischen Gradientenabstieg (SGD) nahezu optimale Parameter erhalten (Abschn. 3.6.6). Wie beim Word2vec-Modell wird Noise Contrastive Estimation (Abschn. 6.1.3) zur näherungsweisen Abschätzung der Wortwahrscheinlichkeiten verwendet.

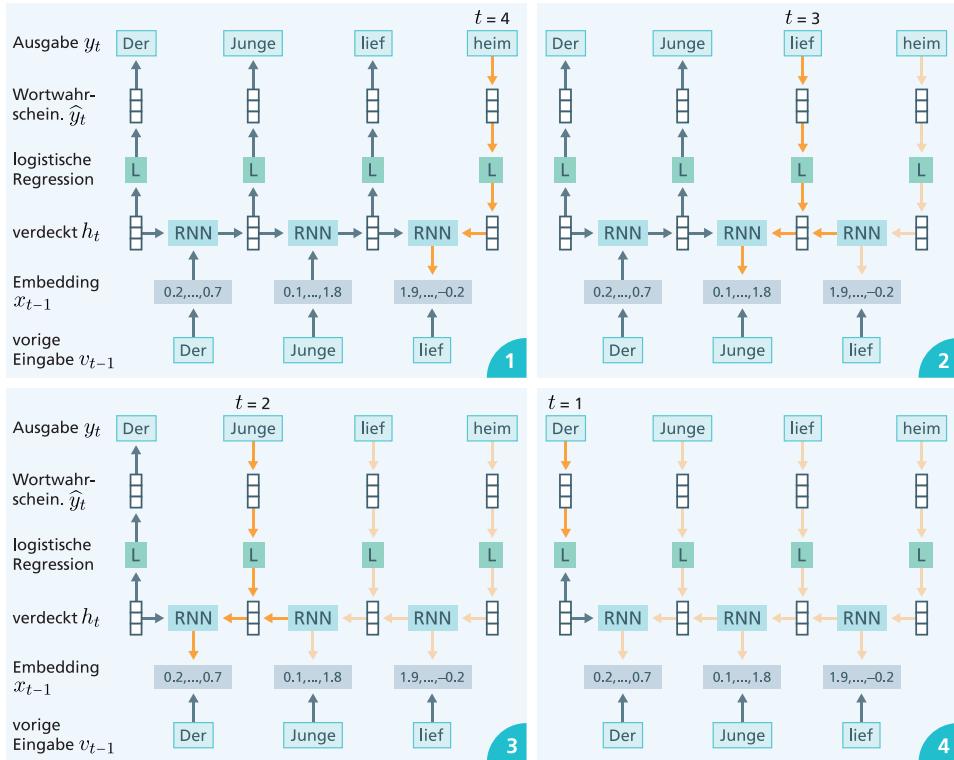
Während der Optimierung wird zunächst für einen Satz die Wahrscheinlichkeit der nächsten Worte aus den bisherigen Wörtern berechnet. Im Gegensatz zu den bisher betrachteten TNN haben wir nicht nur einen einzelnen Wert der Verlustfunktion zu berücksichtigen, sondern für jedes prognostizierte Wort einen anderen Wert der Verlustfunktion. Alle diese Verlustwerte werden gleichzeitig bei der Optimierung ausgewertet und beeinflussen gemeinsam den Gradienten. Abb. 6.21 zeigt den Informationsfluss, der während der Gradientenberechnung stattfindet. Hierbei wird die Berechnungsrichtung der Prognose (graue Pfeile) umgekehrt. Von jedem Ausgabewort wird zunächst das Fehlersignal (Verlustwert) zum logistischen Modell zurückpropagiert (orange Pfeile). Anschließend wird es zum RNN übertragen. Von da aus wird es zu den vorherliegenden RNNs zurückpropagiert. Zum Schluss wird das Fehlersignal in die Embeddings übertragen. Es werden also gleichzeitig die Parameter des RNN, des logistischen Modells und der Embeddings modifiziert.

### 6.3.3 Die Eigenschaften der Gradienten beim RNN

Durch die wiederholte Transformation des verdeckten Vektors  $h_t$  durch das RNN gibt es einen bemerkenswerten Effekt. Die Herleitung in Abb. 6.22 für ein sehr einfaches RNN mit einem eindimensionalen verdeckten Vektor  $h_t$  zeigt, dass die Ableitungen weit zurückliegender Worte im Satz immer den Faktor  $u^k$  enthalten, wobei  $u$  ein Parameter ist.

Ist  $u = 2$ , so ist für  $k = 20$  der Wert von  $u^k$  größer als eine Million. Man spricht dann von einem explodierenden Gradienten. Verändert man die Parameter mit diesem Gradienten, so kommen die Parameterwerte in extreme Bereiche, wo das Modellverhalten sehr unrealistisch wird und alle zukünftigen Gradienten nahezu Null sein können.

Eine einfache und effektive Lösung für dieses Problem besteht darin, dass man die euklidische Länge des Gradientenvektors  $l_{grad}(w)$  berechnet. Ist der Vektor länger als ein Wert  $\gamma$ , z. B.  $\gamma = 5,0$ , so multipliziert man den Gradienten mit dem Faktor  $\gamma/l_{grad}(w)$ , und



**Abb. 6.21** Informationsfluss während der Zurückpropagierung des Fehlersignals, symbolisiert durch orangene Pfeile. Dabei werden die während der Vorwärtspropagierung gespeicherten Werte der verdeckten Vektoren genutzt. Zunächst wird das Fehlersignal der Prognose von „heim“ verwendet und Ableitungen für das logistische Modell, das RNN und die Embeddings von „lief“ berechnet (siehe ①). Dann wird das Fehlersignal für die Prognose von „lief“ genutzt und mit dem Fehlersignal von „heim“ verknüpft. Daraus werden wieder Ableitungen für das logistische Modell, das RNN und die Embeddings von „Junge“ berechnet siehe (②). Analog wird in der unteren Reihe für die verbleibenden Eingabeworte „Junge“ ③ und „Der“ ④ vorgegangen

das Gradientenupdate verändert den Parametervektor  $w$  in die richtige Richtung mit einer moderaten Schrittweite. Diese Strategie wird als Gradienten-Clipping bezeichnet.

Ist hingegen  $u = 1/2$ , so ist für  $k = 20$  der Wert von  $u^k$  kleiner als ein Millionstel. Diesen Fall nennt man verschwindender Gradient. Das bedeutet, dass längerfristige Zusammenhänge nicht durch Gradienten vermittelt werden können; diese kleinen Werte haben kaum Auswirkungen und werden durch die Zufallsschwankungen des stochastischen Gradientenabstieg nahezu vollständig verdeckt.

Gerade im Deutschen sind in einem Satz Zusammenhänge zwischen weit auseinanderliegenden Wörtern wichtig. Beispiel: „Peter sah seine Frau nach einer langen Trennung erst im Sommer 2013 wieder.“ Offenbar ist „wieder“ ein Bestandteil des Verbs „sah“ und der Satz ist nur verständlich, wenn die Beziehung zwischen diesen beiden Wörtern erfasst wird. Aus diesem Grunde waren einfache RNN über viele Jahre nicht sehr erfolgreich.

RNN mit eindimensionalem verdeckten Vektor  $h_t$

$$h_t = \tanh(u * h_{t-1})$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \tanh(x)}{\partial x} (u * h_{t-1}) * u \text{ nach der Kettenregel}$$

Nach zwei Schritten ist

$$h_t = \tanh(u * \tanh(u * h_{t-2}))$$

$$\frac{\partial h_t}{\partial h_{t-2}} = \frac{\partial \tanh(x)}{\partial x} (uh_{t-1}) u * \frac{\partial \tanh(x)}{\partial x} (u * h_{t-2}) u$$

Das heißt, bei jedem Schritt bekommen wir einen zusätzlichen Faktor  $u$ , d.h. nach  $k$  Schritten gilt:

$$\frac{\partial h_t}{\partial h_{t-k}} = u^k * \text{Restterme}$$

**Abb. 6.22** Rekurrente neuronale Netze haben einen exponentiellen Faktor  $u^k$  in den Ableitungen. Dadurch können die Gradienten sehr groß werden: explodierende Gradienten

## 6.4 Das Long-Short Term Memory (LSTM) ist ein Langzeitspeicher

### 6.4.1 Gatter steuern die Speichervorgänge

Eine Lösung wurde von Schmidhuber et al. (1997) vorgeschlagen. Sie entwarfen ein Netz, das Long Short Term Memory (LSTM), welches als Langzeitspeicher für Abhängigkeiten zwischen weit entfernten Eingaben einen zusätzlichen Speichervektor  $c_t$  verwendet (Abb. 6.23, 6.24 und 6.25).

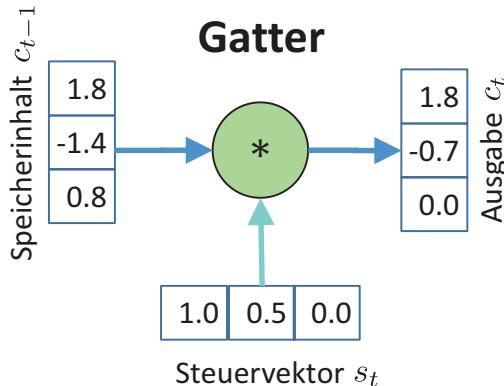
Dieser Speichervektor soll Informationen über einen längeren Zeitraum speichern und bei Bedarf auslesen und löschen können. Allerdings kann man nicht – wie bei einem digitalen Computer – einen Speicherbereich auswählen, der von einem Prozessor mit Daten gefüllt und ausgelesen wird. Vielmehr benötigt man einen Mechanismus, der löschen, speichern und auslesen und zusätzlich noch über Gradienten trainiert werden kann.

Dieser Mechanismus ist ein Gatter, welches in Abb. 6.23 dargestellt ist. Als Eingabe nimmt es einen Vektor mit dem Speicherinhalt  $c_{t-1}$ , welcher ein Vektor vorgegebener Länge von beliebigen Zahlen ist. Weiter gibt es einen Steuervektor  $s_t$ , der gleichen Länge, der nur Zahlen zwischen 0.0 und 1.0 enthalten darf. Diese beiden Vektoren werden komponentenweise multipliziert und ergeben das Resultat  $c_t$ . Hat eine Komponente des Steuervektors den Wert ...

- 1.0, so wird die entsprechende Komponente des Speichervektors nicht verändert.
- 0.0, so wird die entsprechende Komponente des Speichervektors auf 0 gesetzt, d. h. gelöscht.

- einen Wert zwischen 0.0 und 1.0, so wird die entsprechende Komponente des Speichervektors entsprechend reduziert, d. h. teilweise gelöscht.

Schmidhuber und Hochreiter (1997) schlugen die Verwendung dreier Gatter vor, die unabhängig voneinander Komponenten des Speichervektors löschen (Löschgatter), Werte zum Speichervektor addieren (Speichergatter), und Werte aus dem Speichergatter auslesen können (Ausgabegatter). Damit erreichten sie eine hohe Flexibilität bei der Änderung und Nutzung der Inhalte des Speichervektors.



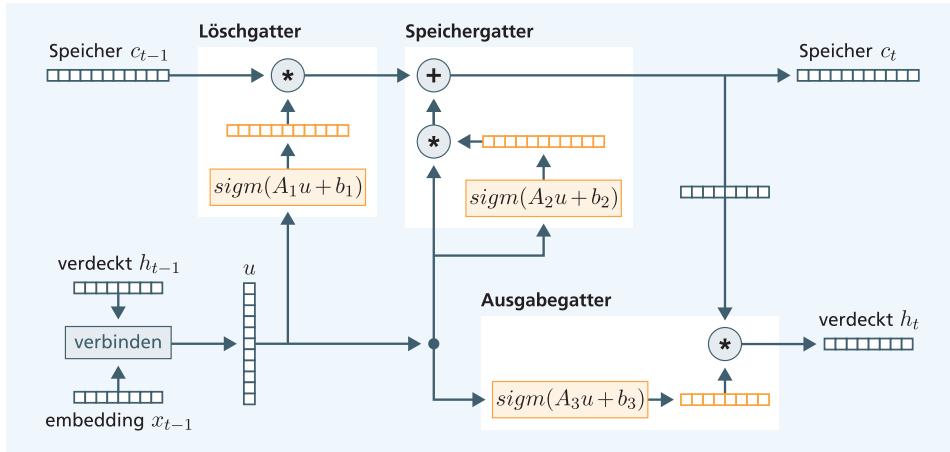
**Abb. 6.23** Ein Gatter modifiziert jede Komponente des Speicherinhaltes durch Multiplikation mit einer Zahl zwischen 0.0 und 1.0 aus dem Steuervektor und erzeugt so eine Ausgabe



**Jürgen Schmidhuber**

studierte Informatik und Mathematik an der TU München. Zusammen mit Sepp Hochreiter analysierte er das Problem der verschwindenden Gradienten. Als Lösung entwickelten beide das Long Short-Term Memory (LSTM). 1995 wurde er wissenschaftlicher Direktor von IDSIA in Lugano. Durch die Verwendung von GPUs konnte er den Trainingszeitbedarf stark reduzieren und damit 2011 die Genauigkeit von CNNs stark erhöhen. Er und seine Arbeitsgruppe lieferten wichtige Beiträge zu vielen Gebieten: Robotersteuerung in partiell sichtbaren Umgebungen, Musikkomposition, Aspekte der Sprachverarbeitung und Erkennen von Handschriften. 2016 erhielt er den IEEE CIS Neural Networks Pioneer Award. Wegen der durch KI erwartbaren Probleme im Arbeitsmarkt empfiehlt Schmidhuber ein bedingungsloses Grundeinkommen.

**Abb. 6.24** Jürgen Schmidhuber, einer der Erfinder des rekurrenten neuronalen Netzes LSTM, \* 17.01.1963 in München. Bildnachweis im Anhang



**Abb. 6.25** Vereinfachtes Schema des LSTMs. Ein Speichervektor  $c_t$  soll Inhalte langfristig speichern. Der vorige verdeckte Vektor und das Embedding des vorherigen Wortes werden aneinandergehängt und bilden die Eingabe  $u$ . Die Steuervektoren der Gatter werden mit unterschiedlichen Sigmoid-Schichten (orange) aus  $u$  erzeugt. Mit einem Löschgatter werden einzelne Komponenten des Speichers gelöscht. Das Speichergatter addiert Inhalte aus  $u$  zu einzelnen Komponenten des Speichers  $c_t$ . Mit dem Ausgabegatter werden einzelne Komponenten des Speichers  $c_t$  ausgelesen und bilden den neuen verdeckten Vektor  $h_t$

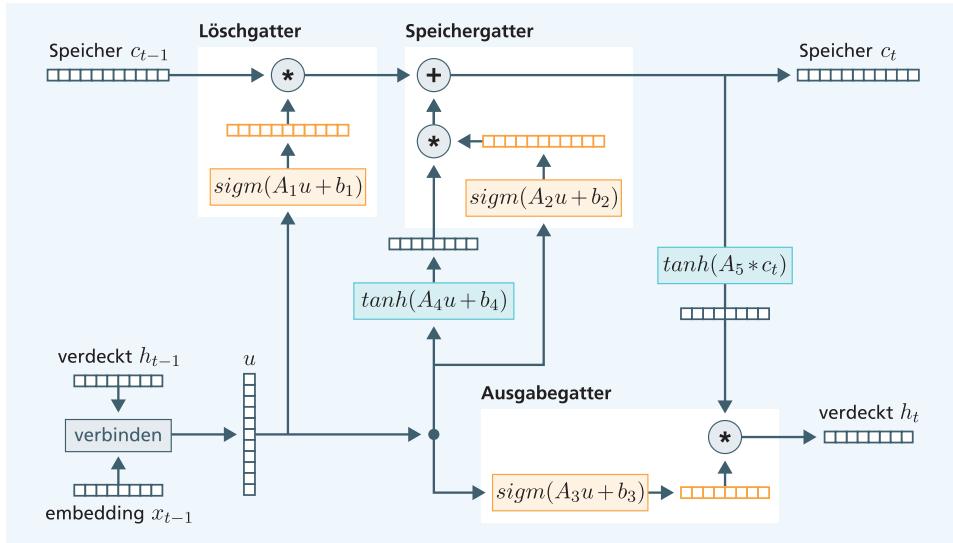
Man benötigt nun noch einen Weg, die Steuervektoren für die einzelnen Gatter zu erzeugen. Hier verwendeten Schmidhuber und Hochreiter (1997) ein Subnetz mit einer Sigmoid-Aktivierungsfunktion (Abb. 4.4), welche immer Ausgaben zwischen 0.0 und 1.0 generiert.

$$s_i = \text{sigmoid}(Au + b)$$

Dabei ist  $u_{t-1} = [h_{t-1}, x_{t-1}]$  der verdeckte Vektor  $h_{t-1}$ , an den die aktuelle Eingabe  $x_{t-1}$ , das Embedding des laufenden Wortes, angehängt wird. Der Index  $t - 1$  wurde der Einfachheit halber weggelassen. Damit umfasst  $u$  alle zum Zeitpunkt  $t - 1$  verfügbaren Informationen.

Die resultierende RNN-Einheit ist in Abb. 6.25 dargestellt. Unten links wird zunächst  $u$  gebildet durch Aneinanderhängen des verdeckten Vektors  $h_{t-1}$  und der Eingabe  $x_{t-1}$ . Das Löschgatter wird von  $u_{t-1}$  durch eine vollverbundene Sigmoidschicht gesteuert und löscht einzelne Komponenten des Speichervektors  $c_{t-1}$ . Das Speichergatter wird ebenfalls von  $u_{t-1}$  durch eine andere vollverbundene Sigmoidschicht gesteuert und addiert einzelne Komponenten von  $u_{t-1}$  zum Speicher  $c_{t-1}$ . Das Ausgabegatter wird ebenfalls von  $u_{t-1}$  durch eine weitere vollverbundene Sigmoidschicht gesteuert und erzeugt aus dem Speicher  $c_t$  den aktuellen verdeckten Vektor  $h_t$ .

Um eine zusätzliche Flexibilität zu gewinnen, erweitern Schmidhuber und Hochreiter (1997) das Netz noch um zwei vollverbundene Schichten mit  $\tanh$ -Aktivierung, welche die Verbindung zwischen dem Speicher  $c_t$  und den Ein-/Ausgaben herstellen. Das vollständige LSTM ist in (Abb. 6.26) dargestellt und bildet eine LSTM-Zelle, die an Stelle des einfachen RNN  $h_t = \tanh(U * h_{t-1} + W * x_{t-1} + b)$  verwendet werden kann.



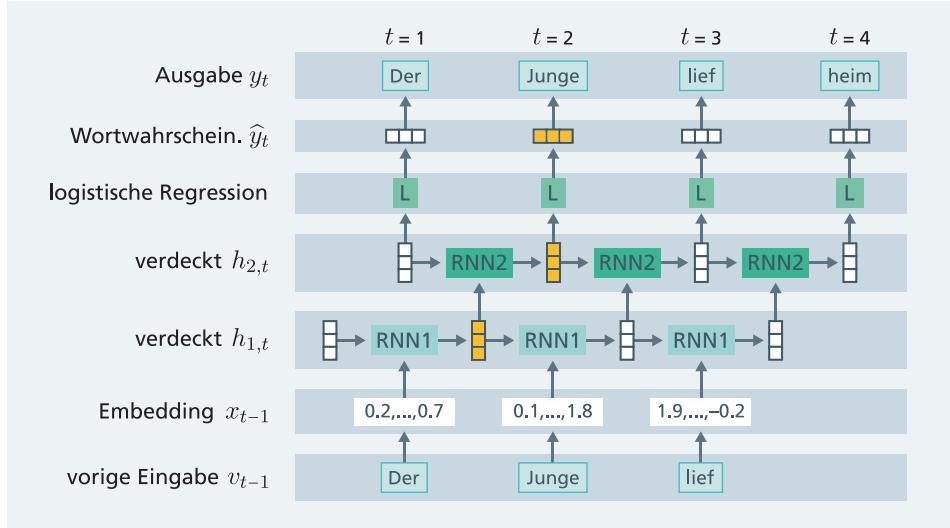
**Abb. 6.26** Das vollständige LSTM enthält neben den Gattern (orange) aus Abb. 6.25 zusätzlich zwei trainierbare tanh-Schichten (blau), um die Eingaben für den Speichervektor zu transformieren und den Speichervektor für das Ausgabegatter vorzubereiten

Der Speichermechanismus des LSTM ist nach seinen Parametern differenzierbar und kann daher durch Training an die Daten angepasst werden. Während des Trainings werden die unbekannten Parameter der Teilnetze  $A_1, \dots, A_5, b_1, \dots, b_5$  durch stochastischen Gradientenabstieg optimiert. Jede Komponente des Speichervektors repräsentiert einen spezifischen Aspekt, der während des Trainings automatisch definiert wird. Die Länge des Speichervektors wird meist zwischen 100 und 2000 gewählt, daher kann der Speichervektor eine Vielzahl von Aspekten speichern. Durch das Training wird auch bestimmt, wie lange ein Aspekt gespeichert bleibt, und wann er in Abhängigkeit von der Eingabe gelöscht wird. Zur Vermeidung von Überanpassung müssen nahezu immer Regularisierungsmethoden verwendet werden. Sehr beliebt ist dabei das Dropout (Abschn. 4.6.4), welches zufällige wechselnde Parameter auf 0 setzt.

Es ist erstaunlich, dass auch für dieses komplizierte Modell das Training mit dem stochastischen Gradientenabstieg funktioniert. Bemerkenswert ist, dass in einem Gatter zwei Größen miteinander multipliziert werden, nämlich der Steuerungsvektor mit dem zugehörigen Eingabevektor. Auch für diese Struktur kann der Gradient berechnet werden. Hierbei wird das Fehlersignal von der Ausgabe eines Gatters aufgespalten und auf die Eingabe sowie auf den Steuerungsvektor zurückpropagiert. Allerdings kann der Rechenaufwand für das Training je nach Problem Stunden oder mehrere Tage betragen.

#### 6.4.2 LSTMs mit mehreren Schichten

Man kann LSTMs verbessern, indem man mehrere Schichten verwendet, deren LSTMs unterschiedliche Parameter enthalten. Hierbei dienen die von der ersten Schicht ausgege-



**Abb. 6.27** LSTM mit 2 Schichten. Um zum Beispiel die Wahrscheinlichkeiten für das Wort an Position ( $t = 2$ ) zu prognostizieren, verwendet RNN1 den verdeckten Vektor  $h_{1,t=1}$  der ersten Schicht und das Embedding des vorigen Wortes „Der“ als Eingabe und generiert den verdeckten Vektor  $h_{1,t=2}$ . Dieser dient dann zusammen mit  $h_{2,t=1}$  als Eingabe für RNN2, welches  $h_{2,t=2}$  prognostiziert. Dieser Vektor wird von dem logistischen Regressionsmodell  $\mathbf{L}$  verwendet, um die Wahrscheinlichkeit des nächsten Wortes an Position  $t = 2$  (beobachtet wurde „Junge“) vorherzusagen. Die so berechneten Vektoren sind orange eingefärbt. Nach diesem Schema werden sukzessive Wahrscheinlichkeiten für sämtliche Worte des Satzes prognostiziert

benen verdeckten Vektoren  $h_{1,t}$  als Eingabe für die LSTM-Zelle der nächsten Schicht (Abb. 6.27). Im Falle weiterer Schichten wird dieses Vorgehen wiederholt. Die berechneten verdeckten Vektoren der obersten Schicht dienen als Eingabe für das logistische Regressionsmodell zur Prognose des nächsten Wortes.

Wie schon bei anderen TNN beobachtet, können zusätzliche Schichten die Abbildungskapazität des mehrschichtigen RNN erhöhen. Während eine zusätzliche Schicht gleicher Größe die Anzahl der Parameter verdoppelt, so wird bei der Verdoppelung der Länge der verdeckten Vektoren und der Speichervektoren die Anzahl der Parameter vervierfacht. Aus diesem Grunde sind mehrere Schichten vorteilhaft und man verwendet bei LSTM-Netzen oft 2–4 Schichten.

### 6.4.3 Anwendungen des LSTM

In den letzten Jahren stand genügend Rechenleistung zur Verfügung, um LSTM-Netze erfolgreich zu trainieren. In einer Anwendung wurde ein LSTM auf 96 MB Text der englischen Wikipedia trainiert, welcher noch Markups für Links zu anderen Artikeln enthielt. Abb. 6.28 zeigt einen vom LSTM erzeugten Text (Karpathy 2015). Der Text ist grammatisch weitgehend korrekt, die Inhalte sind aber zusammenfantasiert. Viele der verwen-

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 Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, 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.

**Abb. 6.28** Dieser Text wurde von einem LSTM erzeugt, welches auf Texten aus der englischen Wikipedia trainiert wurde. Die Texte enthielten Verknüpfungen [[x]] zu anderen Wikipedia-Artikeln

deten Begriffe existieren nicht. Wie man sieht, verlieren LSTMs nach einer Reihe von Worten den Kontext und erzeugen dann Text zu anderen Themen. Mit Hilfe des Modells lassen sich beliebig viele Texte dieser Art generieren.

Ein Standardkorpus zum Test von Sprachmodellen ist der Penn-Treebank-Korpus. Er enthält etwa 4,5 Millionen Worte in amerikanischem Englisch, welche zusätzlich mit syntaktischen Eigenschaften annotiert wurden. Mit diesem Korpus wurden unterschiedliche Arten von Sprachmodellen trainiert, welche jeweils das nächste Wort prognostizieren. Als Maßzahl für die Güte der Prognose des Sprachmodells verwendet man die Perplexität. Das ist die inverse Wahrscheinlichkeit der Testmenge  $w_1, \dots, w_N$ , normalisiert durch die Anzahl der Worte der Testmenge, also  $p(w_1, \dots, w_N)^{-1/N}$ . Je kleiner die Perplexität, desto besser kann das Modell die Testmenge prognostizieren. Abb. 6.29 zeigt die Perplexität für unterschiedliche rekurrente Modelle. Die LSTMs konnten offenbar die Perplexität in den letzten Jahren wesentlich verbessern.

Ein anderes LSTM-Netz operiert auf Zeichenebene: es prognostiziert einen Buchstaben des Textes nach dem nächsten. Es verwendet drei LSTM-Schichten, wobei die Speicher- und verdeckten Vektoren jeweils eine Länge von 512 haben. Als Trainingsmaterial wurden die Werke von Shakespeare im Umfang von 4,4 MB verwendet. Der Text hat den Sprachduktus von Shakespeare gut erfasst und enthält nur wenige orthografische Fehler (Abb. 6.30). Wie beim Wikipedia-Modell geht allerdings der inhaltliche Kontext nach einer Reihe von Worten verloren und das Modell springt zu anderen Themen.

Die grundlegende LSTM-Struktur in Abb. 6.25 (rechts) wurde nach heuristischen Überlegungen konstruiert, und es ist überhaupt nicht klar, ob diese Struktur optimal ist. Melis et al. (2017) haben eine Reihe von rekurrenten neuronalen Netzarchitekturen untersucht

**Abb. 6.29** Perplexität für den Penn Treebank Korpus für unterschiedliche Modelle. Die kleineren Werte sind besser

Autoren	Modell	Anzahl Parameter	Perplexität
(Pascanu et al. 2013)	RNN	6.1 Mio	107.5
(Melis et al. 2017)	LSTM 4 Schichten	10 Mio	60.1
(Melis et al. 2017)	LSTM 4 Schichten	24 Mio	58.3
AWD-LSTM	LSTM 3 Schichten	24 Mio	57.3

PANDARUS: Alas, I think he shall be come approached and the day When little strain would be attain'd into being never fed, And who is but a chain 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.

**Abb. 6.30** Dieser Text wurde von einem LSTM-Netzwerk erzeugt, welches auf den Buchstaben der Dramen von Shakespeare trainiert wurde (Karpathy 2015)

und ihre Performanz verglichen. Dabei hatten alle Modelle die gleiche Anzahl trainierbarer Parameter und die Hyperparameter wurden extensiv optimiert (Abschn. 4.8.3). Hierbei hatten LSTM-Netze leichte Vorteile gegenüber anderen Netzarchitekturen. Insbesondere wurde auch gezeigt, dass Regularisierung durch geeigneten Dropout (Abschn. 4.6.4) sehr wichtig ist.

#### 6.4.4 Bidirektionale LSTM-Netze zur Prognose von Worteigenschaften

Eine Anwendung für RNNs ist die Prognose von Eigenschaften von Sequenzelementen. Jeden Tag tauchen in der Presse und im Internet neue Namen (benannte Entitäten) von Personen, Produkten, Organisationen, Orten usw. auf. Beispielsweise sind in der GeoNames-Datenbank über 10 Millionen geographische Namen verzeichnet und praktisch jedes gebräuchliche Wort der Sprache taucht dort als Ortsname auf. Daher ist es nicht möglich, Namen durch Abgleich mit Listen zu identifizieren, sondern man muss die Eigenschaften von Namen mithilfe von Kontextinformationen erkennen. Diese Aufgabe der Informatonsextraktion wird Eigennamenerkennung oder Named Entity Recognition (NER) genannt (Abb. 6.31).

Ein Benchmark-Datensatz für diese Aufgabe ist der CoNLL-2003 Datensatz. Er besteht aus englischen Agenturmeldungen von Reuters mit 17 Millionen Worten. Jedes Wort wurde manuell mit einer der Kategorien PER (Person), ORG (Organisation), LOC (Ort), MISC (anderer Name) und O (sonstiges Wort) annotiert.

In Abb. 6.32 wird ein bidirektionales LSTM-Modell für diese Aufgabe skizziert. Als Eingabe verwendet es Embeddings für Worte. Lample et al. (2016) schlagen vor, ein „Vorwärts“-LSTM zu verwenden, welches verdeckte Vektoren von vorne nach hinten fort-schreibt und darin die zuvor liegenden Worte und Kategorien speichert. Ein weiteres „Rückwärts“-LSTM propagiert andere verdeckte Vektoren von hinten nach vorne und speichert darin die später im Satz auftauchenden Worte und Kategorien. Die eigentliche Prognose der NER-Kategorien erfolgt durch eine verbesserte Version des logistisches Regressionsmodells, welches die verdeckten Vektoren der beiden LSTMs als Eingabe hat. Ein solches Netz wird bidirektionales LSTM genannt, weil es in zwei Richtungen operiert.

**Joseph Fraunhofer** wurde als elftes Kind eines **Glasermeisters** in **Straubing** geboren. Da seine Eltern starben, als er erst elf Jahre alt war, wurde er von seinem Vormund in die sechsjährige **Spiegelschleiferlehre** nach **München** gegeben. Dort überlebte er **1801** den Einsturz des Hauses seines Lehrherrn. Bei seiner Rettung war **Kurfürst Maximilian IV.** anwesend, der von dem glücklichen Ausgang beeindruckt war und **Joseph Fraunhofer** **18 Dukaten** schenkte. Der **Geheime Rat** und **Unternehmer Joseph von Utzschneider** ermöglichte **Fraunhofer** den Besuch der **Sonntagsschule**.  
**Entitätstypen:** Person, Beruf, Ort, Zeit, Geld, Organisation.

Abb. 6.31 Wikipediatext über Joseph Fraunhofer mit annotierten Namensentitäten

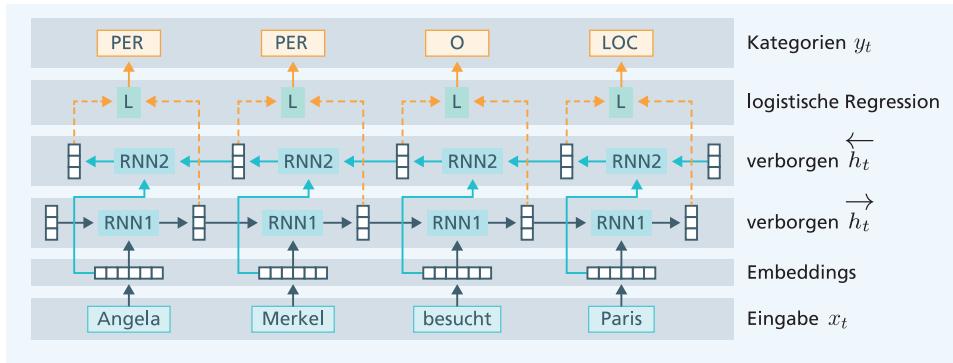


Abb. 6.32 Bidirektionales LSTM (BiLSTM) zur Prognose von Worteigenschaften. Die unterste Schicht prognostiziert vom Satzanfang zum Satzende und die zweite Schicht vom Satzende zum Satzanfang. Beide verdeckten Vektoren werden zur Prognose des Labels durch das logistische Regressionsmodell genutzt

Die Embeddings können im Prinzip mithilfe der CoNLL-Daten berechnet werden. Da der Datensatz aber relativ klein ist, verwenden Lample et al. (2016) vorberechnete Embeddings der Länge 100, die auf sehr großen Dokumentensammlungen trainiert wurden. Zur Berechnung der Embeddings verwenden sie eine Variante von Word2vec, die die Reihenfolge der Worte in der Nachbarschaft berücksichtigt.

Lample et al. (2016) erreichen mit diesem Modell einen F-Wert von 90,2 % für die englischen CoNLL-Daten und von 73,1 % für die deutschen CoNLL Daten. Der F-Wert ist das harmonische Mittel von Präzision und Recall, zusätzlich gemittelt über alle Kategorien. Das englische Modell erkennt also im Schnitt neun von zehn Namen korrekt, während das deutsche Modell nur etwa drei von vier Namen richtig zuordnet.

Man kann mithilfe von RNN auch den Verlauf von Zeitreihen (Abb. 6.33) prognostizieren. Eine Zeitreihe ist eine zeitlich geordnete Folge von einer oder mehreren Messwerten  $x_t$ , z. B. Aktienkurse mehrerer Unternehmen zu bestimmten Zeitpunkten  $t$ . Dabei wird anstelle eines Wort-Embeddings der Vektor  $x_t$  der Zeitreihenwerte zum Zeitpunkt  $t$  als Eingabe des RNN verwendet. Die Ausgabe zum Zeitpunkt  $t$  ist keine Wahrscheinlichkeit von Wörtern, sondern der Vektor der gesuchten Werte, z. B. der Wert  $y_{t+1}$  einer von mehreren Zeitreihen zum nächsten Zeitpunkt. Mushtaq et al. (2019) geben einen Überblick

**Abb. 6.33** Mit rekurrenten neuronalen Netzen können auch Zeitreihen prognostiziert werden, beispielsweise Börsenkurse. Bildnachweis im Anhang



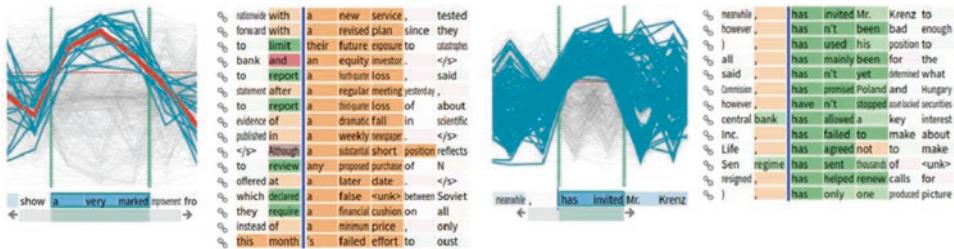
über derartige Ansätze, wobei LSTM-Modelle eine prominente Rolle spielen. Sie können bei Wirtschaftsprägnosen, in der Meteorologie oder auch bei der Überwachung von technischen Prozessen verwendet werden.

#### 6.4.5 Visualisierung von rekurrenten neuronalen Netzen

LSTMs haben umfangreiche Speichervektoren und verdeckte Vektoren und sind in der Lage, komplexe Relationen zwischen Worten und Phrasen zu erfassen. Daher ist es hilfreich, wenn man versteht, wie diese Zusammenhänge in den Vektoren gespeichert werden. Strobelt et al. (2016) haben ein Visualisierungsverfahren für diese Aufgabe entwickelt. Bei einem trainierten Sprachmodell wählt der Anwender eine numerische Schwelle und eine Textstelle in einem Satz aus, z. B. „a very marked“. Das System analysiert diese Textstelle und selektiert diejenigen Komponenten des Speichervektors, die für die Worte der Textstelle eine Aktivierung haben, welche höher ist als die Schwelle. Dies ist oben links in Abb. 6.34 dargestellt.

Anschließend untersucht das System die Aktivierung der ausgewählten Komponenten für den restlichen Text und zeigt Textstellen an (rechte Seite von Abb. 6.34), in denen die Aktivierung der Komponenten ebenfalls hoch ist, z. B. „a new service“ oder „a regular meeting“. Diese Stellen werden durch eine intensive Färbung angedeutet. Offenbar bewirken diese Wortfolgen eine ähnliche Aktivierung der Komponenten des Speichervektors. Diese Stellen enthalten Phrasen mit Substantiven (Nominalphrasen), die fast alle das Wort „a“ enthalten. Im unteren Teil von Abb. 6.34 wird dieses Vorgehen für die Verb-Phrase „has invited“ wiederholt. Hier werden Phrasen wie „has allowed a“ oder „has agreed“ mit ähnlichen Aktivierungen im Speichervektor ausgewählt.

Durch eine flexible Auswahl der Matchkriterien kann die Visualisierung verfeinert und die Bedeutung einzelner Komponenten des Speichervektors verdeutlicht werden. Die Visualisierung kann sowohl für die verdeckten Vektoren als auch die Speichervektoren beliebiger Schichten des RNNs verwendet werden.



**Abb. 6.34** Visualisierung der Aktivierung von Speichervektoren in einem LSTM. Komponenten des Speichervektors mit einer Aktivierung über einer Schwelle für die Worte „a very marked“ bzw. „has invited“ werden ausgewählt. Dies wird durch die Kurven in den rechten und linken Abbildungen gezeigt. Auf den rechten Seiten beider Abbildungen sind Textstellen wiedergegeben, in denen die Aktivierungen ebenfalls hoch sind (intensive Färbung). Hierbei sind zur Illustration Nominalphrasen orange und Verb-Phrasen grün unterlegt. Bildnachweis im Anhang

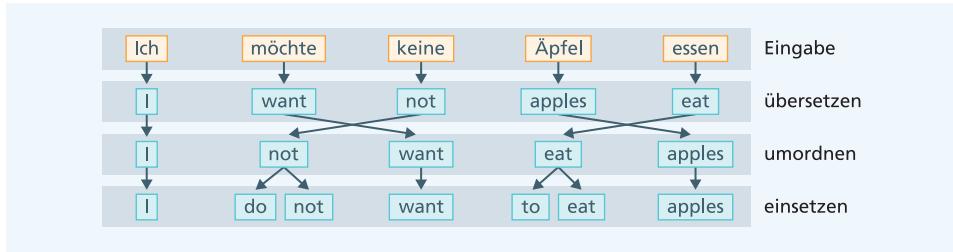
## 6.5 Übersetzung: Transformation einer Sequenz in eine andere Sequenz

Die automatische Übersetzung von Texten von einer Sprache in eine andere ist ein alter Menschheitstraum. Mit der Verfügbarkeit von Computern begann die Forschung hierzu in der Mitte des vorigen Jahrhunderts. Zunächst wurde die maschinelle Übersetzung hauptsächlich im militärischen Bereich verwendet. Seit den 90er-Jahren konnten Übersetzungsprogramme auch online im Internet genutzt werden. Allerdings waren die verfügbaren Systeme oft nicht in der Lage eine befriedigende Übersetzung zu liefern.

Abb. 6.35 zeigt die Schwierigkeiten bei einer Übersetzung. Es reicht nicht, einzelne Worte und Phrasen (abgeschlossene Einheiten aus einem oder mehreren Wörtern) zu übersetzen, sondern die Worte müssen auch umsortiert werden und gegebenenfalls müssen zusätzliche Worte eingefügt und Endungen modifiziert werden. Dies lässt sich meist nicht durch wenige Regeln erfassen. Gebräuchliche, flüssige Formulierungen müssen memo-riert werden, zudem ist ein umfassendes Weltwissen erforderlich. Ein Beispiel ist der Satz „Ich sah den Mann auf dem Hügel mit dem Teleskop“. Hierbei ist nicht klar, ob „sehen mit dem Teleskop“ oder „der Hügel mit dem Teleskop“ gemeint ist. Das kann nur satzüber-greifend aus dem Kontext erschlossen werden.

Kann eine Person eine Sprache in eine andere übersetzen, so muss sie im Grunde beide Sprachen verstehen. Sie muss die Bedeutung und den Kontext der Worte erfassen und Weltwissen haben, um die korrekte Entsprechung von Formulierungen zu kennen. Schon die Griechen im Altertum unterschieden zwischen „Metaphrase“ (wörtlicher Übersetzung) und „Paraphrase“, der Umschreibung eines Inhalts durch Formulierungen, welche den Sinn des Ausgangstextes wiedergeben.

Eine Übersetzung ist oft nicht eindeutig. So vertrat Quine (1969) die These, dass zwischen verschiedenen Übersetzungsvarianten oft keine objektive Reihenfolge festgelegt werden kann. Zudem ist die Sprache im Allgemeinen nur im Kontext der eigenen Erfahrungen interpretierbar. Bar-Hillel (1962) war sogar der Meinung, dass maschinelle Über-



**Abb. 6.35** Eine Übersetzung erfordert oft mehrere Schritte: Nach einer Rohübersetzung müssen die Worte oft umsortiert werden. Danach müssen gegebenenfalls noch zusätzliche Worte und Wortbeugungen eingeführt werden, um die grammatische Struktur der Zielsprache zu beachten

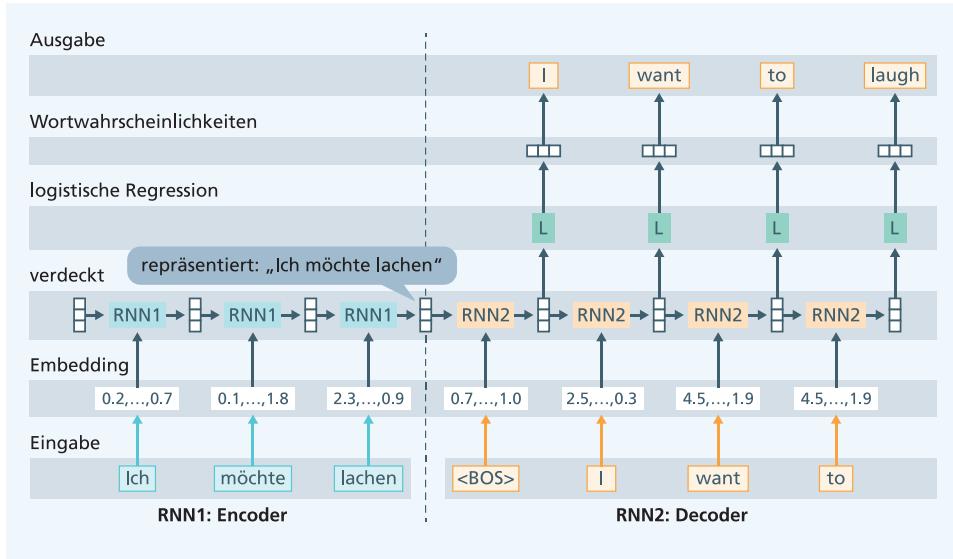
setzung nicht nur praktisch, sondern auch theoretisch unmöglich ist. „Menschliche Übersetzer nutzen ihr Hintergrundwissen häufig unbewusst, um syntaktische und semantische Mehrdeutigkeiten aufzulösen.“ Maschinen können diese Mehrdeutigkeiten oft nicht auflösen oder wenden eine mechanische „Regel an, welche häufig eine falsche Übersetzung ergibt.“ Die Fähigkeit zur Übersetzung einer Sprache ist also nach Bar-Hillel eine Intelligenzleistung, die nicht automatisierbar ist.

### 6.5.1 Sequence-to-Sequence Netze zur Übersetzung

In den 2000er-Jahren wurden zur Übersetzung häufig statistische Verfahren genutzt. Dabei wurden aus großen Sammlungen von Übersetzungen Wörter und Phrasen der Ausgangs- und Zielsprache gemäß ihrer Häufigkeit einander zugeordnet und zur Übersetzung genutzt. Die Genauigkeit dieser Übersetzungsverfahren wurde aber von vielen Nutzern als unzureichend bewertet.

Sutskever et al. (2014) schlugen eine neue Art von TNN vor, die auf rekurrenten neuronalen Netzen aufbaute. Sie gingen von der Beobachtung aus, dass ein rekurrentes neuronales Netz in der Lage ist, einen Satz in einem verdeckten Vektor zu „speichern“. Dies führte zu der Idee, bei der Übersetzung den Satz der Ausgangssprache mit einem Encoder-RNN in einen verdeckten Vektor zu übersetzen. Dieser verdeckte Vektor bildet dann den initialen verdeckten Vektor eines Decoder-RNNs, welches den Satz in die Zielsprache überträgt. Ein solches TNN wird Sequence-to-Sequence Modell genannt (Abb. 6.36).

Das Decoder-RNN prognostiziert die Wahrscheinlichkeit der Ausgabeworte der Zielsprache. Für die Prognose des nächsten Wortes erhält es beim Training die Informationen über das korrekte vorige Wort (oder das Satzanfangssymbol <BOS>) über dessen Embedding und kann dies bei der Berechnung nutzen. Durch die Nutzung der zuvor prognostizierten Worte wird wie – beim Sprachmodell – das Training beschleunigt. So werden nacheinander Wahrscheinlichkeiten für die Folgeworte des Satzes (oder das Satzendesymbol <EOS>) erzeugt. Dies geschieht, bis eine Maximallänge erreicht ist. Das Modell wird als Ganzes in einem Optimierungsprozess trainiert. Ist die prognostizierte Wahrscheinlichkeit des Ausgabewortes gering, so wird das Fehlersignal über das logistische Modell und die verdeckten Vektoren des Decoder-RNNs zurückpropagiert zum Encoder-RNN und schließlich zu den Embeddings. Der Informationsfluss läuft dabei in der entgegengesetzten Rich-



**Abb. 6.36** Ein Sequence-to-Sequence Modell zur Übersetzung eines Satzes. Das Encoder-RNN RNN1 (links) kodiert den Satz der Ausgangssprache in den verdeckten Vektor, der Startwert für den verdeckten Vektor des Decoder-RNNs RNN2 (rechts) ist. Ein logistisches Modell **L** prognostiziert die Wahrscheinlichkeiten der Ausgabewörter. Das vorherige korrekte Wort der Zielsprache wird als Eingabe für die Übersetzung des nächsten Wortes genutzt

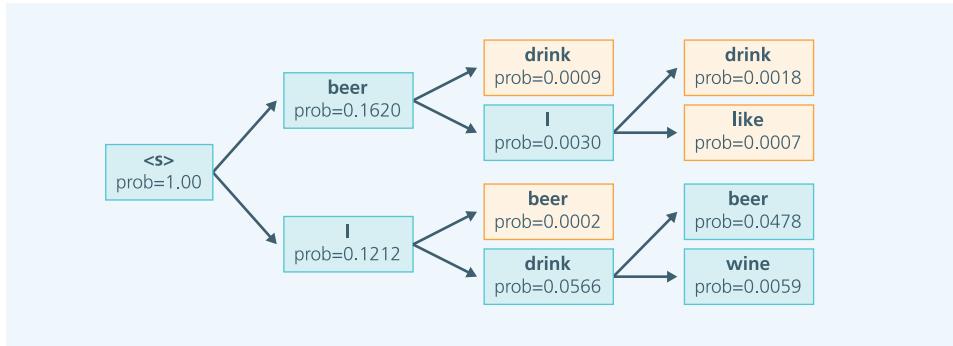
tung der Pfeile in Abb. 6.36. Unbekannte Parameter sind die Embeddings der Worte in beiden Sprachen sowie die Parameter des logistischen Modells, des Decoder-RNNs und des Encoder-RNNs.

Zum Training verwendet das Beispielmodell von Sutskever et al. (2014) einen umfangreichen Datensatz mit 12 Millionen englischen Sätzen und deren französischen Übersetzungen. Es wurde ein festes Vokabular von 160.000 häufigen englischen Wörtern und 80.000 häufigen französischen Wörtern genutzt.

Als Encoder-RNN und Decoder-RNN verwendeten Sutskever et al. (2014) jeweils LSTM-Modelle mit vier Schichten (Abschn. 6.4.2) mit verdeckten Vektoren der Länge 1000 und Embeddings der Länge 1000. Das resultierende RNN hat insgesamt 380 Millionen Parameter, welche durch den stochastischen Gradientenabstieg optimiert werden. Obwohl das Training auf 8 GPUs parallelisiert wurde, dauerte es 10 Tage.

## Erzeugen und Bewertung einer Übersetzung

Ist das Übersetzungsmodell (Abb. 6.36) trainiert, so kann es die Wahrscheinlichkeit einer Wortfolge für die Zielsprache berechnen. Bei der Anwendung wird aber eine konkrete Übersetzung benötigt, Wahrscheinlichkeiten alleine reichen nicht aus. Theoretisch müsste man alle Wortfolgen generieren und diejenige mit der höchsten Wahrscheinlichkeit behalten. Das ist wegen des Rechenaufwandes unmöglich. Die Alternative ist, dass man immer dasjenige Folgewort mit der höchsten Wahrscheinlichkeit auswählt. Das führt aber schnell zu schlechten Ergebnissen. Als Kompromiss kann man eine Beam-Suche durchführen,



**Abb. 6.37** Beam-Suche ( $k = 2$ ) zur Erzeugung einer Übersetzung des Satzes „Ich trinke Bier“. Zur Prognose des nächsten Wortes werden immer die jeweils  $k$  Folgeworte mit den höchsten Wahrscheinlichkeiten erzeugt. Anschließend werden alle Pfade bis auf die  $k$  Pfade mit den höchsten Pfad-Wahrscheinlichkeiten gelöscht (orange). Diese beiden Schritte werden für jeden der  $k$  Pfade wiederholt, solange bis für alle Pfade das Satzende erreicht wurde. Am Schluss wird der Pfad mit der höchsten Gesamtwahrscheinlichkeit gewählt

welche immer  $k$  Alternativen parallel verfolgt. In Abb. 6.37 wird dies an einem kleinen Beispiel mit  $k = 2$  demonstriert. Für  $k$  sind Werte zwischen 2 und 20 gebräuchlich.

Es wird nun ein Maß benötigt, mit dem man die Übersetzungsqualität durch eine Zahl beurteilen kann. Hierzu vergleicht der sogenannte BLEU-Wert die Anzahl der gleichen Wortfolgen der Länge 1–4 in der erzeugten Übersetzung mit den entsprechenden Anzahlen in einer oder mehreren Referenzübersetzungen und rechnet daraus eine Zahl zwischen 0 und 1 aus. Die Wortfolgen bewerten die Übereinstimmungen in geläufigen Formulierungen. Dabei wird noch eine Korrektur vorgenommen, um längeren, repetitiven Übersetzungen ein geringeres Gewicht zu geben. In Bewertungsexperimenten korrelierte BLEU gut mit menschlichen Bewertungen der Übersetzungsqualität.

Die Übersetzungsresultate des Modells sehen auf den ersten Blick sehr gut aus. Abb. 6.38 zeigt eine vom Modell erzeugte Übersetzung und die Referenzübersetzung aus den Testdaten (Sutskever et al. 2014). Dabei gilt wie immer, dass das Modell auf diesen Satz nicht während des Trainings zugreifen konnte.

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 .

**Abb. 6.38** Beispiel einer durch das Modell erzeugte Übersetzung eines Satzes aus den Testdaten vom Englischen ins Französische (oben) und die zugehörige Referenzübersetzung (unten)

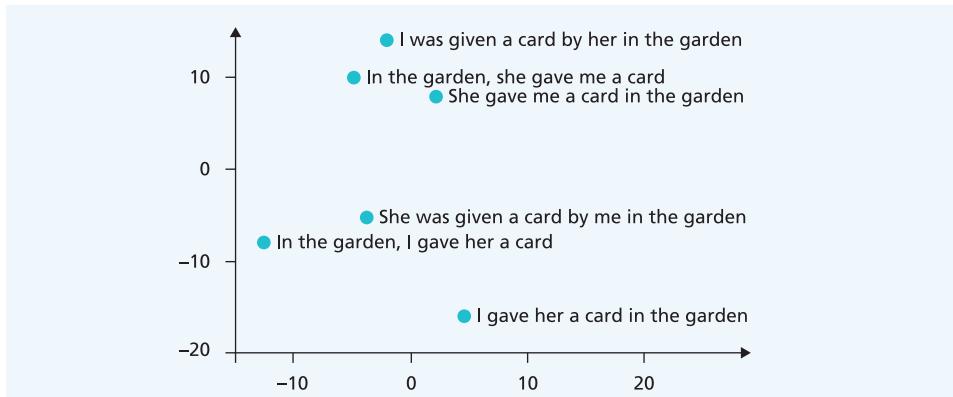
Modell	BLEU Wert
Bestes klassisches Modell	33,3%
Einzelnes LSTM	30,6%
Ensemble von 5 LSTMs	34,8%

**Abb. 6.39** Übersetzungsqualität vom Englischen ins Französische für verschiedene Modelle (Sutskever et al. 2014)

Die formale Bewertung der Übersetzungsqualität mit dem BLEU-Wert auf den Testdaten ergab die Ergebnisse in Abb. 6.39. Während ein einzelnes LSTM einen BLEU-Wert von 30,6 % erzielte, konnte ein Ensemble von fünf LSTMs einen BLEU-Wert von 34,8 % erreichen. Bei einem solchen Ensemble (Abschn. 5.11.2) werden mehrere (hier 5) Modelle unabhängig voneinander mit unterschiedlichen Startwerten trainiert. Bei der Prognose werden die Wahrscheinlichkeiten der Einzelmodelle für die Ausgabeworte am Schluss gemittelt. Dieses Ergebnis zeigte, dass ein neuronales Übersetzungsmodell ein ausgereiftes klassisches Modell übertreffen kann. Es stellte sich heraus, dass das Modell auch lange Sätze mit bis zu 80 Wörtern gut übersetzen konnte. Wegen des beschränkten Wortschatzes hatte das Modell Probleme mit Wörtern, welche nicht im Vokabular waren. Wenn man dieses Problem beseitigen kann, so steigt der BLEU-Wert weiter.

Eine nützliche Eigenschaft des Modells ist, dass es lernt, einen Eingabesatz variabler Länge durch einen Vektor fester Länge zu repräsentieren. Da es immer mehrere Übersetzungen gibt, welche als eine Umschreibung des Ausgangssatzes verstanden werden können, so kann man erwarten, dass die Repräsentation die Bedeutung eines Satzes widerspiegelt.

Abb. 6.40 zeigt die Embeddings unterschiedlicher Sätze in einer zweidimensionalen Projektion. Je nachdem, ob „ich“ oder „sie“ das Subjekt des Satzes ist, liegen die Sätze in



**Abb. 6.40** Zweidimensionale Projektion unterschiedlicher verdeckter Vektoren nach der Verarbeitung der Sätze durch das Encoder-RNN. Die Embeddings der Sätze sind automatisch nach ihrer Bedeutung gruppiert

unterschiedlichen Clustern. Das Modell von Sutskever et al. (2014) erkennt, dass eine Passivformulierung („I was given a card by her“) und eine Aktivformulierung („she gave me a card“) nahezu das gleiche ausdrücken. Auch die interne Struktur der Cluster ist ähnlich. Dies zeigt, dass die Satz-Embeddings die Bedeutung der Sätze gut repräsentieren.

### 6.5.2 Attention: Verbesserung der Übersetzung durch Rückgriff auf die Eingabeworte

Das Sequence-to-Sequence Modell (Abb. 6.36) kodiert den Eingabesatz in einen verdeckten Vektor, der dann als Startvektor für das Decoder-RNN verwendet wird. Das funktioniert gut für kurze und mittellange Sätze. Für lange Sätze wird der verdeckte Vektor aber ein Flaschenhals. Daher kam die Idee auf (Bahdanau et al. 2015), die zuvor berechneten verdeckten Vektoren des Encoder-RNN zur Übersetzung zu nutzen.

Man benötigt nun einen Mechanismus, wie man zu einem Ausgabewort, welches vom Decoder-RNN erzeugt werden soll, einen Rückbezug zu den kontextsensitiven Embeddings der Worte des Eingabesatzes erhalten kann.

In Abb. 6.41 wird diese Vorgehen schematisch gezeigt. Zunächst kodiert das Encoder-LSTM **E** den Eingabesatz in die verdeckten Vektoren  $h_1^{enc}, \dots, h_4^{enc}$ . Das Decoder-LSTM **D** erhält  $h_4^{enc}$  als Startwert und das Satzanfangssymbol „BOS“ als Eingabe und erzeugt den ersten verdeckten Vektor  $h_1^{dec}$ , welcher zur Prognose des ersten Wortes („Ich“) genutzt werden soll. Nun wird, wie in Abb. 6.41 gezeigt, die Ähnlichkeit der verdeckten Vektoren  $h_1^{enc}, \dots, h_4^{enc}$  mit dem verdeckten Vektor  $h_1^{dec}$  des Encoder-RNNs berechnet. Hierzu wird eine Scorefunktion verwendet:

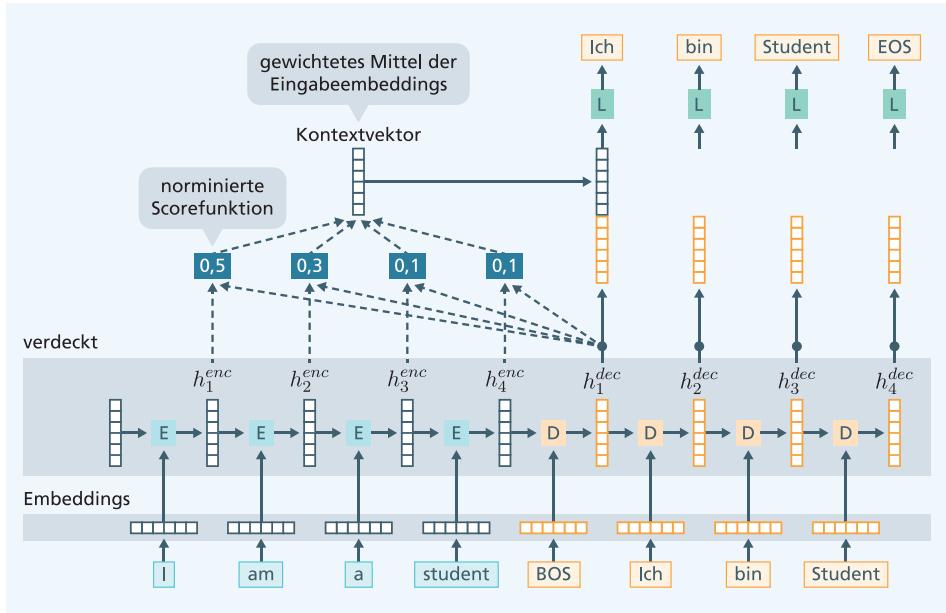
$$\text{score}(h_t^{enc}, h_1^{dec}) = a' * \tanh(W_1 h_t^{enc} + W_2 h_1^{dec})$$

Der Vektor  $a$  und die Matrizen  $W_1$  und  $W_2$  sind Parameter, welche während des Trainings optimiert werden. Die Ergebnisse werden durch eine Softmaxfunktion normiert

$$(\alpha_1, \dots, \alpha_T) = \text{softmax} \left[ \text{score}(h_1^{enc}, h_1^{dec}), \dots, \text{score}(h_4^{enc}, h_1^{dec}) \right]$$

sodass die sogenannten Attentiongewichte ( $\alpha_1, \dots, \alpha_T$ ) alle größer als Null sind und 1 als Summe haben. Sie können als die relative „Aufmerksamkeit“ angesehen werden, welchen die einzelnen verdeckten Vektoren des Encoders vom Zielwort („Ich“) aus erhalten.

Als nächstes bildet man ein gewichtetes Mittel der verdeckten Vektoren des Eingabesatzes  $\alpha_1 * h_1^{enc} + \dots + \alpha_4 * h_4^{enc}$ . Dieser sogenannte Kontextvektor fasst die Information in den verdeckten Vektoren der Eingabe zusammen, die für die Prognose des nächsten Wortes („Ich“) genutzt werden kann. Der Kontextvektor wird an den verdeckten Vektor  $h_1^{dec}$  angehängt und durch das logistische Regressionsmodell L für die Prognose des nächsten Wortes „Ich“ in der Zielsprache verwendet. Diese Berechnungen werden für alle Worte



**Abb. 6.41** Einbeziehung von Encoder-Informationen durch Attention. Zunächst berechnet der Encoder **E** die verdeckten Vektoren  $h_1^{enc}, \dots, h_4^{enc}$  der Eingaben. Der Decoder **D** berechnet dann den verdeckten Vektor  $h_1^{dec}$  aus dem die Wahrscheinlichkeiten für das erste Wort (beobachtet: „Ich“) erzeugt werden soll. Durch die Scorefunktion werden normierte Scorewerte  $score(h_t^{enc}, h_1^{dec})$  für jeden der verdeckten Vektoren  $h_1^{enc}, \dots, h_4^{enc}$  mit  $h_1^{dec}$  bestimmt. Diese „Attentiongewichte“ zeigen die Relevanz des jeweiligen verdeckten Vektors  $h_t^{enc}$  des Eingabesatzes für die Erzeugung des nächsten Ausgabewortes an. Mit den Attentiongewichten wird ein gewichtetes Mittel der verdeckten Eingabevektoren  $h_1^{enc}, \dots, h_4^{enc}$  berechnet, der Kontextvektor. Der Kontextvektor wird zusammen mit  $h_1^{dec}$  zur Prognose des Ausgabewortes („Ich“) durch das logistische Regressionsmodell **L** genutzt. Dies wird für alle Ausgabeworte wiederholt

des Ausgabesatzes wiederholt. Die Erzeugung neuer Worte stoppt, wenn das Satzende-Symbol „EOS“ erzeugt worden ist.

Der Begriff Aufmerksamkeit (Attention) ist laut Wikipedia die Zuweisung von (beschränkten) Bewusstseinsressourcen auf Bewusstseinsinhalte. Diese Berechnung der Beziehung zwischen dem verdeckten Decodervektor  $h_1^{dec}$  und den verdeckten Encodervektoren  $h_t^{enc}$  nennt man daher Attention. Dies hat aber nur entfernte Ähnlichkeit zur menschlichen Fähigkeit der Fokussierung, da es sich hier um rein statistische Korrelationen handelt. Da zwei Sätze miteinander in Beziehung gesetzt werden, spricht man auch von Kreuzattention.

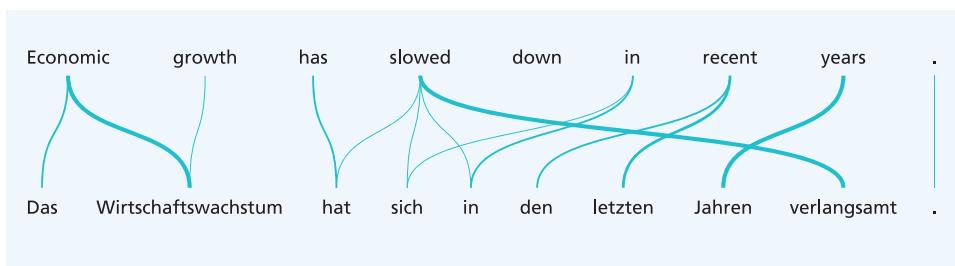
Das Modell wird wieder als Ganzes trainiert. Um den Informationsfluss während der Gradientenberechnung zu verdeutlichen, muss man einfach die Richtung aller Pfeile in Abb. 6.41 umdrehen. Die Verlustfunktion setzt das Ziel, die Wahrscheinlichkeit der korrekten Worte des Ausgabesatzes möglichst stark zu erhöhen. Die Fehlerinformationen werden von den Wortwahrscheinlichkeiten hin zu den Kontextvektoren und den verdeck-

ten Vektor  $h_i^{dec}$  des Decoders und den Ausgabeembeddings zurückpropagiert. Dabei werden auch die Parameter  $a$ ,  $W_1$ ,  $W_2$  der Scorefunktion ebenfalls so optimiert, dass die Wahrscheinlichkeiten der korrekten Worte des Ausgabesatzes möglichst hoch sind. Die Fehlerinformationen werden dann zurück in das Encoder-RNN und die Eingabeembeddings propagiert. Die unbekannten Parameter des logistischen Modells, der Scorefunktion, der beiden RNNs und der Eingabe- und Ausgabeembeddings werden auf diese Weise mit stochastischem Gradientenabstieg trainiert.

### 6.5.3 Übersetzungsergebnisse mit Attention

Obwohl die verdeckten Vektoren den kompletten Satz kodieren, so sind sie doch besonders stark vom gerade einbezogenen Eingabewort beeinflusst. Daher kann man auch die verdeckten Vektoren den jeweiligen Eingabeworten zuordnen. Abb. 6.42 zeigt für einen englischen Satz die Attentiongewichte, die sich bei der Übersetzung ins Deutsche ergeben. Es wird deutlich, dass das Modell automatisch die inhaltliche Relation zwischen den Worten der Sprachen weitgehend rekonstruierten konnte. Das Netz musste diese Zuordnung aus dem übergeordneten Ziel, der Übersetzung des Eingabesatzes in den Satz der Zielsprache, ableiten. Es wird auch klar, dass diese „weiche“ Zuordnung besser ist als eine „harte“ Eins-zu-eins-Zuordnung, weil in vielen Fällen ein Wort eine semantische Beziehung zu mehreren anderen besitzt. Damit lassen sich auch Phrasen unterschiedlicher Länge einander zuordnen.

Es gibt eine Reihe von Möglichkeiten, dieses Basismodell der Attention zu verbessern. Das in Abb. 6.41 vorgestellte Modell hat den Nachteil, dass der Encoder bei der Erzeugung der Embeddings nicht die Worte des Eingabesatzes berücksichtigen kann, welche im weiteren Verlauf des Satzes erscheinen. Beispielsweise ist bei der Erzeugung von  $h_1^{enc}$  aus „I“ nicht bekannt, dass als nächstes Wort „am“ erscheint. Aus diesem Grund wird häufig ein bidirektionales LSTM verwendet (Abschn. 6.4.4), welches mit seinem Vorwärts-LSTM die verdeckten Vektoren  $\tilde{h}_1, \dots, \tilde{h}_T$  und mit seinem Rückwärts-LSTM die verdeckten Vektoren  $\bar{h}_T, \dots, \bar{h}_1$  erzeugt. Die Scorefunktion wird dann auf die aneinandergehängten verdeckten Vektoren  $[\tilde{h}_i, \bar{h}_i]$  angewendet. Mit diesem Ansatz erreicht man durchweg bessere Resultate als mit einem einfachen LSTM.



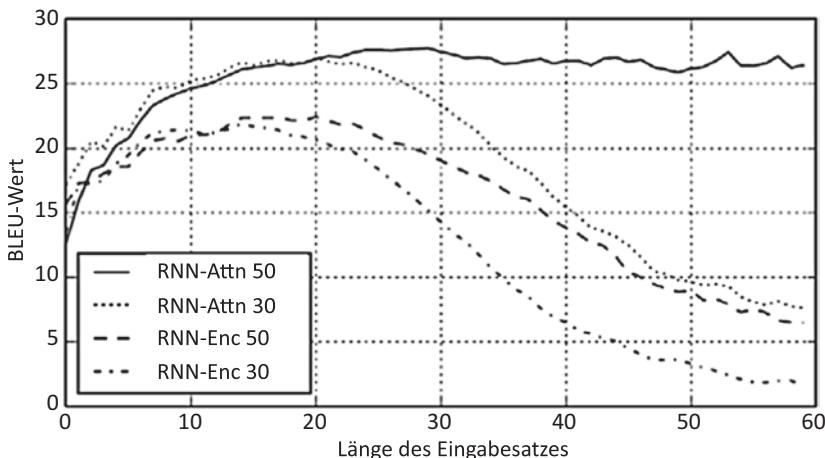
**Abb. 6.42** Attention-Gewichte bei der Übersetzung eines englischen Satzes ins Deutsche. Die Dicke der Verbindungslien symbolisiert die Größe der Attention-Gewichte. Bildnachweis in Kap. 12

Eingabe	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.
Referenz-Übersetzung	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.
Übersetzung durch größeres RNN mit 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.

**Abb. 6.43** Ein Satz aus der Testmenge (oben) mit Referenziübersetzung (Mitte) und der vom größeren Modell mit Attention gelieferten Übersetzung (unten)

Des Weiteren wurden von Bahdanau et al. (2015) mehrere Schichten von LSTMs als Encoder und Decoder verwendet, wobei die verdeckten Vektoren eine Länge von 1000 hatten. Das Modell wurde durch stochastischen Gradientenabstieg mit Minibatches von jeweils 80 Sätzen trainiert. Als Trainingsdatensatz wurde der WMT-2014 Datensatz für Englisch-Französisch mit 850 Millionen Worten genutzt. Das Training dauerte etwa 5 Tage. Abb. 6.43 zeigt zur Illustration einen englischen Satz aus der Testmenge, die von einem Dolmetscher gelieferte Referenziübersetzung, und die von dem größeren RNN-Modell mit Attention generierte Übersetzung ins Französische.

Abb. 6.44 zeigt die mit dem BLEU-Wert gemessene Übersetzungsqualität in Abhängigkeit von der Länge des Eingabesatzes. Es wurden die üblichen Sequence-to-Sequence Modelle (RNN-Enc) und die Modelle mit Attention (RNN-Attn) verglichen. Dabei wurden je zwei Modelle trainiert, einmal mit Sätzen bis zur Länge von 30 Wörtern und zum anderen mit Sätzen bis zur Länge 50. Die Ergebnisse zeigen, dass die Modelle mit Atten-



**Abb. 6.44** Vergleich der BLEU-Werte verschiedener Modelle in Abhängigkeit von der Länge des Eingabesatzes bei einer Übersetzung von Englisch nach Französisch. Das große Attention-Modell (RNN-Attn 50) ist besser als die beiden Standardmodelle (RNN-Enc) und hat gleichbleibende Qualität für alle Satzlängen. Bildnachweis im Anhang

BLEU-Wert für Ausgabe-Sprache							
	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				

**Abb. 6.45** Ergebnisse des WMT 2017 Wettbewerbs auf den Newstest 2018 Daten für die Sprachen Tschechisch (CZ), Deutsch (DE), Englisch (EN), Finnisch (FI), Russisch (RU), Türkisch (TR), Chinesisch (CH). In der ersten Spalte steht die Eingabe-Sprache, in der zweiten Zeile die jeweiligen Ausgabesprachen. Bildnachweis im Anhang

tion schon ab einer Satzlänge ab 8 besser abschneiden als die Standardmodelle. Das RNN-Attn-50 Modell hat für alle Satzlängen eine etwa gleich hohe Performanz, die auch für Sätze bis zur Länge 60 gehalten wird. Dies zeigt, dass die Verwendung von Attention die Übersetzungen nachhaltig verbessert. Schon dieses frühe Übersetzungsmodell RNN-Attn 50 schneidet etwas besser ab als das damalige beste konventionelle Übersetzungsmodell Moses, wenn die Worte des Satzes im Vokabular enthalten waren.

In Abb. 6.45 sind die Ergebnisse des WMT-2017 Wettbewerbs für unterschiedliche Sprachen gezeigt (Bojar et al. 2017). Es handelt sich hier um andere Trainingsdaten als oben. Für Sprachen mit einer komplexen Grammatik und vielen Endungen sind die Ergebnisse nicht so gut wie bei weniger komplexen Sprachen wie etwa Englisch, Chinesisch und Russisch. Eine Ausnahme ist Deutsch als relativ komplexe Sprache, für welche aber auch viele Trainingsdaten zur Verfügung stehen.

Es zeigt sich, dass der Attention-Mechanismus die Qualität der Übersetzungen nachhaltig verbessern kann, insbesondere für lange Sätze. Das Modell ist nicht mehr gezwungen, sämtliche Informationen in den verdeckten Vektor zu kodieren und für die gesamte Sequenz zu speichern, sondern es reicht, die wesentlichen Informationen in den verdeckten Ausgabevektoren fortzuschreiben und die Details durch den Attention-Mechanismus aus dem Eingabesatz zu gewinnen. Zur Architektur von RNN-Übersetzungssystemen führten Britz et al. (2017) einen groß angelegten Vergleich durch und kamen zu den in Abb. 6.46 zusammengefassten Ergebnissen.

- Embeddings/verdeckte Vektoren bis zur Länge von 2048 bringen eine kleine Erhöhung der BLEU-Werte.
- LSTM-Zellen sind konsistent besser als andere RNNs.
- Bidirektionale Encoder mit 2–4 Schichten sind besser als andere Kombinationen.
- Decoder mit 4 Schichten sind am besten.
- Eine gut abgestimmte Beam-Suche mit einer Breite von 5–10 ist mitentscheidend.

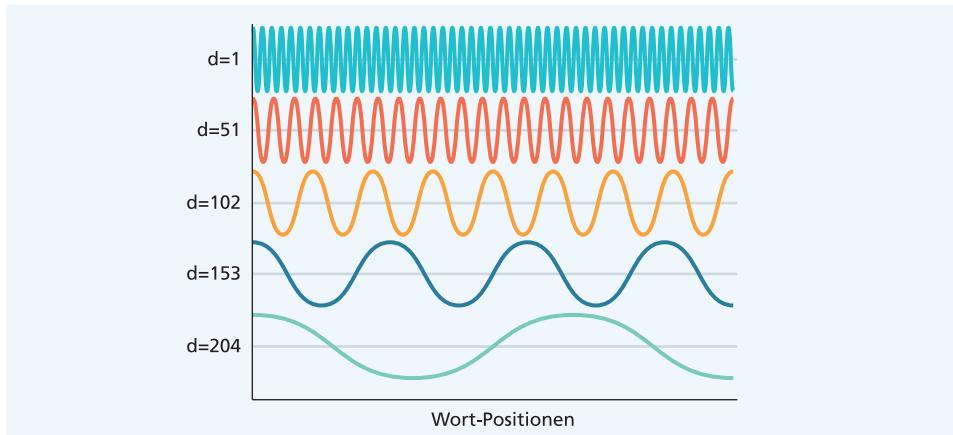
**Abb. 6.46** Wichtigste Ergebnisse eines Vergleichstests von für RNN-Übersetzungssysteme

## 6.6 Transformer-Übersetzungsmodelle

Der Attention-Mechanismus hat sich als sehr effektiv für die Verbesserung der Übersetzung mit einem RNN erwiesen. Er hat aber Probleme bei sehr langen Sätzen und benötigte einen erheblichen Rechenaufwand zum Training.

Vaswani et al. (2017) haben das Transformer-Übersetzungsmodell entwickelt, welches den Satz nicht in einen verdeckten Vektor kodiert, aber dafür sehr viele Attention-Berechnungen durchführt. Dabei wird zunächst die „Korrelation“ der Worte eines Satzes untereinander mit einer „Selbstattention“ evaluiert und hiermit für jedes Wort ein neues „kontextsensitives“ Embedding abgeleitet. Anschließend wird über eine „Kreuzattention“ die Beziehung zwischen den Wörtern des Eingabesatzes und des Ausgabesatzes hergestellt. Das Modell benötigt wesentlich geringere Trainingszeiten und hat ein hohes Verbesserungspotential für die Qualität der Übersetzungen. Es wird in den meisten aktuellen Sequenzanwendungen verwendet. Daher wird es hier trotz seiner komplexen Architektur dargestellt.

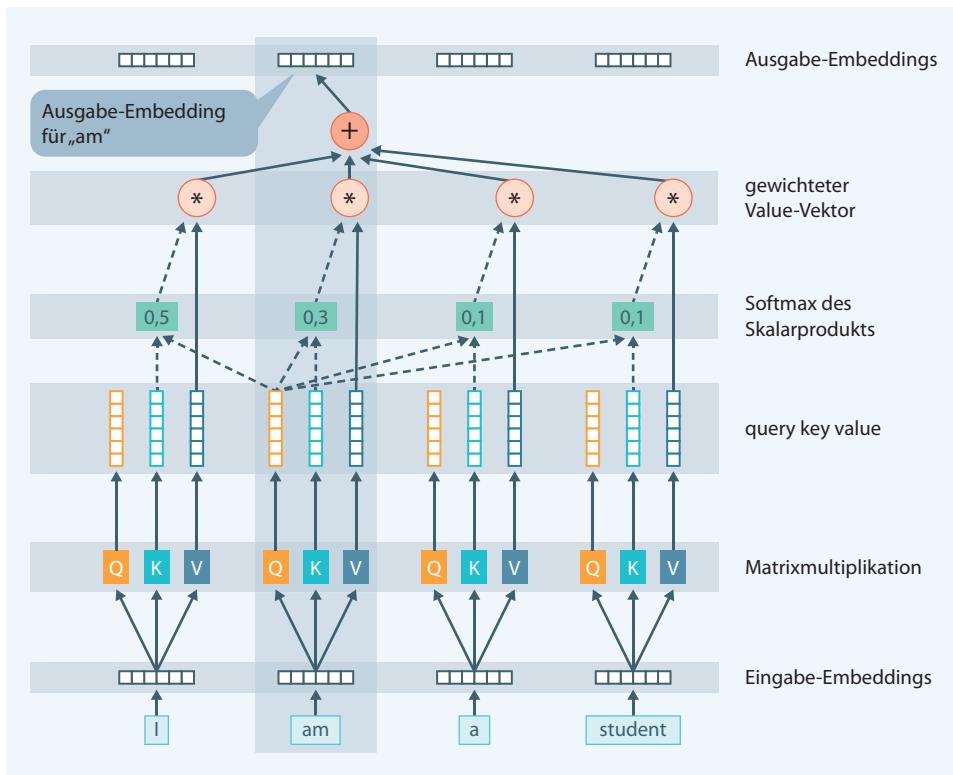
Als unterste Eingabe werden die Embeddings der Eingabeworte verwendet. Auf diese werden gleichlange Positions-Kodierungen für jede Position im Satz aufaddiert und bilden die Eingabe-Embeddings für das Modell. Eine Positions-Kodierung ist erforderlich, damit der Algorithmus die relative Position der Eingabeworte berücksichtigen kann. Als Positions-Kodierungen werden Sinus- und Kosinuswerte unterschiedlicher Frequenzen genutzt (Abb. 6.47). Der Vorteil ist, dass man hiermit beliebige Positions-differenzen durch lineare Funktionen berechnen kann. Bei allen Eingabe-Embeddings (der untersten Schicht) in den folgenden Abschnitten handelt es sich um die Summe von Wortembeddings und Positionse-embeddings.



**Abb. 6.47** Sinus- und Kosinus-Kurven unterschiedlicher Frequenz werden als Positions-Kodierung verwendet

### 6.6.1 Selbstattention analysiert die Abhängigkeiten der Worte eines Satzes

Abb. 6.48 zeigt die Berechnung der Selbstattention an einem kleinen Beispiel (Vaswani et al. 2017). Die Eingabe-Embeddings werden mit drei Matrizen  $Q$ ,  $K$  und  $V$  multipliziert und ergeben drei Vektoren: „query“ (orange), „key“ (hellblau) und „value“ (dunkelblau). Typischerweise sind die query-Vektoren und die key-Vektoren gleich lang, aber kürzer als die Embeddings. Soll nun ein neues Embedding für das zweite Wort „am“ berechnet werden, so bildet man das Skalarprodukt des query-Vektors von „am“ mit den key-Vektoren der anderen Worte.



**Abb. 6.48** Berechnung der Selbstattention für das Wort „am“. Jedes der Wörter wird durch ein Eingabe-Embedding repräsentiert. Dieses Eingabe-Embedding wird mit den drei Matrizen  $Q$ ,  $K$ ,  $V$  multipliziert und ergibt drei Vektoren „query“ (orange), „key“ (hellblau) und „value“ (dunkelblau). Um für das Wort „am“ ein neues Embedding zu berechnen, wird das Skalarprodukt aus der „query“ von „am“ und den „key“-Vektoren aller Worte gebildet und mit Softmax normiert. Mit diesen berechneten Gewichten wird ein gewichtetes Mittel aller value-Vektoren gebildet welches als neues Ausgabe-Embedding für „am“ genutzt wird. Diese Berechnungen werden für alle Eingabewörter wiederholt, um für jedes Wort ein Ausgabeembedding zu erzeugen

$$\gamma_t = [Q * \text{emb}("am")]' * [K * \text{env}(w_t)]$$

Skalarprodukt

Eingabeembedding von "am"      Eingabeembedding von  $w_t$

Es ergeben sich skalare Werte  $\gamma_t$ , die mit der Softmaxfunktion zu Gewichten  $(\tilde{\gamma}_1, \dots, \tilde{\gamma}_T)$  normiert werden. Dann bildet man eine gewichtete Summe der value-Vektoren mit diesen Gewichten und erhält den neuen Embeddingvektor:

$$\text{emb}^{\text{neu}}("am") = \tilde{\gamma}_1 * \text{emb}(w_1) + \dots + \tilde{\gamma}_T * \text{emb}(w_T)$$

Embedding von  $w_1$

neues Embedding von "am"      Gewichtungsfaktor

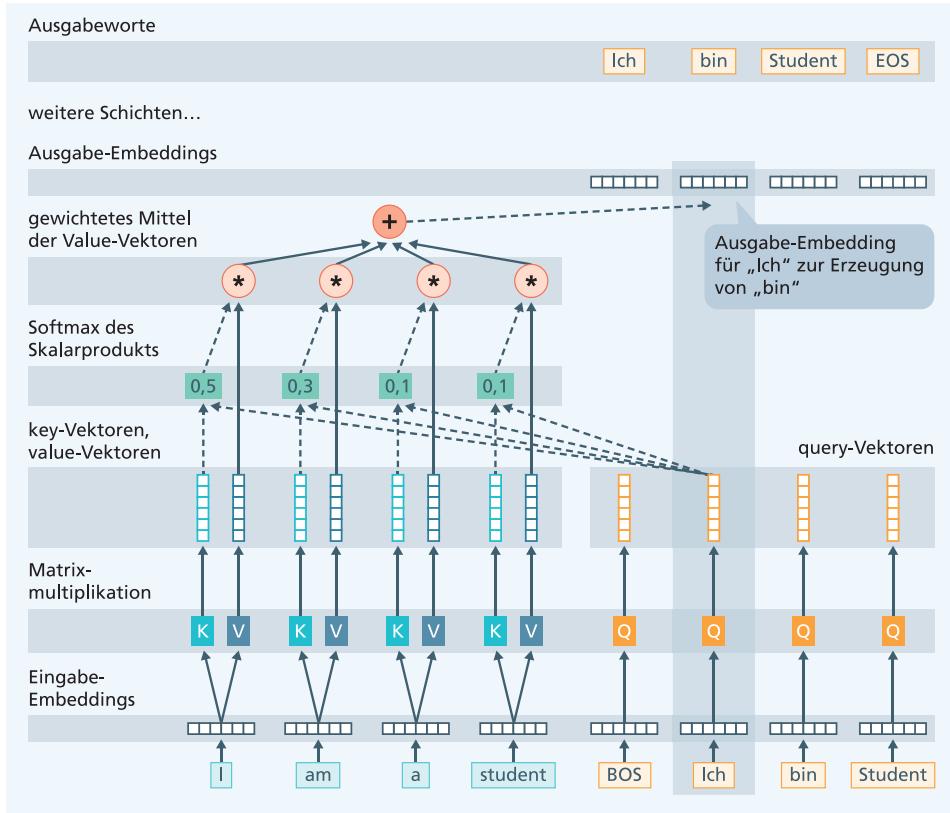
Diese Rechnung wiederholt man parallel für alle Worte des Satzes. Die Matrizen  $Q$ ,  $K$ ,  $V$  sind Parameter, die gelernt werden müssen.

Diese Art von Attention unterscheidet sich also nur in zwei Aspekten von dem einfachen Attention-Modell in Abb. 6.41. Zum einen werden die Embeddings zu Beginn mit den Matrizen  $Q$ ,  $K$ ,  $V$  multipliziert. Dabei entsprechen  $Q$  und  $K$  den Matrizen  $W_2$  und  $W_1$  der Scorefunktion. Zum anderen handelt es sich um Selbstattention, die für Worte des gleichen Satzes berechnet wird, und nicht um Kreuzattention, die zwei Sätze miteinander in Beziehung setzt. Das Schema kann aber, wie weiter unten gezeigt, auch auf Kreuzattention übertragen werden.

## 6.6.2 Kreuzattention analysiert die Abhängigkeiten zwischen Ein- und Ausgabe

Bei der Kreuzattention wird das vorherige Wort der Ausgabesequenz (Übersetzung) mit jedem Wort der Eingabesequenz in Beziehung gesetzt. Die Berechnungen werden analog zur Selbstattention in Abb. 6.48 durchgeführt, mit dem Unterschied, dass die Query-Vektoren aus den Embeddings der Ausgabesequenz berechnet werden. Abb. 6.49 verdeutlicht das Vorgehen am Beispiel der Berechnung des neuen Embeddings für das Ausgabewort „Ich“. Hierzu wird zunächst das Embedding von „Ich“ mit der  $Q$ -Matrix multipliziert, um den query-Vektor zu erhalten. Anschließend werden die Embeddings der Eingabesequenz mit der  $K$ -Matrix sowie der  $V$ -Matrix multipliziert, um die key- bzw. die value-Vektoren zu berechnen. Danach wird ein Skalarprodukt der key-Vektoren und des query-Vektors gebildet und durch Normierung mit Softmax die Gewichte  $(\tilde{\gamma}_1, \dots, \tilde{\gamma}_T)$  berechnet. Diese Gewichte werden mit den value-Vektoren multipliziert, wodurch sich das neue Embedding für das Wort „Ich“ ergibt. Dieses Embedding wird dann nach mehreren Schichten zur Bestimmung des nächsten Ausgabewortes („bin“) verwendet.

Die Positionsdifferenz der Worte wird dabei durch die Positions-Kodierung berücksichtigt. Die Berechnung der Abhängigkeit zwischen weit auseinanderliegenden Wörtern



**Abb. 6.49** Die Transformer-Kreuzattention berechnet für das Decoder-Embedding des vorherigen Wortes („Ich“) der Ausgabesequenz (rechts) eine Attention, um ein neues Embedding zu bestimmen. Dabei greift die Kreuzattention zurück auf die Encoder-Embeddings der obersten Schicht der Eingabesequenz. Die Berechnungen laufen analog zu Abb. 6.48 mit dem Unterschied, dass der query-Vektor (orange) aus dem Embedding des vorherigen Wortes der Ausgabesequenz („Ich“) berechnet wird. Hiermit wird ein neues Embedding für „Ich“ generiert, welches dann (nach mehreren Schichten) zur Erzeugung des gesuchten Wortes „bin“ genutzt werden kann

erfolgt direkt durch die Bildung der Skalarprodukte der jeweiligen Embeddings. Bei einem RNN hingegen muss die Prognose des RNN sehr oft nacheinander ausgeführt werden, um die Relation zwischen zwei weit auseinanderliegenden Wörtern zu bestimmen. Dies erfordert einen höheren Rechenaufwand. Durch die mehrfache Anwendung des Modells werden zudem die unterschiedlichen Beziehungen verschiedener Worte überlagert und immer ungenauer. Daher hat Selbstattention das Potential, die inhaltlichen Relationen der Worte besser zu erfassen.

Es hat sich gezeigt, dass man nicht nur eine Attention-Berechnung pro Schicht ausführen sollte, sondern  $k$  gleichzeitig. Dies wird Multi-Head-Attention genannt. Die Länge der value-Vektoren wird dann als  $\text{Länge}(\text{Embedding}) / k$  gewählt. Damit kann man aus den  $k$  resultierenden gewichteten Mittelwerten der value-Vektoren durch Aneinanderhängen ein neues Embedding bilden.

### 6.6.3 Transformer-Architektur nutzt Selbst- und Kreuzattention

Der Transformer (Vaswani et al. 2017) ist ein TNN zur Übersetzung von Sätzen einer Eingabesprache (z. B. Englisch) in die Zielsprache (z. B. Deutsch). Abb. 6.50 zeigt seine Architektur. Er besteht aus sechs aufeinanderfolgenden Encoder-Blöcken und anschließend aus sechs aufeinanderfolgenden Decoder-Blöcken. Das Modell verwendet Embeddings der Länge 512.

*Encoder:* Der Encoder der untersten Schicht erhält die Embeddings der Eingabeworte und deren Positionen als Eingabe-Embeddings. Jeder Encoder hat zwei Operatoren. Der erste Operator ist eine Selbstattentio, wie oben beschrieben (Abb. 6.48). Sie wird aber nicht nur einmal, sondern achtmal parallel in der gleichen Schicht berechnet (Multi-Head Attention). Der value-Vektor hat dabei die Länge 64, also auch der erzeugte Embedding-Vektor. Die acht Ergebnisse werden aneinandergehängt und ergeben wieder einen Vektor der Länge 512.

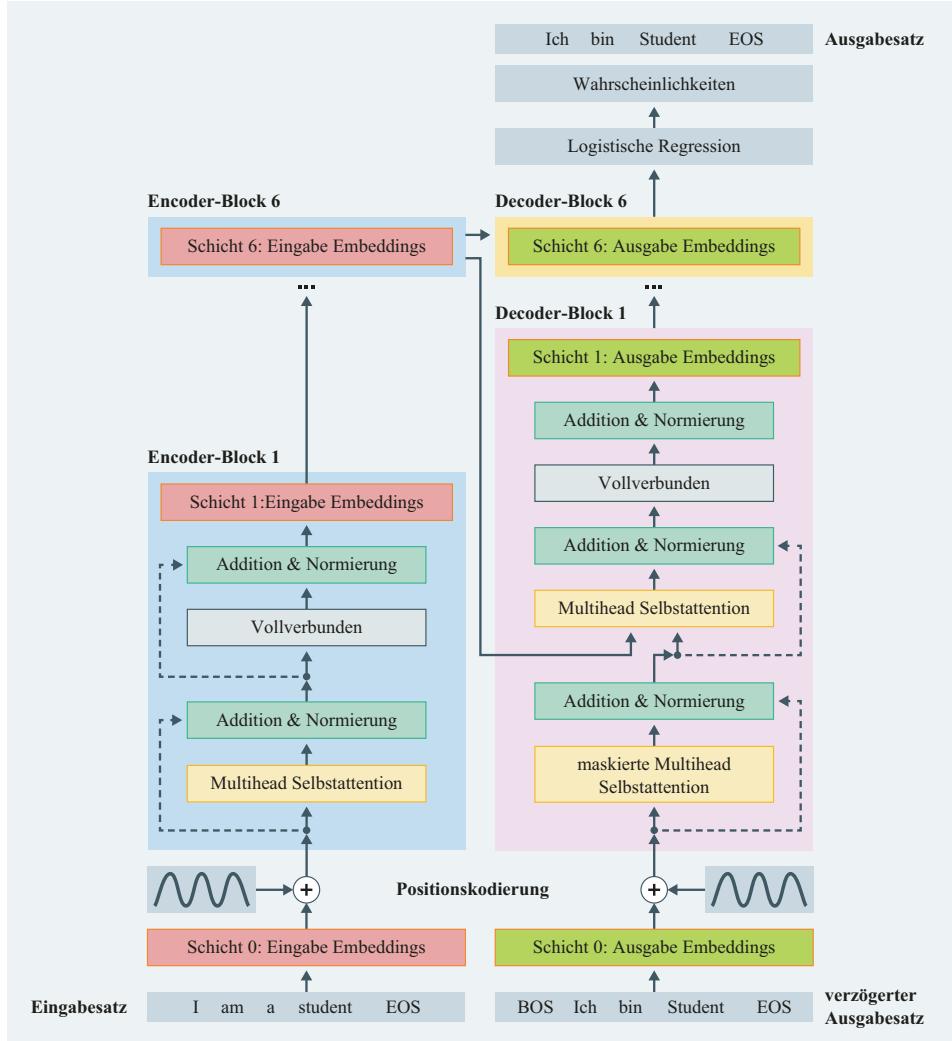
Der zweite Operator ist eine vollverbundene Schicht mit abgeschnittener linearer ReLU-Aktivierung, welche die Embeddings für jedes Wort getrennt transformiert. Ergebnis eines Transformer-Blocks ist ein neues Embedding für jedes Wort der Eingabesequenz.

Um die Schichten herum wird, wie bei ResNet (Abschn. 5.5.2), eine residuale Verbindung (Bypass) verwendet, die die Eingaben einfach nach oben kopiert. Hierdurch wird die Optimierung wesentlich vereinfacht. Nach jedem Operator normalisiert eine Schicht den Mittelwert und die Varianz ähnlich zur Batch-Normalisierung (Abschn. 4.6.4). Die erzeugten Embeddings des Encoders werden von dem Encoder der nächsten Schicht weiterverarbeitet.

*Decoder:* Der Decoder verwendet die von der letzten Schicht des Encoders erzeugten Eingabe-Embeddings, um die Wahrscheinlichkeiten der Ausgabe-Worte in der Zielsprache sukzessive zu berechnen. Um die Ausgabeworte „Ich bin Student EOS“ zu generieren, muss also der Encoder (linke Seite von Abb. 6.50) einmal durchlaufen werden und der Decoder (rechte Seite von Abb. 6.50) viermal, um nacheinander die Wahrscheinlichkeiten der Worte „Ich“, „bin“, „Student“ und „EOS“ zu berechnen.

Der Decoder der untersten Schicht erhält die Ausgabe-Embeddings der vorherigen Wörter des Zielsatzes und Embeddings von deren Positionen als Eingabe. Wie beim Sequence-to-Sequence-Modell sind die Wörter um eine Position verzögert.

Der Decoder enthält drei Operatoren. Der erste Operator ist eine achtfache Selbstattentio (Abb. 6.48) auf den Ausgabe-Embeddings. Durch eine Maske wird hierbei sichergestellt, dass nur vergangene (schon prognostizierte) Worte und deren Embeddings berücksichtigt werden. Durch diesen Trick wird das Modell autoregressiv und kausal, d. h. die Wahrscheinlichkeit für das Ausgabewort hängt nicht von den „zukünftigen“ Ausgabeworten ab. Ein zweiter Operator berechnet eine achtfache Kreuzattentio zu den Eingabe-Embeddings des Encoders der obersten Schicht (Abb. 6.49). Sie läuft ähnlich ab wie die Selbstattentio in Abb. 6.48, nur dass die Query aus den Decoder-Embeddings berechnet wird und Key und Value aus den Embeddings des obersten Transformer-Blocks des En-



**Abb. 6.50** Ein Transformer zur Übersetzung von einer Sprache in eine andere besteht aus sechs Encoder-Blöcken (links) und sechs Decoder-Blöcken (rechts). Der Encoder berechnet eine Selbstattentien und erzeugt in jeder Schicht ein neues Embedding (rosa) der Eingabeworte. Der Decoder berechnet nacheinander für die einzelnen Ausgabeworte in jeder Schicht ein Embedding (hellgrün). Dazu nutzt er eine maskierte Selbstattentien der Ausgabeworte und eine Kreuzattentien zu den Encoder-Embeddings der letzten Schicht. Aus dem Ausgabe-Embedding der letzten Schicht berechnet ein logistisches Regressionsmodell die Wahrscheinlichkeiten für das Ausgabewort. Sowohl im Encoder als auch Decoder gibt es residuale Verbindungen (gestrichelt) und vollverbundene Schichten

coders. Ergebnis ist ein modifiziertes Ausgabe-Embedding der Worte in der Zielsprache. Dadurch hat sie große Ähnlichkeit zu der ursprünglichen Attention eines RNN in Abb. 6.41.

Der dritte Operator ist eine vollverbundene Schicht mit abgeschnittener linearer ReLU-Aktivierung, welche die Embeddings für jedes Wort transformiert. Ergebnis jeder Schicht ist ein neues Embedding für die vorherigen Worte der Ausgabesequenz. Um die Schichten herum wird wieder ein Bypass verwendet. Nach jedem Operator findet wieder eine Normalisierung des Mittelwertes und der Varianz statt.

Aus den berechneten Ausgabe-Embeddings der letzten (sechsten) Transformer-Schicht werden durch eine logistische Regression jeweils die Wahrscheinlichkeiten des nächsten Ausgabewortes berechnet.

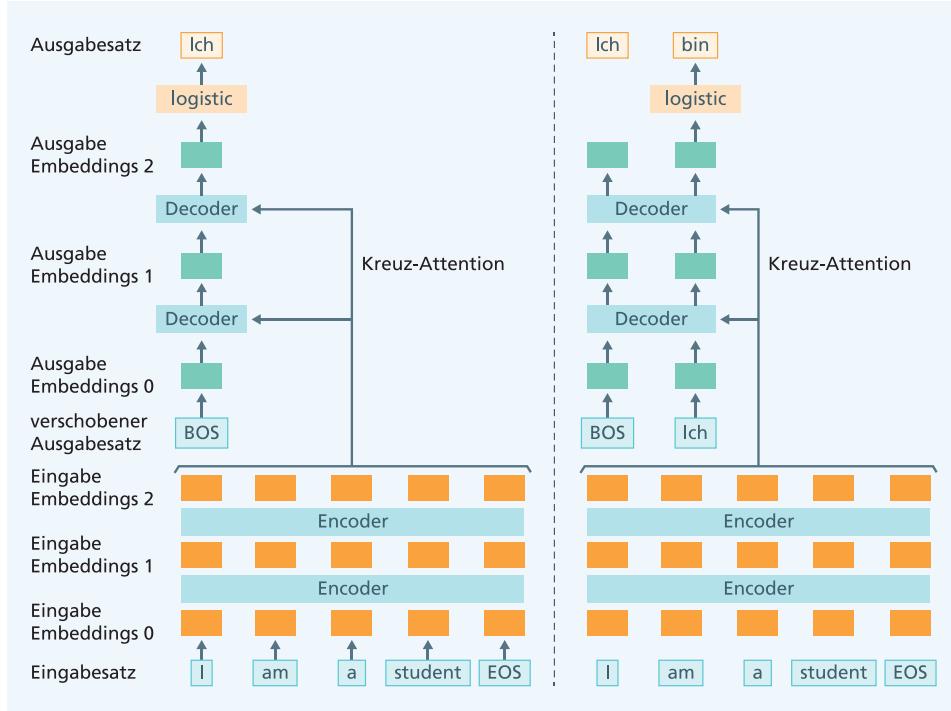
#### 6.6.4 Training des Transformers für die Sprachübersetzung

Es ist instruktiv, sich den Informationsfluss beim Training des Transformers mit jeweils 2 Encoder- und Decoder-Schichten zu verdeutlichen (Abb. 6.51). Wie immer werden die Parameter zu Beginn auf Zufallswerte gesetzt. Für den Vorwärtsschritt werden zunächst für den Eingabesatz „I am a student EOS“ Wort-für-Wort Embeddings ausgewählt, wodurch sich nach Addition der Positions-Embeddings die Eingabe-Embeddings 0 ergeben. Durch Anwendung der Encoder werden anschließend die Eingabe-Embeddings 1 und Eingabe-Embeddings 2 berechnet.

Die Erzeugung der Ausgabeworte erfolgt dann sukzessive für die einzelnen Positionen. Als Eingabe dient der verzögerte Ausgabesatz mit allen Wörtern vor dem Wort, welches prognostiziert werden soll. Für das erste Ausgabewort (Abb. 6.51 links) ist dies das Satzanfangssymbol „BOS“ (Begin of Sentence). Für diese Eingabe wird durch Lookup das entsprechende Ausgabe-Embedding 0 ausgewählt und dieses durch die Decoder-Blöcke propa-giert und Ausgabe-Embedding 1 und Ausgabe-Embedding 2 berechnet. Dabei werden durch Kreuz-Attention Informationen aus der letzten Embedding-Schicht des Encoders, Eingabe-Embeddings 2, übernommen. Schließlich werden durch das logistische Regressionsmodell die Wahrscheinlichkeiten des ersten Ausgabewortes (beobachtet: „Ich“) berechnet.

Wie meist bei der Prognose mit der Softmax-Funktion wird auch hier wieder die Log-Likelihood Verlustfunktion (Abschn. 3.5.2) verwendet. Zur Gradientenberechnung wird die in (Abb. 6.51 links) dargestellte Abfolge in umgekehrter Richtung durchlaufen. Das Fehlersignal wird vom logistischen Regressionsmodell generiert, welches die Wahrscheinlichkeit des ersten Wortes (beobachtet: „Ich“) bestimmt. Anschließend durchläuft es nacheinander von oben nach unten die Decoderblöcke (Abb. 6.50 rechts) und liefert Gradienten für deren Parameter. Wichtig ist, dass sich der Informationsfluss in die achtfache Attention aufspaltet und die jeweiligen Matrizen  $K$ ,  $V$  und  $Q$  modifiziert werden. Über die Kreuz-Attention gelangt das Fehlersignal zu der obersten Schicht des Encoders.

Anschließend wird das Fehlersignal in gleicher Weise durch die Encoder-Blöcke von oben nach unten propa-giert und liefert Gradienten für deren Parameter. Zum Schluss werden Gradienten für die Ausgabe- und Eingabe-Embeddings bestimmt.



**Abb. 6.51** Erzeugung der Worte der Übersetzung durch den Transformer. Zunächst werden durch die Encoder-Blöcke die Eingabe-Embeddings (orange) der obersten (zweiten) Schicht generiert. Zur Bestimmung der Wahrscheinlichkeiten des ersten Ausgabewortes (linke Grafik) wird ein Ausgabe-Embedding des Satzanfangssymbol „BOS“ des verschobenen Ausgabesatzes als Eingabe verwendet. Danach werden durch zwei Decoder-Blöcke und das logistische Regressionsmodell die Wahrscheinlichkeiten des ersten Ausgabewortes (beobachtet: „Ich“) erzeugt. Die Informationen aus dem Eingabesatz werden dabei durch Kreuz-Attention mit dem Eingabesatz übertragen. Die Berechnung der Wahrscheinlichkeiten für das zweite Ausgabewort (rechte Grafik, beobachtet: „bin“) wird analog mit dem verschobenen Ausgabesatz „BOS Ich“ als Eingabe durchgeführt

Anschließend werden auf ähnliche Weise die Wahrscheinlichkeiten für das zweite Ausgabewort (beobachtet: „bin“) berechnet (Abb. 6.51 rechts), wodurch sich in analoger Weise zusätzliche Gradientenvektoren ergeben, die zu den ersten Gradienten addiert werden. Analog werden die weiteren Ausgabeworte prognostiziert und entsprechende Gradienten bestimmt. Insgesamt beeinflusst das Fehlersignal auf verzwickte Weise eine große Anzahl von Parametern. Durch die residualen Verbindungen und die Batch-Normalisierung in Abb. 6.50 wird sichergestellt, dass der stochastische Gradientenabstieg für dieses komplexe Modell funktioniert, und die üblichen Schwierigkeiten, wie explodierende und verschwindende Gradienten, unter Kontrolle sind.

### 6.6.5 Byte-Pair Kodierung zur Reduktion des Vokabulars und zur Repräsentation unbekannter Wörter

Wir haben gesehen (Abschn. 6.5.3), dass die Qualität der Übersetzung stark leidet, wenn Wörter nicht im Vokabular sind. Vaswani et al. (2017) verwenden daher das Byte Pair Encoding (BPE), um ein relativ kleines Vokabular aus Worten und Buchstabenfolgen zu erhalten und gleichzeitig unbekannte Wörter durch Teilworte darstellen zu können.

Das BPE-Vokabular wird an Hand der Trainingsdaten gebildet. Zu Beginn werden nur alle Buchstaben, Ziffern und sonstigen Zeichen der Trainingsmenge als Token bezeichnet und ins Vokabular aufgenommen. Jedes Wort besteht zunächst aus isolierten Buchstaben und einem Wortende-Symbol (Abb. 6.52). Man zählt nun alle Paare von Token und verbindet das Paar, welches am häufigsten auftaucht, zu einem neuen Token des Vokabulars. Verbindungen über ein Wortende-Symbol hinaus sind nicht erlaubt. Dies wird solange fortgeführt, bis man eine vorgegebene Anzahl von Token im Vokabular hat. Im Ablauf des Algorithmus werden die häufigsten Worte als Token dargestellt, während seltenerne Worte aus Token zusammengesetzt werden. Hierdurch kann das Vokabular auf 37.000 Token begrenzt werden. Wichtig ist, dass bei einer Anwendung neue und unbekannte Wörter ebenfalls aus Token zusammengesetzt werden können, da ja auch alle Buchstaben im Vokabular enthalten sind.

### 6.6.6 Ergebnisse für das Transformer-modell

Das Transformer-modell wurde von Vaswani et al. (2017) auf zwei Trainingsdatensätzen trainiert: den Englisch-Deutschen WMT 2014-Übersetzungsdaten mit 4,5 Millionen Sätzen und den Englisch-Französischen WMT 2014-Übersetzungsdaten mit 36 Millionen Sätzen.

Alle Netzparameter, d. h. die Matrizen  $Q$ ,  $K$ ,  $V$  der verschiedenen Schichten, die Parameter der vollverbundenen Schichten und die Wort-Embeddings der Eingabe- und Ausga-

Text	Vokabular
eine * rose * ist * eine * rose *	einor*st*
eine * rose * ist * eine * rose *	eeinor*st*
eine * rose * ist * eine * rose *	eeiin*neor*st*
eine * rose * ist * eine * rose *	eeiin*neor*ro*st*
ei ne * ro se * i st * ei ne * ro se *	eeiin*ne*or*ros*set*
ei ne * ro se * i st * ei ne * ro se *	eeiin*ne*ne*or*rosset*
eine* ro se * i st * eine* ro se *	e ei*ne*in*ne*or*rosse*st*

**Abb. 6.52** Byte Pair Encoding. Die erste Zeile zeigt die Anfangssituation. Das Vokabular besteht aus verwendeten Buchstaben und dem Wortende-Symbol „\*“. Es werden in den nächsten Zeilen sukzessive die häufigsten Paare von Token zu einem neuen Token verbunden und das Vokabular entsprechend erweitert

besprache werden in einem Optimierungslauf auf 8 GPUs trainiert. Insgesamt hat das Modell 213 Millionen Parameter. Das Training für das große Modell dauerte 3,5 Tage.

Für den WMT 2014 Datensatz für die Übersetzung von Englisch nach Deutsch erzielte das Transformer-modell den zum Zeitpunkt der Publikation besten Score mit einem BLEU-Wert von 28,4 (Abb. 6.53). Zudem betragen die Trainingskosten nur ein Bruchteil der anderen Modelle.

Für den WMT 2014-Datensatz für die Übersetzung vom Englischen ins Französische war das Transformer-modell mit einem BLEU-Wert von 41,8 ebenfalls besser als alle anderen Modelle. Auch hier betragen die Trainingszeiten nur 25 % der Zeit, die andere Modelle benötigten. Mittlerweile wurde mit einer Erweiterung des Transformer-modells (mit Rückübersetzung) ein BLEU-Score von 45,6 erzielt.

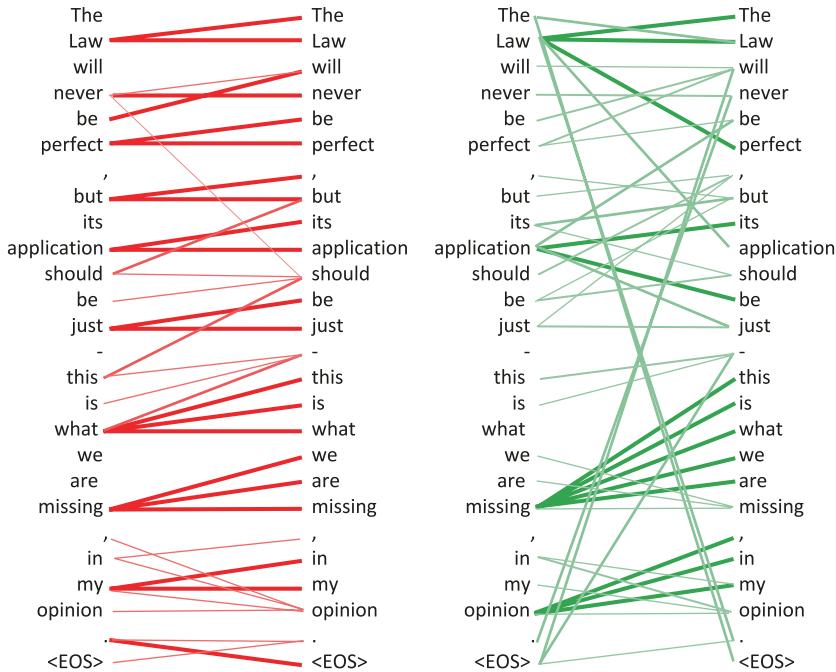
Es ist aufschlussreich, die Ergebnisse der Selbstattentioin zu verdeutlichen. Abb. 6.54 zeigt für einen englischen Satz die Aktivierungen für zwei unterschiedliche Selbstattentioin-Operatoren der fünften Schicht. Es zeigt sich, dass die beiden Operatoren zum Teil unterschiedliche Interpretationsaspekte enthalten. Zum Beispiel ist auf der linken Seite „perfect“ mit „be“ und „perfect“ verbunden, während auf der rechten Seite „be“ und „perfect“ mit „Law“ verknüpft sind. Offenbar hat das Modell gelernt, worauf sich „perfect“ bezieht.

Popel et al., (2018) untersuchten, wie die Optimierung des Transformer-modells verbessert werden kann. Wie Abb. 6.55 zeigt, ist es offenbar besser, viele Trainingsbeispiele anstelle von wenigen zu verwenden. Sie zeigen auch, dass große Modelle mit längeren verdeckten Vektoren und einer höheren Anzahl von Multi-Head-Attention-Modulen pro Schicht sich auszahlen, allerdings auf Kosten eines höheren Rechenzeitbedarfs.

Forscher von Microsoft gaben bekannt (Hassan et al. 2018), dass es mit ihrem maschinellen Übersetzungssystem gelungen sei, mit der gleichen Genauigkeit wie ein Mensch einen Text aus dem Chinesischen ins Englische zu übersetzen. Dabei wurden externe bilinguale Experten konsultiert, um die Übersetzungsqualität zu beurteilen. Dies bedeutet einen Durchbruch bei der maschinellen Übersetzung, den selbst Microsoft nicht so früh erwartet hatte.

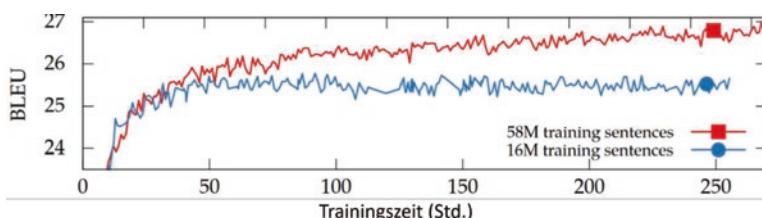
Model	BLEU		FLOPS	
	EN-DE	EN-FR	EN-DE	EN-FR
Deep Att (Zhou et al. 2016)		40,4		$8 * 10^{20}$
ConvS2S (Gehring et al. 2017)	26,4	41,3	$8 * 10^{19}$	$1 * 10^{21}$
Transformer (Vaswani et al. 2018)	28,4	41,8		$2.3 * 10^{19}$

**Abb. 6.53** Resultate für das große Transformer-modell im Vergleich mit dem RNN Attention Modell und dem Convolutional Sequence-to-Sequence Modell. FLOPS sind die Anzahl der Gleitkommaoperationen zum Training. Die Trainingskosten des Transformers betragen nur einen Bruchteil der Alternativmodelle



**Abb. 6.54** Darstellung der Attention-Gewichte auf der 5. Schicht des Transformer-modells. Es handelt sich zwei komplementäre Multi-Head Selbstattentions. Die Höhe der Gewichte wird durch die Stärke der Linien ausgedrückt

Basis für das Microsoft-System war ein Transformer-modell mit zusätzlichen Verbesserungen. Das System verfolgt die Strategie des dualen Lernens: Dabei wird gleichzeitig ein Übersetzungsmodell vom Chinesischen ins Englische und vom Englischen ins Chinesische trainiert. Durch diese Rückübersetzung gewinnt man ein zusätzliches Fehler-signal, da die zweite Übersetzung gleich dem Ausgangssatz sein soll. Damit lassen sich zusätzlich einsprachige Texte zum Training von Übersetzungsmodellen nutzen.



**Abb. 6.55** Das Transformer-Netz mit einem großen Trainingsdatensatz erreicht schneller eine gute Übersetzungsqualität als das mit einer kleineren Trainingsmenge (Popel et al. 2018). Bildnachweis im Anhang



**Abb. 6.56** Bei einer Ausbildungsveranstaltung kann ein simultanes Übersetzungssystem die Sprachbarriere überwinden und Interaktionen erlauben. Bildnachweis im Anhang

### 6.6.7 Simultane Übersetzung erfordert Wartezeiten

In vielen Situationen, z. B. Konferenzen und Lehrveranstaltungen, ist es wichtig, dass die Übersetzung nahezu gleichzeitig mit den Worten des Sprechers verfügbar wird (Abb. 6.56). Nur so können die Zuhörer die Mimik und die Gesten des Sprechers adäquat interpretieren. Diese Simultanübersetzung ist aber bei vielen Sprachpaaren sehr schwierig wegen der unterschiedlichen Wortstellung der Sprachen. Abb. 6.57 zeigt als Beispiel eine deutschen Satz und seine Übersetzung ins Englische. Das Verb „einigen“ im Deutschen Satz erscheint als 12. Wort, während „agreed“ in der englischen Übersetzung als 7. Wort auftritt. Um eine zuverlässige Übersetzung zu liefern, müsste also die Übersetzung mit mindestens 5 Worten Verzögerung einsetzen.

Aufgrund des gleichzeitigen Verstehens und Produzierens in zwei Sprachen ist Simultanübersetzung für den Menschen aufgrund der notwendigen Konzentration und der Belastung der Stimme jedoch äußerst herausfordernd und anstrengend. Die Anzahl qualifizierter Simultandolmetscher ist weltweit sehr begrenzt und jeder kann jeweils nur etwa 15 bis 30 Minuten an einem Stück übersetzen, weil die Fehlerrate bereits nach wenigen Minuten stark ansteigt.

Menschliche Simultandolmetscher starten ihre Übersetzung meist einige Sekunden, nachdem der Sprecher begonnen hat, und sind einige Sekunden nach dem Ende der Sprecheräußerung fertig. Ma et al. (2019) schlagen daher für eine maschinelle Simultanübersetzung vor, auf Grund der bisher verfügbaren Wörter des Eingabesatzes die nächsten Wörter mit einem Sprachmodell zu prognostizieren. Hieraus erhalten sie Hinweise auf fehlende Teile (z. B. Verben) des Eingabesatzes. Ihr „warte- $k$ “ Modell basiert auf dem Transformer und startet die Übersetzung, nachdem  $k$  Wörter des Eingabesatzes verfügbar

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

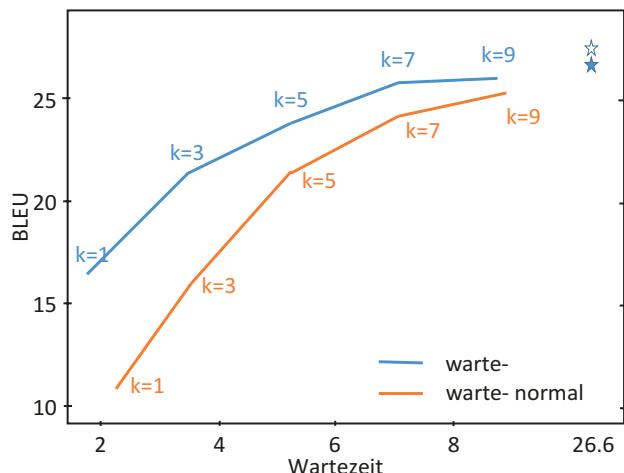
**Abb. 6.57** Deutscher Eingabesatz (erste Zeile), die zugehörige Wort-für-Wort Übersetzung (zweite Zeile) und die englische Übersetzung (dritte Zeile). Die englische Übersetzung startet, nachdem das dritte deutsche Wort „man“ verfügbar ist. Daher muss das System bei der Übersetzung z. B. das Wort „agreed“ vorhersagen, weil „einigen“ erst später ausgesprochen wird

sind. Dazu erhält der Decoder des Transformers zum Zeitpunkt  $k$  Zugriff auf die Embeddings der ersten  $k$  Wörter des Encoders und erzeugt mit dem Decoder ein erstes Wort der Übersetzung. Ist das nächste Wort  $k + 1$  bekannt, so wird der Encoder hiermit aktualisiert und der Decoder erzeugt mit den neuen Embeddings das nächste Ausgabewort usw. Encoder und Decoder werden also laufend aktualisiert. Hierdurch ist der Decoder gezwungen, eine Übersetzung schon aufgrund weniger Wörter zu erzeugen und lernt implizit die Prognose der folgenden Worte des Originalsatzes (Abb. 6.57).

Die Übersetzungsqualität hängt natürlich von der Wartezeit  $k$  ab (Abb. 6.58). Beginnt die Übersetzung nach dem ersten Wort ( $k = 1$ ), so ergibt sich für die Übersetzung von Deutsch nach Englisch auf dem WMT15-Datensatz ein BLEU-Wert von 17. Bei  $k = 3$  erreicht man einen BLEU-Wert von 22, der sich für  $k = 9$  auf 26 steigert. Beginnt man die Übersetzung mit dem normalen Transformer-modell schon nach  $k$  Wörtern (warte- $k$  normal), so ergeben sich viel schlechtere Werte. Die Sterne zeigen den BLEU-Wert, wenn der gesamte Satz verfügbar ist ohne Beam-Suche (gefüllter Stern) und mit Beam-Suche (offener Stern).

Dieses Modell lässt sich kombinieren mit einer Spracherkennung und einem Spracherzeugungssystem, sodass die Übersetzung direkt generiert werden kann. Jia et al. (2019) haben ein Übersetzungssystem vorgestellt, welches direkt von Audio nach Audio übersetzt.

**Abb. 6.58** Übersetzung vom Deutschen ins Englische. Ergebnisse des prognostizierenden Transformermodells (warte- $k$ ) und eines nicht angepassten Transformer-modells (warte- $k$  normal), wenn beide die Übersetzung nach  $k$  Wörten starten



## 6.7 BERT: Sprachmodelle für die Repräsentation von Bedeutungen

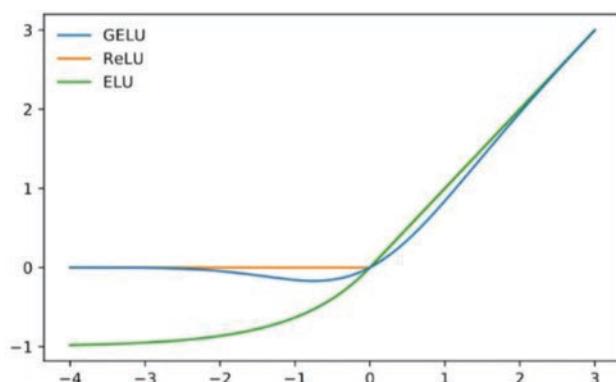
### 6.7.1 BERT-Architektur

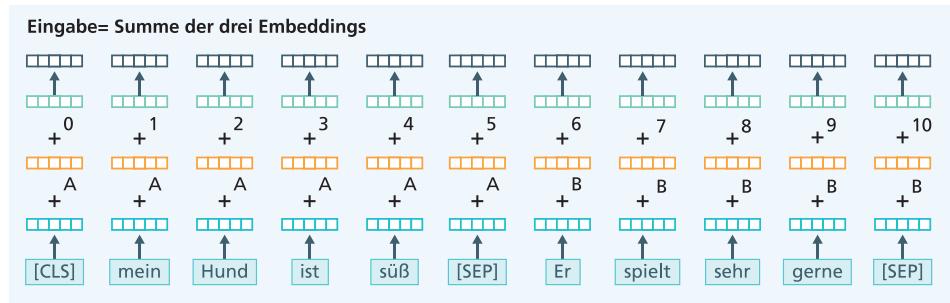
BERT ist eine Abkürzung für **Bidirectional Encoder Representations from Transformer** (Devlin et al. 2018). Es nutzt den Encoder und die Selbstattenton des Transformers (Abb. 6.48), um Embeddings für den Eingabetext zu lernen. Diese können dann in weiteren Anwendungen zur Sprachverarbeitung genutzt werden. Der Vorteil ist, dass mit einem großen, nicht-annotierten Trainingsdatensatz ein allgemeines Embedding-Modell für eine Sprache erzeugt wird. Anschließend können mit relativ wenigen Trainingsdaten spezielle Aufgaben gelöst werden. Ein solches Vorgehen nennt man auch Transferlernen (Abschn. 5.7).

Zunächst muss eine Art der Wortkodierung ausgewählt werden. Es wird eine Zerlegung in Teilworte ähnlich dem Byte-Pair Encoding (Abschn. 6.6.5) eingesetzt. Zudem werden trainierte Positions-Embeddings anstelle der trigonometrischen Funktionen (Abb. 6.47) verwendet, um die Wortposition zu kodieren. Die Selbstattenton wird mithilfe einer GELU-Aktivierungsfunktion (Abb. 6.59) berechnet, die bessere Ergebnisse zur Folge hat (Hendrycks und Gimpel 2016).

Die Eingabe besteht aus zwei Sätzen, wobei der erste Satz mit der Markierung [CLS] beginnt und mit [SEP] endet. Der zweite Satz wird angehängt und endet ebenfalls mit der Markierung [SEP]. Dieser zweite Satz ist in der Hälfte der Fälle tatsächlich der Folgetext des ersten Satzes und in der anderen Hälfte der Fälle ein zufällig ausgewählter Satz. Die gemeinsame Länge der Sätze ist auf 512 Wörter (Token) beschränkt. Es wird zusätzlich ein trainierbares Embedding verwendet, um den Indikator für den ersten und zweiten Satz abzubilden. Insgesamt ist das Eingabe-Embedding die Summe der Byte-Pair-Embeddings,

**Abb. 6.59** Die GELU-Aktivierungsfunktion zusammen mit der ReLU-Aktivierung und der ELU-Aktivierung





**Abb. 6.60** BERT-Eingabe-Embeddings: Es werden das Wort-Embedding (unten), das Textindikator-Embedding (Mitte) für A=erster Text und B=zweiter Text und ein Positions-Embedding (oben) (Abb. 6.47) addiert. Die Summe bildet die BERT-Eingabe

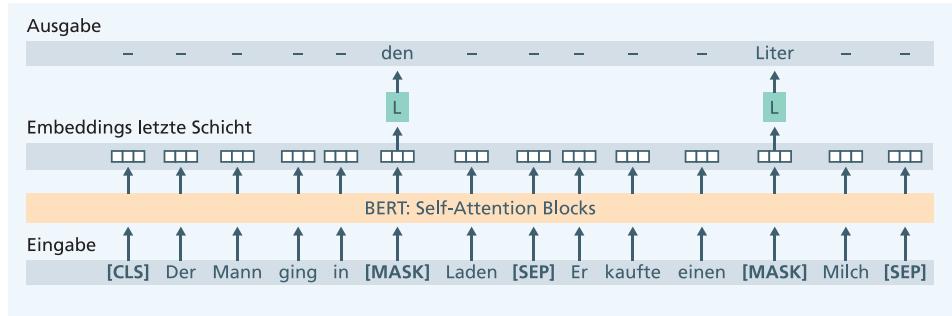
des Positions-Embeddings und des Indikator-Embeddings. Abb. 6.60 zeigt ein Beispiel. BERT verwendet als Modell den Encoder des Transformers mit einer Anzahl von Selbst-attention-Blöcken (Abb. 6.50).

## 6.7.2 BERT-Prognoseaufgaben zum unüberwachten Vortraining

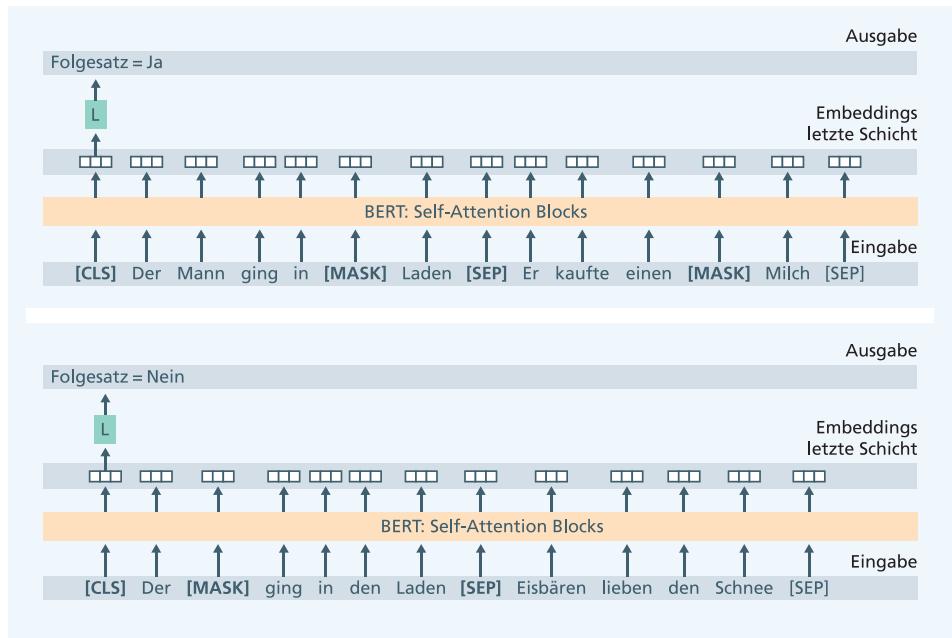
Selbstattention benötigt eine Prognoseaufgabe, um die Attention-Parameter bestimmen zu können. Diese sollte eine sehr detaillierte Kenntnis der Bedeutung von Wörtern und Sätzen erfordern, um Embeddings mit möglichst tiefgehenden Informationen erforderlich zu machen. BERT nutzt gleich zwei Prognoseaufgaben. Vor dem Training wird jedem Satz mit der Wahrscheinlichkeit 0.5 nach einem Trennzeichen [SEP] der echte Folgesatz angefügt, oder sonst ein zufällig ausgewählter anderer Satz.

Die erste Prognoseaufgabe heißt maskiertes Sprachmodell (Devlin et al. 2018). Hierzu werden 15 % zufällig ausgewählte Wörter in einem Satz als Prognosewörter ausgewählt. 80 % der Prognosewörter werden durch das spezielle Wort [MASK] ersetzt, 10 % der Prognosewörter werden durch ein zufälliges anderes Wort ersetzt und 10 % der Prognosewörter werden beibehalten. Die Aufgabe ist nun, an den jeweiligen Positionen mithilfe einer logistischen Regression die Wahrscheinlichkeit der Prognosewörter zu prognostizieren. Ein Beispiel findet sich in Abb. 6.61. Hierbei wird das Embedding an der Position von [MASK] in der letzten Schicht des Encoders als Eingabe für das logistische Regressionsmodell verwendet. Damit ist der Encoder gezwungen, aus der Umgebung des Prognosewortes alle verfügbaren Informationen über das Prognosewort zu sammeln und in das Embedding an der [MASK]-Position zu kodieren.

Die zweite Prognoseaufgabe ist die Klassifikation des nächsten Satzes als echter Folgesatz (IstNächster) oder zufällig ausgewählter Satz (KeinNächster). Hierbei wird das für das [CLS]-Token berechnete Embedding genutzt, um die Wahrscheinlichkeit von Ist-Nächster mit einer logistischen Regression zu prognostizieren. In Abb. 6.62 sind zwei Beispiele wiedergegeben.



**Abb. 6.61** Prognoseaufgabe zum unüberwachten Vortraining: Das maskierte Sprachmodell. Das logistische Regressionsmodell muss die Wahrscheinlichkeit der mit MASK verdeckten Worte vorhersagen. Dabei verwendet es die Embeddings an der MASK-Position der letzten Schicht als Eingabe



**Abb. 6.62** Zwei Beispiele zum unüberwachten Vortraining zur Klassifikation des nächsten Satzes. Das logistische Regressionsmodell **L** muss die Wahrscheinlichkeit prognostizieren, ob der Satz nach dem ersten [SEP] der Folgesatz des Satzes davor ist. Dabei wird das Embedding von [CLS] der letzten Schicht als Eingabe für **L** verwendet

Im Gegensatz zum LSTM-Sprachmodell hat BERT entscheidende Vorteile:

- Das Modell kann bei der Prognose des Wortes Informationen von davor- und dahinterstehenden Wörtern verwenden.

- Die Beziehungen zu weiter entfernt liegenden Wörter werden direkt modelliert und müssen nicht über eine wiederholte Anwendung des rekurrenten Modells prognostiziert werden.
- Das Modell ist gezwungen, Informationen über mögliche Folgetexte als Ganzes zusammenzustellen.

Das Modell wurde auf dem BookCorpus (Zhu et al. 2015) mit Texten aus Büchern und der Wikipedia mit insgesamt 3,3 Milliarden Wörtern über 40 Epochen trainiert. Das „große“ Modell hat 24 Transformer Blocks, verdeckte Vektoren der Länge 1024 und 12 parallele Selbstattention-Module. Insgesamt hat es 345 Millionen Parameter und benötigte bei 16 TPUs (sehr leistungsfähige GPU-Prozessoren) vier Tage Trainingszeit.

---

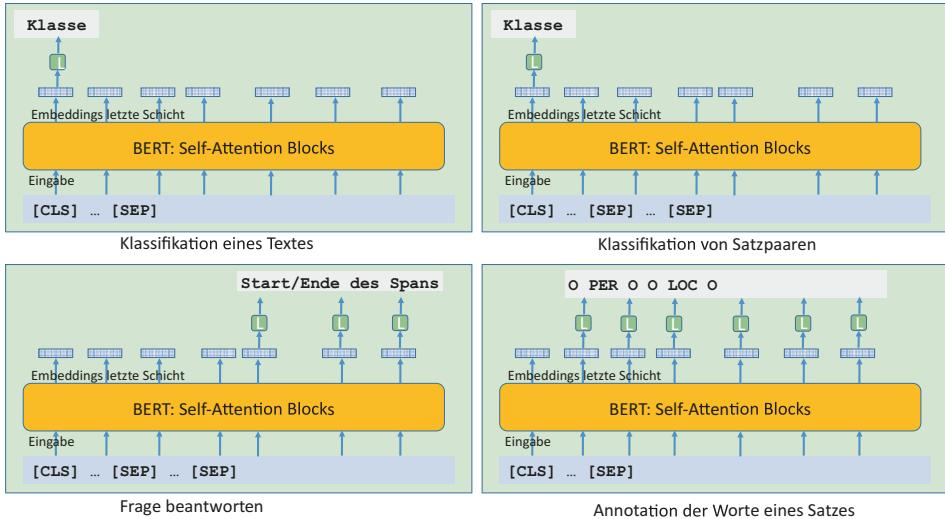
## 6.8 Transferlernen mit umfangreichen BERT-Modellen der Sprache

Mithilfe von Transferlernen kann man Modelle für Aufgaben trainieren, für die eigentlich zu wenige Trainingsdaten vorhanden sind (Abschn. 5.7). Hierzu trainiert man das Modell zunächst für ein ähnliches Problem, für welches sehr umfangreiche Trainingsdaten verfügbar sind. Dabei lernt das Modell grundlegende Zusammenhänge über die Domäne, z. B. über Syntax und Semantik von Sprache. Anschließend findet ein kurzes Feintuning auf den eigentlichen Trainingsdaten statt, bei dem das Modell an die spezielle Aufgabe angepasst wird. Transferlernen wurde schon in den 90er-Jahren diskutiert (Thrun 1998), setzt sich aber erst jetzt wegen vielfach vergrößerter Rechnerleistung durch. In diesem Abschnitt wird Transferlernen für das unüberwacht trainierte BERT-Modell und Transformer-Übersetzungsmodelle diskutiert.

### 6.8.1 Semantische Klassifikationsaufgaben

Das vortrainierte BERT-Model (Abschn. 6.7) wurde für eine Reihe von Klassifikationsaufgaben genutzt, für die ein detailliertes Verständnis der Sprache notwendig ist. Hierbei wurde das Embedding von [CLS] als Eingabe eines logistischen Regressionsmodells verwendet, welches die Wahrscheinlichkeit der jeweiligen Klassen prognostiziert (Abb. 6.63 oben). Alle Parameter von BERT und dem logistischen Regressionsmodell wurden angepasst, um die Klassifikationswahrscheinlichkeit zu optimieren. Das Training für dieses Feintuning dauerte drei oder vier Epochen (Devlin et al. 2018).

In Abb. 6.64 ist eine typische Aufgabe dargestellt. Es werden zwei Texte angegeben und es ist zu entscheiden, ob die Aussage des zweiten Textes eine Folge des ersten Textes ist, einen Widerspruch zum ersten Text bildet oder neutral ist. Die Texte stammen aus unterschiedlichen Quellen.



**Abb. 6.63** Es gibt unterschiedliche Möglichkeiten, BERT für semantische Aufgaben zu nutzen. Bei der Klassifikation eines Textes wird die Klassenwahrscheinlichkeit für das CLS-Token berechnet (oben links). Genauso verfährt man bei der Klassifikation von Satzpaaren (oben rechts). Bei der Beantwortung einer Frage wird Start und Ende der Antwort im Text markiert (unten links). Bei der Annotation der Worte eines Satzes wird für jedes Wort ein Klassenlabel ausgegeben

In Abb. 6.65 sind die Ergebnisse von fünf solchen semantischen Tests zusammengefasst. Die zweite Spalte der Tabelle liefert eine kurze Beschreibung der Aufgabe. In der 3. Spalte wird die von BERT erzielte Genauigkeit auf der Testmenge aufgeführt, in der letzten Zeile das beste andere Ergebnis. BERT übertrifft in allen Fällen die Ergebnisse der anderen Systeme und kann zum Teil beträchtliche Verbesserungen des State of the Art erreichen.

Erster Text	Zweiter Text	Label
Der Alte tröstete Ca'daan immer, außer heute.	Ca'daan kannte den Alten sehr gut.	neutral
ja jetzt weißt du ob wenn jeder wie im August wenn alle im Urlaub sind oder sowas wir uns ein bisschen lässiger anziehen können oder	Der August ist ein schlechter Monat für den Urlaub in der Firma.	Widerspruch
Am anderen Ende der Pennsylvania Avenue begannen die Leute, sich für eine Tour durch das Weiße Haus aufzustellen.	Die Menschen bildeten am Ende der Pennsylvania Avenue eine Schlange.	Folge

**Abb. 6.64** Die Multi-Genre Natural Language Inference (MultiNLI) besteht aus 433k Sätzen, die aufeinander bezogen sind. Sie können neutral sein, einen Widerspruch bilden oder eine Folgerung ausdrücken. Die Texte entstammen unterschiedlichen Quellen, z. B. Roman (oben), Telefongespräch (Mitte) und Reportage (unten)

Name	Aufgabe	BERT	2-ter
MNLI	Zwei Sätze: ist der zweite eine Folge, ein Widerspruch oder neutral?	86,7	82,1
QQP	Zwei Sätze: sind sie semantisch äquivalent?	72,1	70,3
QNLI	Zwei Sätze: ist der 1. die Frage und der 2. die zugehörige Antwort?	91,1	88,1
CoLA	Ist ein Satz „sprachlich akzeptabel“?	60,5	45,4
MPRC	Zwei Sätze: sind sie semantisch äquivalent?	89,3	86,0

**Abb. 6.65** Die Performanz von BERT auf unterschiedlichen semantischen Klassifikationsaufgaben

BERT wurde auch auf den CoNLL2003 Benchmark zur Erkennung von Eigennamen (Abb. 6.31) angewendet. Dabei wurde jedem Wort ein Namenslabel zugeordnet (Abb. 6.63 unten rechts). Es konnte auf der Testmenge einen F1-Wert von 92,4 verbuchen, was eine Verbesserung des State of the Art-Wertes von 0,2 bedeutet.

Sentimentanalyse und Opinionmining haben die Aufgabe, in Texten ausgedrückte Gefühle und Meinungen zu erkennen und in zwei oder mehr Klassen einzurichten. Die Stanford Sentiment Treebank bewertet ganze Sätze als negativ oder positiv. Ein Beispiel ist der Satz „*This movie was actually neither that funny, nor super witty.*“, der eine negative Meinung ausdrückt, obwohl in dem Satz die Worte *funny* und *witty* vorkommen. BERT war ebenfalls in der Lage, diese Aufgabe besser zu erfüllen als die bisherigen Methoden und erreichte eine Genauigkeit von 94,9 % (Devlin et al. 2018).

### 6.8.2 Die Beantwortung von Fragen

Ein weiterer Benchmark ist SQuAD1.1 (Stanford Question Answering Dataset) (Rajpurkar et al. 2016). Hier wird getestet, ob ein Programm den Inhalt eines Textes „verstanden“ hat und Fragen hierzu beantworten kann. Er besteht aus Passagen von Artikeln der englischen Wikipedia und Fragen hierzu. Die Antworten müssen als Wortfolgen in der Passage gefunden werden (Abb. 6.66). Die Frage und die Passage werden als zwei Texte in das BERT-Modell eingegeben. Das BERT-Modell gibt dann für jedes Eingabewort  $w_i$  einen Embeddingvektor  $u_i$  aus. Während des Feintunings beim Training mit den SQuAD-Trainingsdaten werden Vektoren  $s$  und  $r$  trainiert. Die Wahrscheinlichkeit, dass  $w_i$  das Startwort der Antwort ist, wird dann durch  $p_i = \exp(s' * u_i) / \sum_j \exp(s' * u_j)$  gegeben.  $q_i = \exp(r' * u_i) / \sum_j \exp(r' * u_j)$  ist dann die Wahrscheinlichkeit, dass  $w_i$  das Endwort der Antwort ist (Abb. 6.63 unten links).

Das ursprüngliche BERT-Modell erreichte einen F1-Wert von 91,8 %. Derzeit ist eine Variante von BERT, das XLNet (Yang et al. 2019) das beste Modell und erzielt einen F1-Wert von 95,1 %. Die Performanz menschlicher Experten liegt bei 91,2 %, sodass sowohl BERT als auch XLNet die menschliche Genauigkeit übertrifft. Insgesamt erreicht XLNet eine teilweise wesentlich höhere Genauigkeit als BERT in 20 semantischen Aufgaben.

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.

**Frage**

Which empire was the last one Genghis Khan conquered before he died?

**Antwort**

Western Xia

**Abb. 6.66** Der SQuAD Benchmark besteht aus einer Passage aus einem Wikipedia-Artikel (oben), einer Frage (links) und einer Antwort (rechts), die aus der Passage entnommen wurde (gelb unterlegt). Bildnachweis im Anhang

SQuAD2.0 macht die Aufgabe des Textverständnisses realistischer. Bisher war nämlich sicher, dass die Antwort auch in der Passage zu finden war. Es wurden zu den 100.000 Fragen von SQuAD1.1 weitere 50.000 Fragen formuliert, die den beantwortbaren Fragen ähnlich sind, aber nicht aus der Passage beantwortet werden können. Das Verfahren muss also zunächst entscheiden, ob die Antwort überhaupt in der Passage enthalten ist. Ist dies der Fall, so muss es die Worte der Antwort ausgeben. Bestes Verfahren ist derzeit eine Modifikation von BERT. Es erreicht einen F1-Wert von 92,4 % (Lan et al. 2019). Der F-Wert menschlicher Experten ist 89,5 %. Damit ist das Modell auch auf diesem Benchmark besser als menschliche Leser.

Wichtig ist bei all diesen Ansätzen, dass die Verfahren direkt auf andere Sprachen übertragen werden können: es werden keine Wörterbücher, Grammatiken oder sonstige Lexika verwendet. Mittlerweile konzentriert sich die Forschung auf die Lösung komplexerer Frageprobleme, bei denen die Antwort in großen Textmengen gefunden werden muss und bei denen Weltwissen und logisches Schließen erforderlich sind. Insgesamt hat sich gezeigt, dass unüberwachtes Lernen mittlerweile sehr erfolgreich bei Anwendung in der Sprachverarbeitung ist.

Lee et al. (2019) haben BERT auf einer großen Sammlung medizinischer Fachartikel trainiert. Durch Feintuning auf speziellen Trainingsdaten haben sie Extraktionsverfahren für spezielle Anwendungen bestimmt. Sie erreichten signifikante Verbesserungen für die Extraktion biomedizinischer Entitäten (0,5 % F1-Verbesserung auf 86 %) und biomedizinischer Relationen (3,5 % F1-Verbesserung auf 81,3 %). Schließlich haben sie noch die Beantwortung biomedizinischer Fragen trainiert, z. B. „What does mTOR stand for?“ mit der Antwort „mammalian target of rapamycin“. Hier erreichten sie eine absolute F1-Verbesserung von 9,6 %.

**Abb. 6.67** Weltwissen umfasst grundlegende, allgemein bekannte Fakten wie „Wasser löscht Feuer“, hier bei der Waldbrandbekämpfung in Kalifornien. Bildnachweis im Anhang



### 6.8.3 Extraktion von Weltwissen

In der KI besteht Weltwissen aus Fakten über das tägliche Leben, wie etwa „Zwiebeln sind scharf“ und „Wasser löscht Feuer“ (Abb. 6.67). Die meisten Menschen gehen davon aus, dass jeder Mensch diese Fakten kennt. Um sinnvoll agieren zu können, muss auch ein Agent der Künstlichen Intelligenz auf derartiges Weltwissen Zugriff haben.

Bisherige Ansätze in der KI haben versucht, Weltwissen manuell zu kodieren. Ein berühmtes Beispiel ist Cyc, eine seit 1984 kontinuierlich entwickelte Wissensdatenbank (Lenat 1995). Sie formuliert Weltwissen als Fakten und Regeln, und erlaubt es, daraus logische Schlüsse abzuleiten. Derartige Ansätze haben den großen Vorteil, dass logisch wahre Schlussfolgerungen hergeleitet werden können. Der Nachteil gegenüber tiefen neuronalen Netzen ist, dass nicht automatisch neue Zusammenhänge aus Daten gelernt werden können, sondern explizit manuell eingegeben werden müssen.

Das BERT-Modell bietet erstmals einen Weg, Weltwissen teilweise unüberwacht aus einer großen Textkollektion zu extrahieren. Hierzu kann man z. B. SWAG (Situations With adversarial Generations) verwenden, einen Datensatz zum Testen von Weltwissen (Zellers et al. 2018). Er besteht aus 113.000 Fragen zu einer vorgegebenen Situation mit jeweils vier möglichen Antworten. Abb. 6.68 zeigt zwei solche Fragen und die möglichen Antworten.

<p>On stage, a woman takes a seat at the piano. She</p> <ul style="list-style-type: none"> <li>a) sits on a bench as her sister plays with the doll.</li> <li>b) smiles with someone as the music plays.</li> <li>c) is in the crowd, watching the dancers.</li> <li>d) nervously sets her fingers on the keys.</li> </ul>	<p>The woman is now blow drying the dog. The dog ...</p> <ul style="list-style-type: none"> <li>a) is placed in the kennel next to a woman's feet.</li> <li>b) washes her face with the shampoo.</li> <li>c) walks into frame and walks towards the dog.</li> <li>d) tried to cut her face, so she is trying to do something very close to her</li> </ul>
--	---

**Abb. 6.68** Zwei Aufgaben aus dem SWAG Datensatz. Die korrekte Antwort muss ausgewählt werden. Die Antwort auf die erste Frage ist d) und a) auf die zweite Frage

ten. Bei einer Reihe von bisherigen Datensätzen dieser Art musste man hinterher feststellen, dass es unbeabsichtigte Wortmuster in den möglichen Antworten gab, welche auf die korrekte Antwort hinwiesen. Aus diesem Grunde wählten die Autoren ein kompliziertes Verfahren, um solche Effekte auszuschließen. Insgesamt wurden mit fortgeschrittenen Sprachmodellen vier mögliche Antworten konstruiert, welche gut zu der Situation passten. Eine der Antworten traf auf die Situation zu, die anderen stimmten mit Sicherheit nicht.

Devlin et al. (2018) haben diese Aufgabe mit dem vortrainierten bidirektionalen Transformer BERT (Abschn. 6.7) bearbeitet. Das Modell wurde auf dem BookCorpus und der Wikipedia mit insgesamt 3,3 Milliarden Worten über 40 Epochen vortrainiert (Abschn. 6.7.2). Das „große“ Modell hat 24 Transformer Blocks, verdeckte Vektoren der Länge 1024 und 12 parallele Selbstattention-Module (Abschn. 6.6.1). Insgesamt hat es 345 Millionen Parameter und benötigte bei 16 TPUs (sehr leistungsfähige Prozessoren) vier Tage Trainingszeit.

Die SWAG-Daten wurden in eine Trainingsmenge von 73.000, eine Validationsmenge von 20.000 und eine Testmenge von 20.000 Beispielen zerlegt. Auf der Trainingsmenge wurde ein Feintuning des vortrainierten Modells durchgeführt (Devlin et al. 2018). Dabei wurde die Situationsbeschreibung jeweils mit einer Antwort verknüpft, z. B. „[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]“ (Abb. 211). Der Ausgabevektor des BERT-Modells ist dann Eingabe für ein logistisches Regressionsmodell, welches die Wahrscheinlichkeit der vier Alternativen prognostiziert (Abb. 6.63 oben rechts). Dieses Feintuning wurde für drei Epochen durchgeführt. Das Modell muss hierbei etwas über eine Szene wie in Abb. 6.69 aussagen können.

Die Ergebnisse sind in Abb. 6.70 zusammengestellt. Während das beste vorherige Modell auf den Testdaten eine Genauigkeit von 59,2 % hatte, erreichte BERT eine Genauigkeit von 86,3 %. Dies ist 1,3 % besser als die Genauigkeit menschlicher Experten. Insofern erreichte BERT auf den Testdaten für Weltwissen eine höhere Genauigkeit als Fachleute.

**Abb. 6.69** Auf der Bühne nimmt eine Frau am Piano Platz. Die amerikanische Pianistin Robin Spielberg. Diese Szene passt zu einer Aufgabe des SWAG Datensatzes. Bildnachweis im Anhang



System	Genaugkeit (%)
ESIM + ELMo	59,2
BERT-LARGE	86,3
Menschliche Experten	85,0

**Abb. 6.70** Genaugigkeit von verschiedenen Algorithmen auf den Testdaten von SWAG. (Quelle: Devlin et al. 2018)

Insgesamt können TNN in den untersuchten Bereichen für diese Aufgabe sprachliches Weltwissen besser reproduzieren als Menschen.

Wichtig ist, dass BERT auch auf die Informationen aus der großen Trainingsmenge, dem BookCorpus und der Wikipedia, mit insgesamt 3,3 Milliarden Worten, zurückgreifen kann. Diese enthält wesentlich mehr Informationen, als die relativ kleine Trainingsmenge mit 73.000 Frage-Antwort-Beispielen, welche beim Feintuning genutzt wurden. Damit ist es mithilfe des Transferlernens möglich, auch unüberwacht Wissen aus großen Textsammlungen zu extrahieren und abfragbar zu machen.

Petroni et al. (2019) zeigen, dass BERT Fakten und Relationen (z. B. Geburtsort einer Person) genauso gut und teilweise besser als bisherige überwachte Lernverfahren extrahieren kann. Allerdings gilt dies nicht für alle Relationen, in welchen gleichzeitig mehrere Objekte involviert sind ( $N$ -nach- $M$  Relationen). Eine Anwendung des BERT-Ansatzes findet sich in dem Aristo-System (Clark et al. 2019). Aufgabe dieses Systems ist die Beantwortung von Multiple-Choice-Fragen aus dem „Grade 8 New York Regents Science Exam“ für Schüler der 8. Klasse (Abb. 2.10). Das System nutzt drei Arten von Subsystemen: für Information Retrieval (d. h. intelligente Suche in Dokumenten), für das logische Schließen, und Sprachmodelle zur Prognose fehlender Worte und Sätze.

Die Suche in Dokumenten findet einerseits über klassische Suchmaschinen statt, andererseits aber auch gemäß der Übereinstimmung von Worten und n-Grammen. Die Subsysteme für logisches Schließen nutzen unter anderem Wissensgraphen, welche relevante Phrasen durch inhaltliche Relationen verbinden. Die Sprachmodelle nutzen AristoBERT, eine Variante von BERT, um Fragen analog zu dem Schema von Abb. 6.62 zu beantworten. Dieses wird auch mit Feintuning für die Aufgabe trainiert. Jedes Modul gibt einen Punktwert für Konfidenz aus, der durch ein logistisches Modell kalibriert wird. Diese Punktwerte werden dann bei der Auswahl der Antwort verwendet.

Es zeigt sich, dass die BERT-Module die höchste Genaugigkeit bei der Beantwortung erzielen und die anderen Module nur marginale Beiträge leisten. Insgesamt kann das System 91,6 % der Fragen der achten Klasse korrekt beantworten und 83,5 % der Fragen der zwölften Klasse (Clark et al. 2019). Obwohl Aristo nur Multiple-Choice-Fragen ohne Diagramme beantwortet und nur den Bereich der Naturwissenschaft erfasst, stellt es einen wichtigen Meilenstein für Systeme dar, die Inhalte erfassen können.

Für viele Jahre galt das Denken in der symbolischen KI als diskrete, symbolische Manipulation von Aussagen, die in einer formalen Sprache ausgedrückt wurden. Mit dem Aufkommen des tiefen Lernens hat sich diese Metapher geändert. Algorithmen können intellektuell fordernde Aufgaben auf „assoziative“ Weise mit TNN anstatt mit expliziten

Darstellungssprachen lösen. Dies deutet darauf hin, dass die Maschine tatsächlich etwas über Sprache und die Welt „gelernt“ hat und Mechanismen besitzt, mit denen sie dieses Wissen kombinieren und verwenden kann.

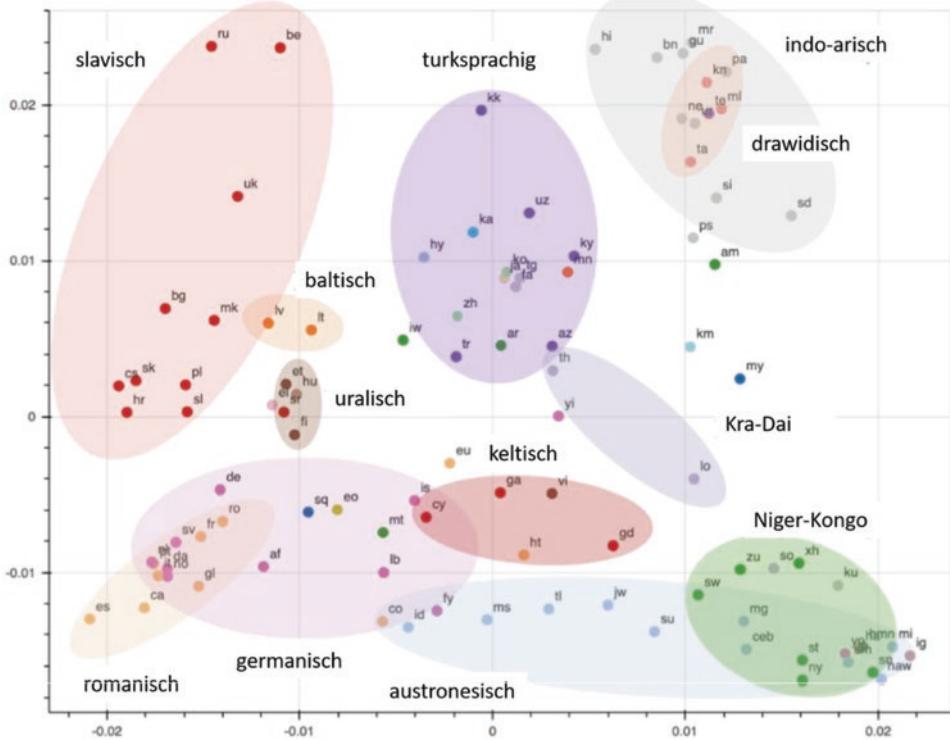
### 6.8.4 Transferlernen für Übersetzungsmodelle

In den letzten Jahren hat die neuronale maschinelle Übersetzung (NMT) sehr große Fortschritte erzielt (vgl. Abschn. 6.6.6). Allerdings werden für gute Übersetzungsleistungen Trainingsmengen von vielen hunderttausend Satzpaaren aus beiden Sprachen benötigt. Nun gibt es auch Sprachpaare, für die nur sehr wenige Trainingsdaten zur Verfügung stehen. Hier kann man potentiell Übersetzungsmodelle verwandter Sprachen nutzen, um die Übersetzungsergebnisse zu verbessern.

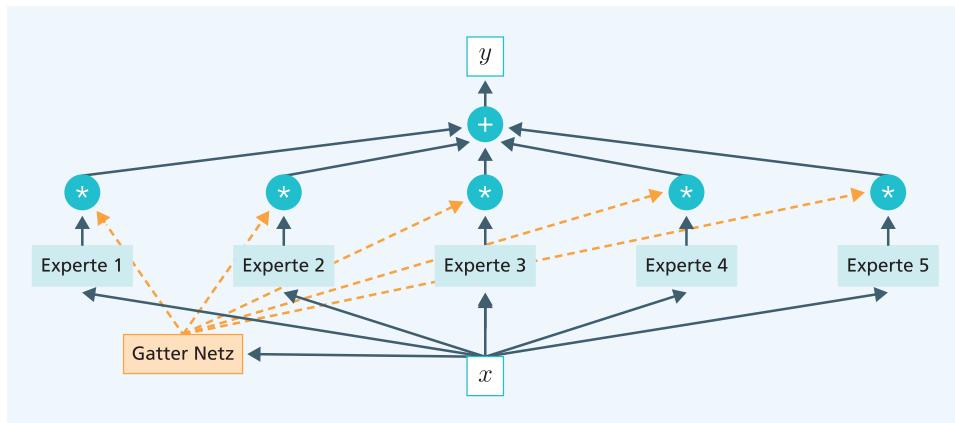
Multilinguale Übersetzungsmodelle nutzen ein einziges Übersetzungsmodell, um viele Sprachen ineinander zu übersetzen (Johnson et al. 2017). Hierbei wird dem Eingabesatz ein Kürzel für die Zielsprache (z. B. „DE“ für Deutsch) vorangestellt. Aharoni et al. (2019) verwenden beispielsweise ein Transformer-modell (Abschn. 6.6) mit sechs Schichten im Encoder und Decoder mit verdeckten Vektoren der Länge 2048, acht Attention-Heads und insgesamt 93 Millionen Parametern. Besonders für Sprachen mit kleinen Trainingsmengen wird der BLEU-Wert durch dieses multilinguale Modell stark verbessert. Beispielsweise hatte das einfache Übersetzungsmodell von Weißrussisch nach Englisch bei einer Trainingsmenge von 4500 Sätzen einen BLEU-Wert von 2,8 und das multilinguale Modell erzielte einen BLEU-Wert von 21,7. Hiermit wird deutlich, dass das multilinguale Modell durch Transferlernen die Übersetzung einzelner ressourcenärmer Sprachpaare stark verbessern kann.

Kudugunta et al. (2019) untersuchen die Embeddings, welche von multilingualen Übersetzungsmodellen für die Worte der einzelnen Sprachen erzeugt werden. Sie analysieren die Embeddings von 103 Sprachen mit einer multidimensionalen Technik (Singular Value Canonical Correlation Analysis), welche die Embeddings in einer Ebene anordnet. Abb. 6.71 zeigt, dass sich die Embeddings in Sprachfamilien gruppieren lassen, die hier durch Ellipsen markiert sind. Dabei kann man aber auch Untergruppen unterscheiden, zum Beispiel die südslawischen und die nordslawischen Sprachen.

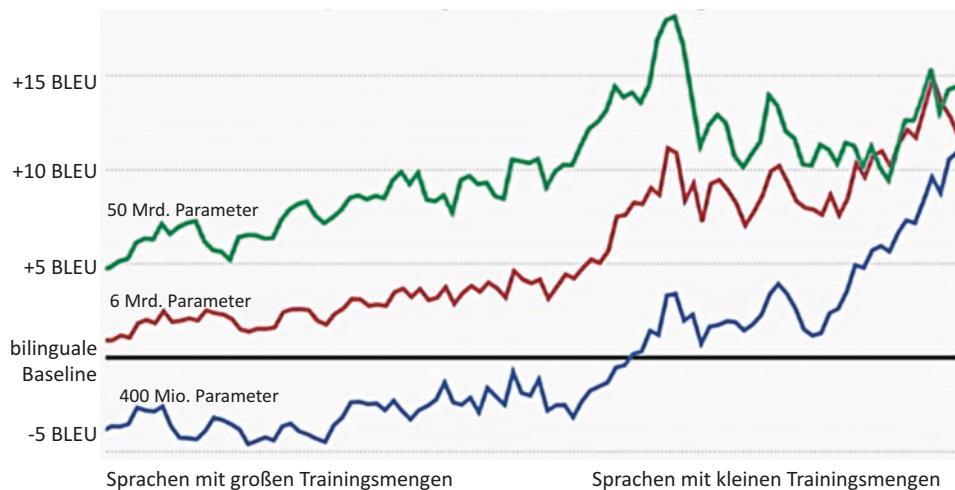
Für Sprachen mit großen Trainingsdatensätzen bringen multilinguale Sprachmodelle normalerweise keine Verbesserung, sondern eine Verschlechterung des BLEU-Werts. Bapna und Firat (2019) stellen ein Modell vor, welches Mixture-of-Experts genannt wird (Shazeer et al. 2017). Es besteht aus  $k$  unterschiedlichen „Experten-Netzen“  $E_1(x), \dots, E_k(x)$ , welche aus einer Eingabe  $x$  jeweils eine Ausgabe  $y$  berechnen (Abb. 6.72). Hinzu kommt ein Gatternetz  $G(x)$ , welches für jedes Netz ein Gewicht erzeugt, mit dem es die Eingabe  $x$  verarbeiten soll. Die Expertennetze können z. B. unterschiedliche Layer in einem Transformermodell sein. Wichtig ist, dass das Gatternetz für jede Eingabe nur wenige Experten auswählt, sodass für die restlichen Experten gar keine Berechnungen angestellt werden müssen. Sowohl die Expertennetze als auch das Gatternetz werden gemeinsam trainiert.



**Abb. 6.71** Im Rahmen eines multilingualen Modells wurden Texte aus 103 Sprachen ins Englische übersetzt. Mithilfe der SVCCA Technik wurden die Embeddings der Wörter der einzelnen Sprachen in einer Ebene dargestellt. Sie lassen sich offensichtlich nach einzelnen Sprachfamilien gruppieren, welche durch Ellipsen dargestellt sind. Nah verwandte Sprachen können bei der Übersetzung stärker voneinander profitieren. Bildnachweis im Anhang



**Abb. 6.72** Mischung von Experten-netzen, welche von einem Gatternetz gesteuert werden. Das Gatternetz entscheidet, welche der Netze eine Eingabe verarbeiten. Nur für diese wenigen Netze muss die Ausgabe berechnet werden. Die Expertennetze können z. B. unterschiedliche Transformer-Schichten sein



**Abb. 6.73** Verbesserung der Übersetzungsqualität im Vergleich zu den 103 bilingualen Resultaten, wenn man die Anzahl der Parameter des multilingualen Modells von 400 Mio. auf 50 Mrd. Parameter erhöht. Es wird deutlich, dass das multilinguale Modell bei genügend großer Kapazität für alle Sprachen eine wesentliche Verbesserung bringt. Bildnachweis im Anhang

Durch diese Konstruktion war es (Bapna und Firat 2019) möglich, die Darstellungsmächtigkeit der Übersetzungmodelle drastisch zu steigern und Modelle mit 50 Milliarden Parametern zu trainieren.

Insgesamt wurden für das Modell mehr als 25 Milliarden Satzpaare aus 103 Sprachen zum Training verwendet. Hierzu war die Nutzung spezialisierter paralleler Rechnerarchitekturen erforderlich. Wie Abb. 6.73 zeigt, ist für derartig umfangreiche Modelle die Übersetzungsqualität eines multilingualen Modells für alle Sprachpaare mindestens 5 BLEU-Punkte höher als bei normalen bilingualen Modellen. Bei Sprachen mit nur wenig Trainingsdaten steigen die BLEU-Werte teilweise um mehr als 15 Punkte. Mit dieser Art von Transferlernen lassen sich die vorhandenen linguistischen Informationen verwandter Sprachen so effizient ausnutzen, dass selbst die ressourcenreichen Sprachen davon profitieren. Allerdings ist der Rechenaufwand extrem hoch.

### 6.8.5 Anwendung von BERT in der Web-Suche

Die Suche nach Webseiten im Internet identifiziert bisher Dokumente, in denen die Worte der Suchanfrage vorkommen. Die Wortreihenfolge und die dadurch ausgedrückte Bedeutung der Suchanfrage spielt dabei keine Rolle. Die Anfrage „2019 brazil traveler to usa need a visa“ resultiert bisher bei Google auch in Treffern der Art „U.S. citizens can travel to Brazil without the red tape of a visa ...“ (Nayak 2019), weil der Suchalgorithmus aus der Frage nicht die Richtung der Reise („to usa“) erkennt (Abb. 6.74). Google gibt nun in seinem Blog bekannt, dass solche Feinheiten von Fragen mithilfe von BERT von der Suchmaschine berücksichtigt werden.

The screenshot shows a search interface with two columns. On the left, under 'Anfrage' (Query), is the text '2019 brazil traveler to usa need a visa'. Below it, under 'Antwort bisher' (Old Answer), is a snippet from the Washington Post dated 2019/03/21, which states: 'U.S. citizens can travel to Brazil without the red tape of a visa... Mar 21, 2019 Starting on June 17, you can go to Brazil without a visa ...'. On the right, under 'Antwort mit BERT-Ranking' (New Answer), are links to 'USEmbassy.gov > br > Visas' and 'Tourism & Visitor | U.S. Embassy & Consulates in Brazil', followed by the text: 'In general, tourists traveling to the United States require valid B-2 visas.' and an ellipsis.

**Abb. 6.74** In einer Webanfrage kann der bisherige Suchalgorithmus nicht die Bedeutung der Konjunktion „to“ richtig zuordnen (links). Dies ist möglich mit dem neuen BERT-Ranking-Verfahren

Im einfachsten Fall erfolgt eine Websuche in drei Schritten. Zunächst wird über den separaten Zeichenketten-Vergleich der Suchbegriffe eine große Zahl (z. B. 1000) möglicherweise relevanter Dokumente zusammengestellt. Anschließend werden diese Dokumente durch einen Ranking-Algorithmus hinsichtlich ihrer Relevanz bewertet. Zum Schluss werden die ersten zehn dieser Dokumente dem Nutzer präsentiert. Der BERT-Algorithmus wird in der Google-Suche beim Ranking eingesetzt. Beim Training von BERT ist die Klassifikation des nächsten Satzes als echter Folgesatz (IstNächster) oder zufällig ausgewählter Satz (KeinNächster) vorgesehen. Man kann daher einfach als ersten Satz die Webanfrage und als Folgesatz das gefundene Dokument verwenden, und die zurückgegebene Wahrscheinlichkeit des „Folgesatzes“ als Ranking-Score nutzen. Dieses Kriterium wird mithilfe des MS-MARCO-Datensatzes für das Ranking von Antworten trainiert. Die Qualität eines solchen Rankingverfahrens wird durch den mittleren reziproken Rang der besten zehn gefundenen Antworten ( $MRR@10$ ) gemessen. Während die besten klassischen Verfahren auf den Testdaten bei einem  $MRR@10$  von 28,1 lagen, erreichen BERT-gestützte Ansätze mittlerweile einen  $MRR@10$ -Wert von 39,3. Dies bedeutet wesentlich bessere Treffer für Websuchen, wobei der Algorithmus allerdings immer noch einzelne Fehler macht (Nayak 2019).

## 6.9 Die Beschreibung von Bildern durch Text

Automatische Bildbeschreibung hat das Ziel, maschinell für ein Bild eine Beschreibung in natürlicher Sprache zu erstellen. Hierbei geht es nicht nur darum, die wichtigsten Objekte in einem Bild aufzuzählen, sondern auch die Relation dieser Objekte und ihre Bedeutung oder Rolle in der dargestellten Szene zu beschreiben. In Abb. 6.75 sieht man auf einem Bild aus dem MS-COCO Datensatz folgende Objekte: Kleinkind, Handy, Stofftier und



- Ein Baby sitzt auf dem Boden und spielt mit einem Handy.
- Ein kleines Mädchen und ein Spielzeugfuchs mit liegender Frau
- Ein kleiner Junge und sein Stofftier spielen mit einem Spiel
- Ein kleines Baby hält ein Smartphone neben einem Stofftier.
- Ein kleines Kind hält ein Telefon in der Nähe eines Stofftiers.

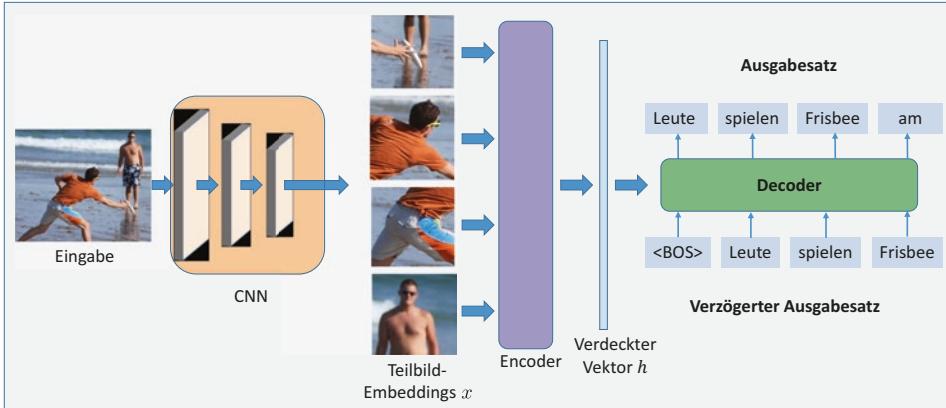
**Abb. 6.75** Bild eines Kleinkindes und zugehörige Beschreibungen (oben) aus dem MS-Coco-Datensatz. Bildnachweis im Anhang

Frau. Ein wichtiger Aspekt der Szene ist, dass das Baby mit dem Handy spielt und nicht etwa eine SMS schreibt. Hierzu wird Alltagswissen benötigt, z. B. was Babys normalerweise tun. Über dem Bild sind fünf alternative Bildunterschriften aufgeführt. Ist ein System in der Lage, zutreffende Bildbeschreibungen zu erstellen, so kann es Menschen mit Sehbehinderungen dabei helfen, Bilder zu erfassen. Darüber hinaus ist eine Bildbeschreibung oft auch Grundlage für die Suche von Bildern in großen Bilddatenbanken.

Für das Training von Bildbeschreibungen wurde bisher meist der MS-COCO Datensatz (Lin et al. 2014) verwendet. Er umfasst 120.000 Bilder, die jeweils mit fünf unterschiedlichen Texten annotiert sind. Sharma et al. (2018) haben einen neuen Datensatz von 3,3 Millionen Bildern mit Beschreibung geschaffen, die Conceptual Captions-Daten. Diese Bilder wurden im Internet gesammelt und mit Text annotiert. Die Bildbeschreibungen wurden in Hinblick auf Informationsgehalt, Flüssigkeit und Lernbarkeit der Resultate ausgesucht. Dabei wurden Eigennamen durch passende Oberbegriffe ersetzt.

Sharma et al. (2018) verwenden zur Erzeugung einer Bildbeschreibung ein System mit drei Komponenten (Abb. 6.76): Ein tiefes CNN (Abschn. 5.5) erhält ein Eingabebild und erzeugt daraus einen Vektor  $x$  mit Bild-Embeddings für jede  $8 \times 8$  Pixelregion. Ein Encoder-Modul transformiert  $x$  in einen verdeckten Vektor  $h$ . Die Aufgabe des Encoders besteht darin, die wichtigsten Eigenschaften der vom CNN extrahierten Merkmale miteinander in Beziehung zu setzen und für die Erzeugung einer Beschreibung aufzubereiten. Ein Decoder erzeugt dann aus  $h$  eine Bildbeschreibung  $z$  als Ausgabe.

Sharma et al. (2018) vergleichen zwei Varianten dieser Struktur. Die erste Variante „L2L“ besteht aus einem LSTM-Encoder und LSTM-Decoder wie bei einem ÜbersetzungsmodeLL für Sprache (Abschn. 6.5.1). Die zweite Variante „T2T“ nutzt ein Transformermodell (Abschn. 6.6) als Encoder und Decoder.



**Abb. 6.76** Blockfunktionsbild eines TNN zur sprachlichen Beschreibung eines Fotos. Zunächst erzeugt ein CNN Embeddings für  $8 \times 8$  Pixelregionen. Anschließend bereitet ein Encoder diese auf und erzeugt einen verdeckten Vektor  $h$ . Zum Schluss generiert ein Decoder daraus den Beschreibungstext. Bildnachweis im Anhang

Abb. 6.77 zeigt Beispielbilder und die erzeugten zugehörigen Bildbeschreibungen. Die Evaluation der Qualität von Bildbeschreibungen mit Textvergleichsmaßen (z. B. BLEU) ist nicht aussagekräftig. Daher werten hier menschliche Betrachter das Ergebnis aus. Die Ergebnisse auf den Flickr-Daten sind in Abb. 6.78 dargestellt. Hier schneidet das Transformer-modell T2T besser ab. Allerdings ist es von den 65,9 % der 1+ Bewertungen noch ein weiter Weg bis zur menschlichen Leistung von 96,0 %. Mittlerweile wurden neue Methoden vorgestellt (Li et al. 2020), welche die Objekte des Bildes und ihre Position detailliert beschreiben und damit bessere Bildunterschriften produzieren.

Es zeigt sich, dass es noch wesentlicher Fortschritte bedarf, bis die Verfahren brauchbar für die Praxis sind. Dazu müssen einerseits aussagekräftigere Informationen über Objekte



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

**Abb. 6.77** Zwei Bilder und die von einem RNN und einem Transformer (T2T) prognostizierten Beschreibungen (Sharma et al. 2018). Bildnachweis im Anhang

Modell	1+	2+	3+
L2L	57,1%	41,8%	27,7%
T2T	65,9%	50,6%	35,5%
Mensch	96,9%	90,3%	78,5%

**Abb. 6.78** Qualität Bildbeschreibungen. Drei Betrachter werden gefragt, welche der Beschreibungen eines Bildes „gut“ sind. Der Anteil der Beschreibungen, die mindestens ein Betrachter gut fand, steht unter 1+. Unter 3+ ist der Anteil der Fälle, bei denen alle drei Betrachter die Beschreibung als gut beurteilten

und deren räumliche und sachliche Relationen aus Bildern extrahiert werden können. Diese müssen dann mit anderweitig erworbenem Weltwissen über plausible Zusammenhänge zwischen Objekten kombiniert werden.

---

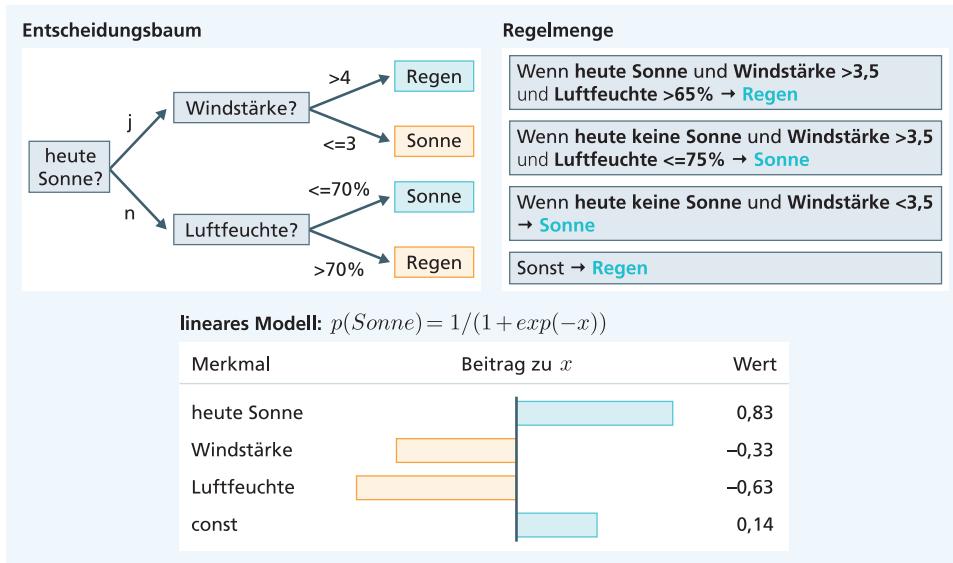
## 6.10 Die Erklärung der Prognosen von TNN

### 6.10.1 Erklärungen sind notwendig

Tiefe neuronale Netze funktionieren in vielen Anwendungsfällen gut. Allerdings ist nicht klar, wie die Ergebnisse im Einzelfall zustande kommen und ob den Resultaten vertraut werden kann, es handelt sich also um Black-Box-Modelle. In der Europäischen Union wurde im Jahre 2018 die Datenschutz-Grundverordnung (DSGVO) in Kraft gesetzt. Ein innovativer Aspekt der DSGVO sind die Klauseln zur automatisierten (algorithmischen) Entscheidungsfindung zu Personen, einschließlich der Erstellung von individuellen Profilen. Hier gibt es in einem gewissen Ausmaß zum ersten Mal für alle Personen ein Recht auf „aussagekräftige Erklärungen der Logik“, wenn automatisierte Entscheidungen gefällt werden. Ein Beispiel für eine solche Entscheidung ist z. B. die Berechnung eines Kreditscores, welcher die Kreditwürdigkeit einer Person repräsentiert, durch ein Modell. Alle Beteiligten sind sich einig, dass die Umsetzung eines solchen Prinzips dringend erforderlich ist und dass es sich dabei um eine große, offene wissenschaftliche Herausforderung handelt. Erklärbare Modelle werden benötigt, um das Vertrauen der Benutzer in ein trainiertes System sicherzustellen, um faire und begründbare Entscheidungen zu erhalten, oder um Einsicht und tieferes Verständnis der zu analysierenden Daten zu erlangen.

Guidotti et al. (2018) geben einen detaillierten Überblick über die Erklärbarkeit von KI-Modellen. Sie unterscheiden verschiedene Dimensionen der Interpretierbarkeit.

- Ein TNN ist global interpretierbar, wenn die gesamte Logik des Modells klar ist.
- Das Modell ist lokal interpretierbar, wenn nur die Gründe für eine einzelne Prognose erklärt werden können.



**Abb. 6.79** Drei einfache Modelltypen, die geeignet sind einen Zusammenhang zu erklären: Entscheidungsbaum (oben links), Regelmenge (oben rechts) und lineares Modell (unten)

## 6.10.2 Globale Erklärungsmodelle

Ein Ansatz zur Erklärung von TNN ist die Bestimmung einfacher globaler Erklärungsmodelle, welche die Prognosen des TNN weitgehend imitieren und andererseits interpretierbar sind. Die wichtigsten derartigen Modelle sind Entscheidungsbäume, Regelmengen und lineare Modelle, welche in Abb. 6.79 dargestellt sind.

Bei einem Entscheidungsbaum werden sukzessive Fragen zu einzelnen Merkmalen beantwortet. Abhängig von der Antwort werden sukzessive zu anderen Merkmalen Fragen beantwortet. Zum Schluss gibt es in Abhängigkeit von der gefundenen Merkmalskombination eine Prognose für die gewünschte Größe.

Bei einer Regelmenge werden nacheinander Regeln abgearbeitet. Diese bestehen aus logischen Bedingungen zu Merkmalswerten und geben eine Prognose für eine Hilfsvariable oder die gewünschte Größe ab. Wichtig ist, dass die Regeln in der vorgegebenen Reihenfolge behandelt werden.

Ein lineares Modell prognostiziert eine AusgabevARIABLE, z. B. eine Temperatur oder aber (über eine Softmaxfunktion) eine Wahrscheinlichkeit. Die Eingabemerkmale werden mit gesonderten Faktoren multipliziert und alle Beiträge aufaddiert. Die Faktoren zeigen den positiven oder negativen Beitrag jeder EingabevARIABLEN. „const“ ist ein Merkmal mit dem festen Wert 1, welches immer vorhanden ist und die globale Tendenz repräsentiert. Eine Variante des linearen Modells ist das verallgemeinerte additive Modell (generalized additive model, GAM). Hier werden die einzelnen Merkmale eines linearen Modells noch zuvor mit einer glatten Funktion transformiert, z. B. mit einem Logarithmus oder mit

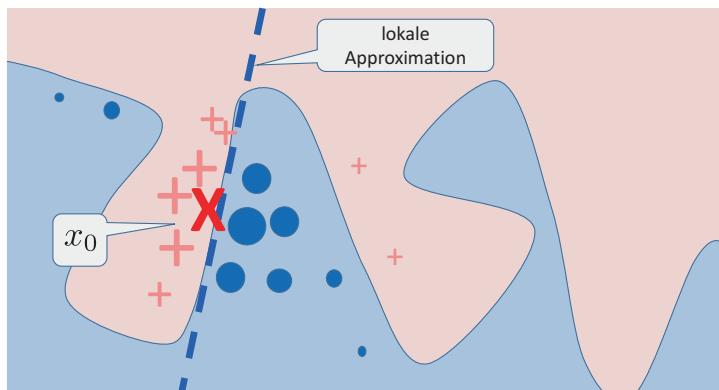
$a_2x^2 + a_1x$ . Die Parameter des linearen Modells zeigen dann immer noch die Beiträge der einzelnen Merkmale zur Prognose an.

Selbst wenn Modelle dieser Art verwendet werden, so hängt die Interpretierbarkeit von der Komplexität der Modelle ab, z. B. gemessen an der Anzahl der Merkmale oder der Merkmal-Wert-Kombinationen. Denn hat ein Entscheidungsbaum mehrere Dutzend Entscheidungsabfragen, so ist das Zusammenspiel dieser Abfragen oft auch nicht durchschau-bar. Die Erklärungsmodelle sind direkt anwendbar auf numerische Daten. Schwieriger wird es bei Bildern, Texten und anderen Sequenzen. Hier müssen erst gut interpretierbare Merkmale aus diesen Daten abgeleitet werden. Erste Ansätze sind in Abschn. 5.6.1 dar-gestellt.

### 6.10.3 Lokale Erklärungsmodelle

Ribeiro et al. (2016) haben das LIME-Verfahren vorgeschlagen, welches für beliebige maschinelle Lernverfahren eine nachvollziehbare Erklärung für die Prognose einer einzelnen Eingabe  $x_0$  liefern kann. Die Idee dieses lokalen Erklärungsmodells besteht darin, dass die Erklärung lokal aus den Trainingsdaten abgeleitet werden kann, die in der Nachbarschaft des zu erklärenden Datensatzes  $x_0$  liegen und gemäß ihrer Entfernung zu  $x_0$  gewichtet werden.

Das Konzept lässt sich anhand von Abb. 6.80 erklären. Es geht hier darum, eine Eingabe  $x_0$  anhand eines Klassifikators als „blau“ oder „rosa“ zu klassifizieren. Die Grenze zwischen „blau“ und „rosa“ ist eine komplexe gebogene Linie. LIME versucht nun nicht, die komplette Grenze zu erfassen, sondern möchte nur die Grenze in der Nähe der zu klas-sifizierenden Eingabe  $x_0$  erklären.

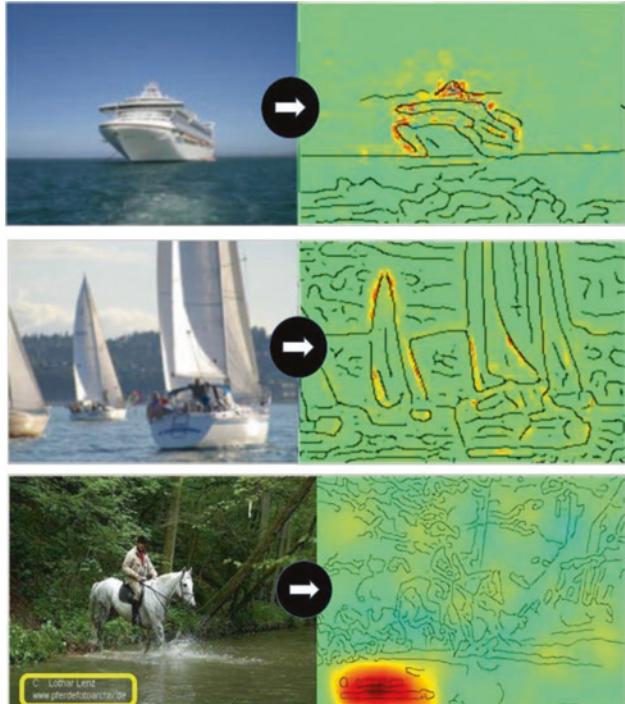


**Abb. 6.80** LIME erklärt eine TNN-Klassifikation lokal in der Nähe einer Eingabe  $x_0$  (x rot). Es werden die Trainingsdaten in der Nähe von  $x_0$  gesucht und durch Ihren Abstand zu  $x_0$  gewichtet. Die Gewichte werden durch die Größe der Punkte symbolisiert. Anschließend wird ein lineares Modell für die gewichteten Trainingspunkte berechnet (gestrichelte Linie) und dieses zur Erklärung der Prognose genutzt. Bildnachweis im Anhang

Dazu berechne man die euklidischen Abstände zwischen  $x_0$  (rotes x) und den anderen Datenpunkten. Die inverse Distanz definiert die Gewichte der Datenpunkte (Kreuze und Punkte), welche durch ihre Größe symbolisiert sind. Mit Hilfe dieser gewichteten Datenpunkte wird ein lineares Modell trainiert (gestrichelte Linie), welches die Klassifikation in der Nähe von  $x_0$  gut erklärt. Die Größe der Parameter des linearen Modells bildet in Abb. 6.79 (unten) die Grundlage einer Erklärung. Hall (2019) gibt detaillierte Hinweise zur Nutzung dieser Technik.

Die ebenenweise Relevanz-Propagierung (LRP) ist eine Methode zur Erklärung (Montavon et al. 2018), die auf viele maschinelle Lernverfahren anwendbar ist. In einer ersten Phase findet für ein Testbeispiel eine Vorwärtspropagierung statt, bei der die Werte der verborgenen Ebenen gespeichert werden. In einer zweiten Phase werden die Ausgabeaktivierungen der letzten Schicht auf die Komponenten der vorherigen verborgenen Vektoren so verteilt, dass die Summe der Aktivierungen erhalten bleibt (Montavon et al. 2018). Auf diese Weise können die Aktivierungen schrittweise bis zur Eingabe, z. B. den Eingabepixeln eines Bildes, zurückpropagiert werden. Das Verfahren kann in den theoretischen Rahmen einer Taylor-Approximation eingebettet werden. Abb. 6.81 zeigt in den oberen beiden Bildern die LRP-Erklärung für die Klassifizierung als Schiff. Hierbei werden bei dem Passagierschiff die Kontur und die Aufbauten markiert, während bei den Segelschiffen die Segel eine entscheidende Rolle spielen. Bei der gleichen Klasse können also sehr unterschiedliche Eigenschaften für die Klassifikation wichtig sein. In dem unteren Bild

**Abb. 6.81** Erklärung der Klassifikation von Bildern. Für die Klassifikation als Schiff sind unterschiedliche Merkmale durch rote Einfärbung als relevant markiert (oberste zwei Bilder). Die Klassifikation des untersten Bildes als Pferd beruht allein auf dem Quellenhinweis unten links (Lapuschkin et al. 2019). Bildnachweis im Anhang



sieht man, dass der Quellenhinweis unten links für die Klassifikation als Pferd relevant ist. Entfernt man diesen Hinweis, so wird das Bild nicht mehr als Pferd klassifiziert. Es stellte sich heraus, dass die Quellenangabe auf allen Trainingsbildern mit Pferden vorhanden war. Dies zeigt, dass die Erklärung der Klassifikation notwendig ist für die Beurteilung der Übertragbarkeit des Klassifizierers auf neue Daten. Lapuschkin et al. (2019) haben eine Reihe von Programmen zur Erklärung von TNN zur Verfügung gestellt.

## 6.11 Zuverlässigkeit der Texterkennung

### 6.11.1 Robustheit bei Textfehlern und Änderung der Domäne

Menschen haben eine erstaunlich hohe Fähigkeit, Texte mit Fehlern zu lesen. Sie können trotz massiver Tippfehler noch den Sinn eines Satzes verstehen. Ein Beispiel ist folgender Text aus Twitter: „Government confirms blast n nuclear plants n japan ... don't knw wht s gona happen nw ...“ (Ritter et al. 2011). Wendet man auf Twittertexte ein Standardmodell zur Namenserkenntnis (Abschn. 6.4.4) an, so erhält man einen F-Wert von 0,44. Durch Verwendung von Trainingsdaten, welche diese Fehler abdeckten, konnte man mit dem gleichen Modell einen F-Wert von 0,67 erreichen. Ritter et al. (2011) zeigen damit, dass es unbedingt notwendig ist, für Twitter und ähnliche Domänen eigene Trainingsdaten zu verwenden.

Karpukhin et al. (2019) untersuchen die Anfälligkeit von Sequence-to-Sequence-Modellen zur Übersetzung auf Textfehler. Sie analysieren den Einfluss von Zeichenfehlern auf das Übersetzungsergebnis. Werden beispielweise bei einer Übersetzung von Deutsch nach Englisch im Eingabesatz 10 % der Buchstaben verändert, so sinkt der BLEU-Wert von 34,2 auf 27,9 (Abb. 6.82). Die Autoren schlagen vor, während des Trainings des Übersetzungsmodells unterschiedliche Arten von Buchstabenveränderungen an den Worten der Trainingssätze durchzuführen: Löschung, Einfügung, Ersetzung und Buchstabenvertauschung. Diese Änderungen werden dynamisch vorgenommen, sodass in jeder Epoche andere Modifikationen stattfinden. Die Verteilung der Buchstabenänderungen

Sprachen	% Eingabefehler	BLEU bei Training ...	
		ohne Fehler	mit Fehlern
de-en	0,0%	34,2	33,5
de-en	9,7%	27,9	31,3
de-en	39,4%	12,5	23,3
fr-en	0,0%	39,6	39,9
fr-en	13,5%	30,5	34,1
fr-en	53,7%	11,5	19,4

**Abb. 6.82** BLEU-Wert für die maschinelle Übersetzung bei unterschiedlich hohen Eingabefehlern im zu übersetzenden Text (3. Spalte). Werden beim Training diese Fehler zufällig hinzugefügt, so verbessert sich der BLEU-Wert (4. Spalte). (Karpukhin et al. 2019)

gen orientiert sich dabei an der empirischen Häufigkeit von Tippfehlern. Wie in der letzten Spalte von Abb. 6.82 ersichtlich, führt die zusätzliche Verwendung fehlerhafter Trainingstexte zu einer wesentlichen Steigerung des BLEU-Wertes, ohne die Qualität bei der Übersetzung fehlerfreier Texte merklich zu beeinträchtigen.

### 6.11.2 Anfälligkeit für böswillige Änderung von Eingaben

In Abschn. 5.12.2 wurde gezeigt, dass sich Bilder bei einer adversarialen Attacke durch minimale Änderungen so manipulieren lassen, dass das Ergebnis der Bildklassifikation verändert wird. Dieses Vorgehen lässt sich aber nicht direkt auf die Verarbeitung textueller Daten übertragen. Zum einen sind Pixelwerte kontinuierlich, aus Buchstaben und Wörtern zusammengesetzte Texte aber diskret. Zum anderen sind kleine Änderungen bei Pixelwerten für Betrachter kaum erkennbar, wohingegen Änderungen an Buchstaben sofort ins Auge fallen.

In Abb. 6.83 finden sich adversariale Beispiele für die automatische Sprachübersetzung (Ebrahimi et al. 2018). Ziel des Angreifers ist es, in der Übersetzung einen Zieltexxt abzuändern und gegebenenfalls durch einen Solltext zu ersetzen. Dazu werden zunächst über die Berechnung von Ableitungen Worte oder Silben im Ausgangstext gesucht, welche die Wahrscheinlichkeit des Zieltexxes stark beeinflussen. Diese Worte oder Silben werden durch andere Textteile ersetzt, die das Zielwort vermeiden und gegebenenfalls in der Übersetzung den Solltext produzieren. In Abb. 6.83 werden zwei Beispiele für adversariale Eingaben und die daraus entstehenden Übersetzungen gezeigt.

Einen Überblick über adversariale Attacken bei der Verarbeitung natürlicher Sprache geben Zhang et al. (2019). Hierbei gibt es sehr viele verschiedene Methoden zur Generierung der Änderungen und auch sehr viele Anwendungsbereiche, z. B. Textklassifika-

Original: Eingabe	1901 wurde eine Frau namens <b>Auguste</b> in eine medizinische Anstalt in Frankfurt gebracht.	Das ist Dr. Bob Childs – er ist Geigenbauer und <b>Psychotherapeut</b> .
Adversarial: Eingabe	1901 wurde eine Frau namens <b>Afui-guste</b> in eine medizinische Anstalt in Frankfurt gebracht.	Das ist Dr. Bob Childs – er ist Geigenbauer und <b>Psy6hotearpeiut</b> .
Original: Ausgabe	In 1931, a woman named <b>Augustine</b> was brought into a medical institution in Frankfurt.	This is Dr. Bob Childs – he's a violin maker and a <b>psychotherapist</b> .
Adversarial: Ausgabe	In 1931, a woman named <b>Rutgers</b> was brought into a medical institution in France.	This is Dr. Bob Childs – he's a brick maker and a <b>psychopath</b> .

**Abb. 6.83** Adversariale Beispiele für die automatische Übersetzung von Deutsch nach Englisch. Im ersten Beispiel will der Angreifer den Namen der Person unterdrücken. Im zweiten Beispiel will er „therapist“ durch „psychopath“ ersetzen. (Ebrahimi et al. 2018)

tion, maschinelle Übersetzung, Fragen beantworten, Dialogsysteme etc. Insgesamt ist die Erzeugung adversarialer Beispiele schwieriger als in der Bildverarbeitung und wird häufig noch manuell durchgeführt. Als Abwehrmaßnahmen werden unterschiedliche Strategien diskutiert. Die wichtigste Abwehrstrategie ist adversariales Training, bei dem ein TNN sowohl mit den normalen Trainingsbeispielen als auch mit den adversarialen Beispielen trainiert wird.

---

## 6.12 Zusammenfassung und Trends

Tiefe neuronale Netze verarbeiten textuelle Eingaben heute fast immer als Embeddings. Bisher wurden sie mit vortrainierten einfachen Modellen berechnet, etwa Word2vec. Heute werden sie oft gleichzeitig mit dem Modell trainiert. Die Differenzen von Embeddings sind oft inhaltlich als Relationen interpretierbar.

Sprachmodelle haben die Aufgabe, das jeweils nächste Wort eines Satzes zu prognostizieren. Das Long-Short Time Memory (LSTM) war das erste erfolgreiche Sprachmodell. Es war auch die Grundlage für die ersten Übersetzungsmodelle, die aus zwei LSTMs bestanden: Das erste Encoder-LSTM kodiert den Eingabesatz in einen verborgenen Vektor. Das zweite Decoder-LSTM dekodiert diesen verborgenen Vektor in einen Satz der Zielsprache.

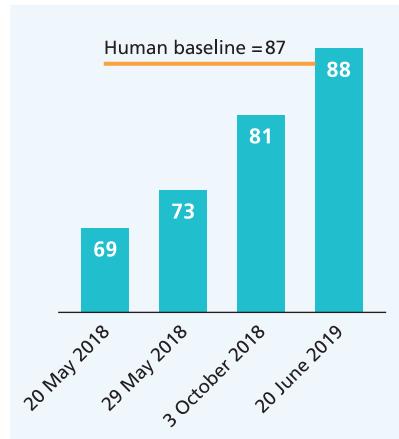
Diese Übersetzungsmodelle konnten mit Attention verbessert werden. Dabei wird die Ähnlichkeit zwischen einem verborgenen Vektor des Decoder-LSTMs und den verborgenen Vektoren des Encoder-LSTMs berechnet. Entsprechend dieser Ähnlichkeit werden die verborgenen Vektoren des Encoder-LSTMs zur Erzeugung des Ausgabewortes benutzt.

Das Transformermodell erweitert und verallgemeinert diesen Mechanismus und führt ihn in vielen Schichten parallel durch. Zudem wird eine Selbstattention auf den Embeddings des Eingabesatzes mehrfach durchgeführt, wodurch kontextsensitive Embeddings erzeugt werden, die sehr viele Informationen über die Nachbarwörter enthalten. Hierdurch werden die Übersetzungsmodelle bei gleicher oder höherer Genauigkeit schneller trainierbar.

Das BERT-Modell hat die Selbstattention für zwei Trainingsaufgaben angewendet: beim maskierten Sprachmodell müssen einzelne fehlende Worte prognostiziert werden und beim Fortsetzungsmodell muss entschieden werden, ob der zweite Satz die Fortsetzung des ersten ist. Dieses Modell wurde mit sehr vielen Schichten ausgestattet und auf sehr großen Textsammlungen trainiert. Es erreichte als Sprachmodell die bislang höchste Genauigkeit.

Durch wenige Iterationen auf einem kleinen Trainingsdatensatz lässt sich das BERT-Modell durch Transferlernen auf eine große Zahl von semantischen Problemen anwenden, z. B. Fragen beantworten, logische Folgerungen testen etc. Hier wurden in der letzten Zeit viele neue Bestmarken erzielt, und oft eine höhere Genauigkeit als bei Menschen. Abb. 6.84 zeigt die Entwicklung der Genauigkeit bei dem GLUE-Benchmark, der eine Mischung solcher Aufgaben umfasst. Dies ist ein echter Durchbruch im Bereich der Textinterpretation.

**Abb. 6.84** Erhöhung der Genauigkeit bei GLUE, einer Sammlung von semantischen Textanalyseproblemen im Laufe eines Jahres. Mittlerweile ist dort die menschliche Leistung erreicht. Bildnachweis im Anhang



Ein kurzer Abschnitt behandelt die Beschreibung von Bildern durch Text. Dies ist z. B. möglich durch ein CNN zur Bildverarbeitung, einem anschließenden LSTM-Encoder und einem LSTM-Decoder, der den Text generiert. Die Genauigkeit dieser Modelle ist noch wesentlich schlechter als beim Menschen.

Die Beschreibung von Bildern durch Text erfolgt über Trainingsdaten, bei denen die im Bild enthaltenen Objekte durch Bildunterschriften annotiert sind. Meist fehlt dabei eine Beschreibung der im Bild sichtbaren Aktionen. Derart trainierte Modelle machen scheinbar offensichtliche Fehler, wenn sie versuchen, visuelle Szenen zu beschreiben. Vielmehr werden auch Tätigkeitsbeschreibungen für das Training benötigt. Dies legt nahe, dass ein KI-System ein vernünftiges Weltmodell von Objekten und Menschen haben muss, um wirklich zu verstehen, was in einer visuellen Szene geschieht.

Die diskutierten Modelle sind anfällig für Textfehler. Allerdings kann man die Trainingsdaten durch simulierte Fehler erweitern und erreicht so eine viel höhere Robustheit. Adversariale Angriffe sind kein so großes Problem wie bei Bildern, da man Textmodifikationen sehr leicht erkennt.

Wir haben gesehen, dass Worte, Sätze oder auch ganze Dokumente durch Embeddings in Form von Vektoren beschrieben werden. Dieses Lernen von Repräsentationen ist ein allgemeiner Ansatz (Bengio et al. 2013), der in den weiteren Kapiteln auch auf andere Medien (Sprache, Videos, Aktionen, etc.) ausgedehnt wird. Diese Repräsentationen werden während des Trainings für spezielle Aufgaben gelernt, und ihre Interpretation hängt immer von der jeweiligen Netzwerkarchitektur ab. Der Vorteil dieser Repräsentationen ist, dass sie die zuvor verarbeiteten Inhalte in kondensierter Form zusammenfassen und dass sie nahtlos von nachfolgenden Schichten weiterverarbeitet werden können.

### Trends

- Im Bereich der Sprachübersetzung werden derzeit Ansätze untersucht, Übersetzungsmodele auch für Sprachen ohne umfangreiche bilinguale Trainingsdaten zu trainieren. Diese Modelle starten mit wenigen Wörtern, für die Übersetzungen bekannt sind. Sie trainieren Sprachmodelle auf einsprachigen Textsammlungen, nutzen Übersetzung und Rückübersetzung und erzeugen hieraus erstaunlich leistungsfähige Übersetzungsmodelle (Lample et al. 2018).
- Das vortrainierte BERT-Modell kann Weltwissen aus sehr großen Textsammlungen extrahieren. Nun gibt es Ansätze, dieses Weltwissen in strukturierterer Form zu erwerben. Sap et al. (2019) haben umfangreiche Trainingsdaten gesammelt, in denen zu Ereignissen plausible Vorbedingungen und Konsequenzen zusammengestellt werden. Ein Beispiel ist das Ereignis „Person X backt Brot“. Vorbedingungen sind z. B. „Zutaten kaufen“ oder „Backofen anschalten“, während mögliche Resultate wären z. B. „wird schmutzig“ oder „isst“. Aus dieser Art von Daten lassen sich TNN trainieren, welche Informationen über plausible Vorbedingungen oder Konsequenzen von Ereignissen besitzen.
- Die von vortrainierten Sprachmodellen erzeugten Embeddings werden allgegenwärtig. Es wird selten vorkommen, dass ein aktuelles Modell derartige Embeddings nicht verwendet. Zusätzlich wird es Repräsentationen für spezialisierte Inhalte geben, welche nicht direkt in den Embeddings der Sprachmodelle enthalten sind. Diese unterschiedlichen Embeddings und Repräsentationen können dann kombiniert werden.
- Mittlerweile ist es möglich, zur Beantwortung von Fragen die Ergebnisse von großen Sprachmodellen mit einer Websuche zu kombinieren. Riedel et al. (2020) nutzen das BART-Modell (Lewis et al. 2019), um eine Frage zu kodieren, hierzu Suchergebnisse aus dem Internet einzuholen und schließlich die Suchergebnisse mit dem Hintergrundwissen des Sprachmodells zu einer Antwort zu kombinieren.

---

## Literatur

- Aharoni, R., Johnson, M. und Firat, O. 2019.** Massively multilingual neural machine translation. [Online] 2019. [Zitat vom: 21. 10. 2019.] arXiv preprint arXiv:1903.00089.
- Bahdanau, D., Cho, K. und Bengio, Y. 2015.** Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. *ICLR*. 2015.
- Bapna, Ankur und Firat, Orhan. 2019.** Exploring Massively Multilingual, Massive Neural Machine Translation. [Online] 11. 10. 2019. [Zitat vom: 21. 10. 2019.] <https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html>.

- Bar-Hillel, Y.** 1962. The future of machine translation. Times Literary Supplement. Times Newspapers, London. April 20th. 1962. (1969)
- Bengio, Y., Courville, A., Vincent, P.:** Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* **35**(8), 1798–1828 (2013)
- Bojanowski, P., et al. 2017.** Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. 2017.
- Bojar, O., et al. 2017.** Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation* (pp. 169–214). 2017.
- Britz, D., et al. 2017.** Massive Exploration of Neural Machine Translation Architectures. *EMNLP*. 2017.
- Clark, P., et al. 2019.** From ‘F’ to ‘A’ on the N.Y. Regents Science Exams: An Overview of the Aristo Project. [Online] 2019. [Zitat vom: 09. 09. 2019.]
- Devlin, J., et al. 2018.** Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*. [Online] 2018. arXiv:1810.04805.
- Diaz, F., Mitra, B. und Craswell, N. 2016.** Query Expansion with Locally-Trained Word Embeddings. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Vol. 1, pp. 367–377). 2016.
- Dyer, C. 2014.** Notes on noise contrastive estimation and negative sampling. [Online] 2014. arXiv preprint arXiv:1410.8251.
- Ebrahimi, J., Lowd, D., Dou, D.** On adversarial examples for character-level neural machine translation. *COLING* **2018**, 653–663 (2018)
- Elman, Jeffrey L. 1990.** Finding Structure in Time. *Cognitive Science*. **14** (2): 179–211. doi:[https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E). 1990.
- Firth, J.R.** A synopsis of linguistic theory 1930–1955. Studies in linguistic analysis, S. 1–32. Reprinted in F.R. Palmer, ed. (1968). Selected papers of J.R. Firth 1952–1959. Longman, London (1957)
- Guidotti, R., et al.:** A survey of methods for explaining black. box models. *ACM computing surveys (CSUR)* **51**(5), 93 (2018)
- Hall, P. 2019.** On the Art and Science of Explainable Machine Learning: Techniques, Recommendations, and Responsibilities. [Online] 2019. [Zitat vom: 24. 08. 2019.] arXiv:1810.02909v3.
- Hassan, H., et al. 2018.** Achieving human parity on automatic Chinese to English news translation. *arXiv preprint*. [Online] 2018. arXiv:1803.05567. (2018)
- Hendrycks, D. und Gimpel, K. 2016.** Bridging nonlinearities and stochastic regularizers with gaussian error linear units. [Online] 2016. arXiv preprint arXiv:1606.08415.
- Jia, Y., et al.** Direct speech-to-speech translation with a sequence-to-sequence model. [Online] 12. 04 2019. <https://arxiv.org/abs/1904.06037>.
- Johnson, M., et al. 2017.** Google’s multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5, 339–351. 2017.
- Karpathy, Andrej. 2015.** The Unreasonable Effectiveness of Recurrent Neural Networks. *Andrej Karpathy blog*. [Online] 2015. [Zitat vom: 25. 03. 2019.] <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>.
- Karpukhin, V., et al.:** Training on synthetic noise improves robustness to natural noise in machine translation. [Online] 2019. [Zitat vom: 15. 06. 2019.] arXiv preprint arXiv:1902.01509. (2019)
- Kudugunta, S. R., et al. 2019.** Investigating Multilingual NMT Representations at Scale. [Online] 2019. [Zitat vom: 21. 10. 2019.] arXiv preprint arXiv:1909.02197.
- Lample, G., et al.:** Neural architectures for named entity recognition. *Proceedings of NAACL-HLT*, S. 260–270. (2016)
- Lample, G., et al. 2018.** Phrase-based & neural unsupervised machine translation. [Online] 2018. arXiv preprint arXiv:1804.07755.

- Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K. R.** Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, **10**(1), 1–8 (2019)
- Lan, Z., et al. 2019.** Albert: A lite bert for self-supervised learning of language representations. [Online] 2019. [Zitat vom: 09. 01. 2020.] arXiv preprint arXiv:1909.11942.
- Lee, J., et al. 2019.** Biobert: pre-trained biomedical language representation model for biomedical text mining. [Online] 2019. [Zitat vom: 16. 07. 2019.] arXiv preprint arXiv:1901.08746.
- Lenat, D. B. 1995.** CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, **38**(11), 33–38. 1995.
- Levy, O., Goldberg, Y. und Dagan, I. 2015.** Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, **3**, 211–225. 2015.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... Zettlemoyer, L.: Bart:** Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461. (2019)
- Li, X., et al.:** Oscar: Object-semantics aligned pre-training for vision-language tasks. European Conference on Computer Vision. Springer, Cham (2020)
- Lin, T., et al.:** Microsoft COCO: common objects in context. In European conference on computer vision, S. 740–755. Springer, Cham (2014)
- Ma, M., et al.:** STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, S. 3025–3036. (2019)
- Melis, G., et al.:** On the state of the art of evaluation in neural language models. [Online] 2017. arXiv preprint arXiv:1707.05589 (2017)
- Mikolov, T., et al. 2013.** Efficient Estimation of Word Representations in Vector Space. [Online] 2013. [Zitat vom: 22. 03. 2019.] <https://arxiv.org/abs/1301.3781>.
- Montavon, G., Samek, W., Müller, K. R.:** Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* **73**, 1–15 (2018)
- Mushtaq, M. F., et al.:** Neural network techniques for time series prediction: A review. *Int. J. Engineering Information Computing and Appl. (IJE AIS)*, **1**(1). (2019)
- Nayak, Pandu. 2019.** Understanding searches better than ever before. [Online] 2019. [Zitat vom: 25. 10. 2019.] <https://blog.google/products/search/search-language-understanding-bert>.
- Petroni, F., et al.:** Language models as knowledge bases? arXiv preprint arXiv:1909.01066. (2019)
- Popel, M., Bojar, O.:** Training tips for the transformer model. Karls-Universität, Prag (2018)
- Quine, Willard Van Orman.:** Ontological relativity. Chapter 2. Ontological relativity and other essays. Columbia University Press. pp. 26–68. ISBN 0231083572. 1969.
- Rajpurkar, P., et al. 2016.** Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint*. [Online] 2016. arXiv:1606.05250.
- Ribeiro, M. T., Singh, S. und Guestrin, C.:** Model-agnostic interpretability of machine learning. [Online] 2016. [Zitat vom: 24. 08. 2019.] arXiv preprint arXiv:1606.05386. (2016)
- Riedel, S., Kiela, D., Lewis, P. und Piktus, A.:** Retrieval augmented generation: Streamlining the creation of intelligent natural language processing models. (2020) <https://ai.facebook.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models>
- Ritter, A., Clark, S. und Etzioni, O.:** Named entity recognition in tweets: an experimental study. EMNLP (pp. 1524–1534). Association for Computational Linguistics (2011)
- Rumelhart, D.E., Hinton, G.E. und Williams, R.J. 1986.** Learning Representations by back-propagating errors. *Nature* **329** (9), pp. 533–536. 1986.
- Sap, M., et al. 2019.** ATOMIC: an atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, pp. 3027–30). 2019.

- Schmidhuber, Jürgen und Hochreiter, Sepp. 1997.** Long short-term memory. *Neural Comput* 9.8: 1735–1780. 1997.
- Sharma, P., et al. 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* (pp. 2556–2565). 2018.
- Shazeer, N., et al. 2017.** Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. [Online] 2017. [Zitat vom: 21. 10. 2019.] arXiv preprint arXiv:1701.06538.
- Strobelt, H., et al.:** Visual analysis of hidden state dynamics in recurrent neural networks. [Online] 2016. arXiv preprint arXiv:1606.07461 (2016)
- Sutskever, I., Vinyals, O. und Le, Q. V.:** Sequence to sequence learning with neural networks. Advances in neural information processing systems , S. 3104–3112. (2014)
- Thrun, S. (1998).** Lifelong learning algorithms. In Learning to learn. (pp. 181–209). [Buchverf.] S. Thrun und L. Pratt. *Learning to Learn*. s.l. : Springer, Boston, MA., 1998.
- van der Maaten, L.J.P. und Hinton, G.E. 2008.** Visualizing Data Using t-SNE. *Journal of Machine Learning Research*. 9: 2579–2605. 2008.
- Vaswani, Ashish, et al. 2017.** Attention is All you Need. *NIPS 2017: 6000–6010*. 2017.
- Wu, L. Y., et al. 2018.** Starspace: Embed all the things! *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- Yang, Z., et al. 2019.** XLNet: Generalized Autoregressive Pretraining for Language Understanding. [Online] 2019. [Zitat vom: 09. 01. 2020.] arXiv preprint arXiv:1906.08237.
- Zellers, R., et al. 2018.** Swag: A large-scale adversarial dataset for grounded commonsense inference. [Online] 2018. arXiv preprint arXiv:1808.05326.
- Zhang, W. E. et al.:** Adversarial attacks on deep learning models in natural language processing: A survey. <https://arxiv.org/abs/1901.06796>. (2019)
- Zhu, Y., et al. 2015.** Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *Proceedings of the IEEE international conference on computer vision* (pp. 19–27). 2015.



# Gesprochene Sprache verstehen

7

## 7.1 Spracherkennung

Spracherkennung ist die Erfassung von gesprochener Sprache in Tonsignalen und die Übertragung von Sprache in Text. Eine gutes Spracherkennungssystem ersetzt die Schreibkraft, welche früher zum Diktat erschien und anschließend die Korrespondenz in die Maschine tippte. Heute ermöglicht Spracherkennung auch die verbale Steuerung von vielen Geräten, angefangen von der Telefonfunktion in Autos über das Smartphone (Abb. 7.1) bis hin zur verbalen Steuerung des Lichts oder der Heizung in der Wohnung. Darüber hinaus werden durch Spracherkennung viele Dialoganwendungen möglich, z. B. die für die zentrale Rufannahme und die Abfrage der Anliegen von Anrufern in Unternehmen oder auch für die Reservierung von Terminen in Restaurants. Ein besonders boomendes Anwendungsbereich sind persönliche Assistenten, wie Alexa von Amazon, Siri von Apple oder Google Home. Mit einem solchen persönlichen Assistenten kann der Nutzer Musik abspielen lassen, seinen Terminkalender bearbeiten, Suchanfragen im Internet stellen usw. Der persönliche Assistent wird damit ein „Diener“, der zu jeder Tageszeit zur Verfügung steht und die Wünsche des Benutzers erfüllt.

### 7.1.1 Warum ist Spracherkennung schwierig?

Die Menschen können Sprache im allgemeinen ohne Anstrengung verstehen. Für Maschinen ist dies aus einer Reihe von Gründen schwierig. Wenn jemand zu uns spricht, z. B. in einem Restaurant, so müssen wir die Worte von dem Hintergrundgeräusch trennen. Diese Hintergrundgeräusche können verschiedene Ursachen haben, z. B. Unterhaltungen anderer Personen, Verkehrslärm, Wind etc. Eine weitere Schwierigkeit ist, dass Personen unterschiedlich schnell sprechen und man im Fluss der Sprache meist nicht hören kann, wann

**Abb. 7.1** Spracherkennung durch die Gboard Tastatur-App auf dem Smartphone



die Worte enden oder wann ein neuer Satz anfängt. Zudem hat jede Person eine eigene Sprachmelodie und es gibt viele Dialekte und Akzente. Außerdem sind gesprochene Sätze grammatisch oft nicht korrekt und es gibt viele Füllwörter wie „hmm“ und „äh“, die erkannt und weggelassen werden müssen. Aber auch in der Sprache selbst gibt es oft Mehrdeutigkeiten. Viele Worte hören sich ähnlich an, haben aber sehr unterschiedliche Bedeutungen, z. B. küsste und Küste oder Wahl und Wal.

Besondere Probleme bereitet die Erkennung der Sprache bei unterschiedlichen Sprechern. Daher waren Spracherkennungssysteme zunächst auf einzelne Sprecher ausgerichtet und mussten noch durch Zusatztraining an diesen Sprecher angepasst werden. Bei der Erkennung kontinuierlicher Sprache sind die Anfänge einzelner Worte nicht im Audiostrom durch Pausen markiert. Daher mussten bei den ersten Spracherkennungssystemen die einzelnen Wörter mit Pausen dazwischen diktieren werden.

### 7.1.2 Wie kann man Sprachsignale im Computer darstellen?

Schall besteht aus Bewegungen der Luftmoleküle oder anderer Teilchen. Diese bewegen sich rhythmisch auf die Schallquelle zu und wieder weg. Hierdurch ändert sich der Luftdruck (auch Schalldruck genannt), welcher periodisch ansteigt und wieder abfällt. Bevor der Computer die Sprache erkennen kann, muss ein Mikrofon die Schwingungen der gesprochenen Sprache in ein wellenförmiges elektrisches Signal umwandeln. In Abb. 7.2 ist zum Beispiel der Ton einer Klarinette als Schalldruck über die Zeit dargestellt.

Das aufgenommene analoge Sprachsignal muss dann zur Verarbeitung in ein digitales Signal umgewandelt werden, eine Folge von reellen Zahlen. Dies kann z. B. so geschehen, dass man den Wert des Schalldrucks zu bestimmten Zeitpunkten aufzeichnet und dann im

**Abb. 7.2** Messung des Schalldrucks über die Zeit von 20 msec für den Ton einer Klarinette. Bildnachweis im Anhang



Rechner weiterverarbeitet. Aber das erfordert viel Speicher, weil die Aufzeichnungsrate hoch sein muss.

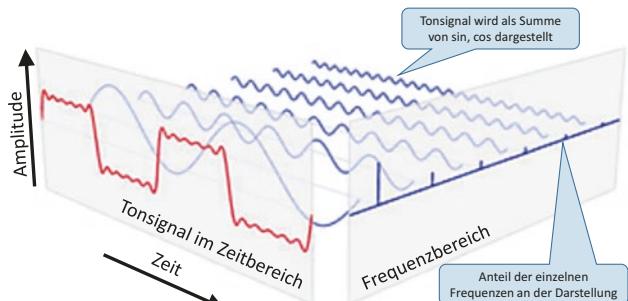
Eine Alternative bietet die Repräsentation des Audiosignals im Frequenzbereich mithilfe der Fouriertransformation, welche die Schwingungen in Frequenzinformationen innerhalb von Zeitintervallen überträgt. Ein solches Signal im Frequenzbereich erfordert weniger Speicher, obwohl die Information über die gesprochenen Worte erhalten bleibt.

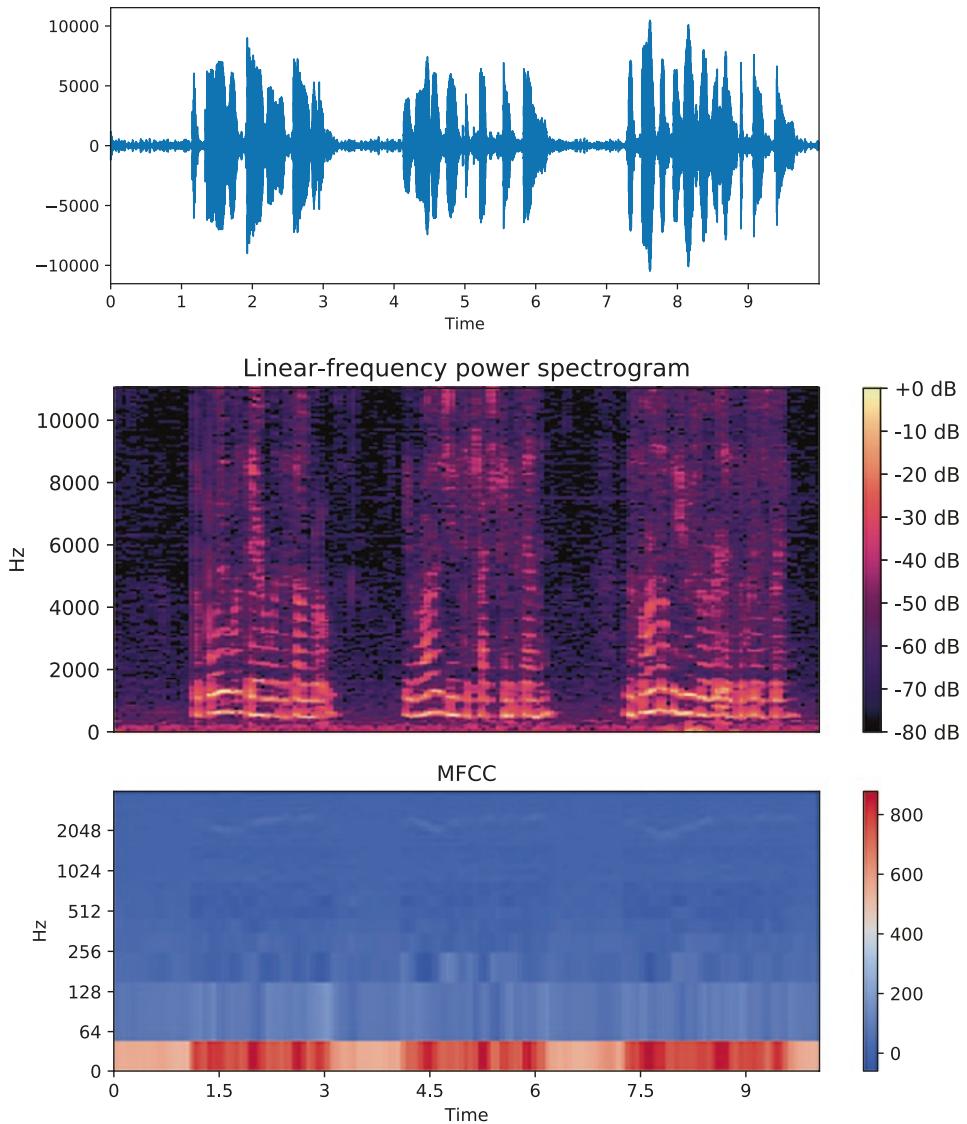
Die Fouriertransformation kann das Audiosignal als eine Summe von einfachen Funktionen – Sinus und Kosinus – repräsentieren. Dies ist in Abb. 7.3 schematisch dargestellt. Hier wird das Audiosignal durch eine Summe von sechs Sinuskurven approximiert. Jede dieser Sinuskurven hat eine bestimmte Frequenz (Schwingungshäufigkeit pro Zeit). Man benötigt dann nur noch die „Anteile“ jeder der Kurven – sechs Zahlen – zur Repräsentation des Signals. Wichtig ist, dass nur die für menschliche Sprache relevanten Frequenzen (20 Hertz bis ca. 8000 Hertz) berücksichtigt werden müssen.

In Abb. 7.4 ist das Tonsignal eines gesprochenen Satzes und das zugehörige Spektrogramm dargestellt. Es enthält für jedes Intervall von 10 Millisekunden die Energie innerhalb einzelner Frequenzintervalle. Hohe Energien werden durch die rote Farbe und niedrige Energien durch blaue Farbe symbolisiert.

Für die Spracherkennung ist nicht die Tonhöhe wichtig, welche sich ja bei Frauen, Männern und Kindern unterscheidet. Vielmehr ist die relative Höhe der Frequenzen entscheidend. Zudem sind die Frequenzen unterhalb von 1000 Hz wichtiger als die höheren

**Abb. 7.3** Repräsentation eines Tonsignals (rot) durch Sinuskurven, von denen jede eine spezielle Frequenz hat. Die Anteile der einzelnen Kurven (blau, grün, violett) sind durch kleine Balken (rechts) gekennzeichnet. Bildnachweis im Anhang





**Abb. 7.4** Tonsignal eines Satzes (oben) im Zeitverlauf über 10 Sekunden (x-Achse). In der Mitte ist das Spektrogramm, welches für jedes 10 msec-Intervall die Größe der jeweiligen Schwingungsfrequenzen eines Intervalls darstellt. Rote Farbe zeigt hohe Energie an und blaue Farbe bedeutet niedrige Energie in dem jeweiligen Frequenzintervall. Unten sind die daraus abgeleiteten Mel Frequency Cepstral Coefficients (MFCC) dargestellt

Frequenzen. Diese Anforderungen werden von Varianten des Spektrogramms berücksichtigt, z. B. den Mel Frequency Cepstral Coefficients (MFCC), welche häufig als Eingangssignal für die Spracherkennung genutzt werden. Die Details zu deren Berechnung werden von (MFCC 2019) dargestellt. Sie bestehen aus einem Vektor von etwa 40 Merkmalen für jedes 10-msec-Intervall.

### 7.1.3 Messung der Genauigkeit der Spracherkennung

#### Die Wortfehlerrate WER

Zur Messung der Genauigkeit der Spracherkennung wird die Wortfehlerrate (WER) verwendet. Sie berechnet die „Differenz“ zwischen einem korrekten Referenzsatz und dem vom Modell erkannten Satz (Abb. 7.5). Hierzu werden zunächst die Worte der beiden Sätze möglichst gut einander zugeordnet. Dies kann wie bei der Levenshtein-Distanz mit Hilfe der dynamischen Programmierung durchgeführt werden (Levenshtein 1965). Dabei unterscheidet man folgende Arten von Fehlern:  $S$  ist die Anzahl der Ersetzungen,  $D$  ist die Anzahl der Löschungen und  $I$  ist die Anzahl der Einsetzungen. Die Wortfehlerrate wird dann definiert als  $WER = (S + D + I)/N$ , wobei  $N$  die Länge des Referenzsatzes ist. Betrachten wir z. B. den Referenztext „dieser schöne Tag endet“ und die Modellprognose „diese Tag endet tragisch“. Dann muss man eine Ersetzung („diese“ -> „dieser“), eine Einfügung („schöne“) und eine Löschung („tragisch“) vornehmen, um den Referenzsatz zu erzeugen. Die Wortfehlerrate ist daher  $3/4 = 0,75$ , was sehr schlecht ist.

#### Bekannte Testdatensätze zur Spracherkennung

Ein Standard-Benchmark zur Messung der Genauigkeit der Spracherkennung ist der Switchboard-Korpus. Er enthält Aufnahmen von Telefongesprächen zwischen Fremden über allgemeine Themen wie Sport und Politik. Da die Gesprächsteilnehmer nicht zu eigenen Familienmitgliedern sprechen, ist die Unterhaltung in nordamerikanischem Standard-Englisch. Es handelt sich um etwa 2400 Telefongespräche zwischen 543 Sprecherinnen und Sprechern (302 männlich und 241 weiblich) aus allen Teilen der USA. Eine detaillierte Beschreibung und Beispiele gibt es bei (Switchboard 1997) und in Abb. 7.6.

Ein weiterer oft verwendeter Datensatz ist der LibriSpeech-Korpus (Panayotov et al. 2015). Er besteht aus etwa 1000 Stunden vorgelesener Hörbücher. Die Hauptschwierigkeit



**Abb. 7.5** Wortfehlerrate (WER) zur Messung der Güte der Spracherkennung. Unten ist der korrekte Referenztext und oben der vom Modell erzeugte Text. Als Fehlerarten werden unterschieden: Ersetzung (blau), Einfügung (rot) und Löschung (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.

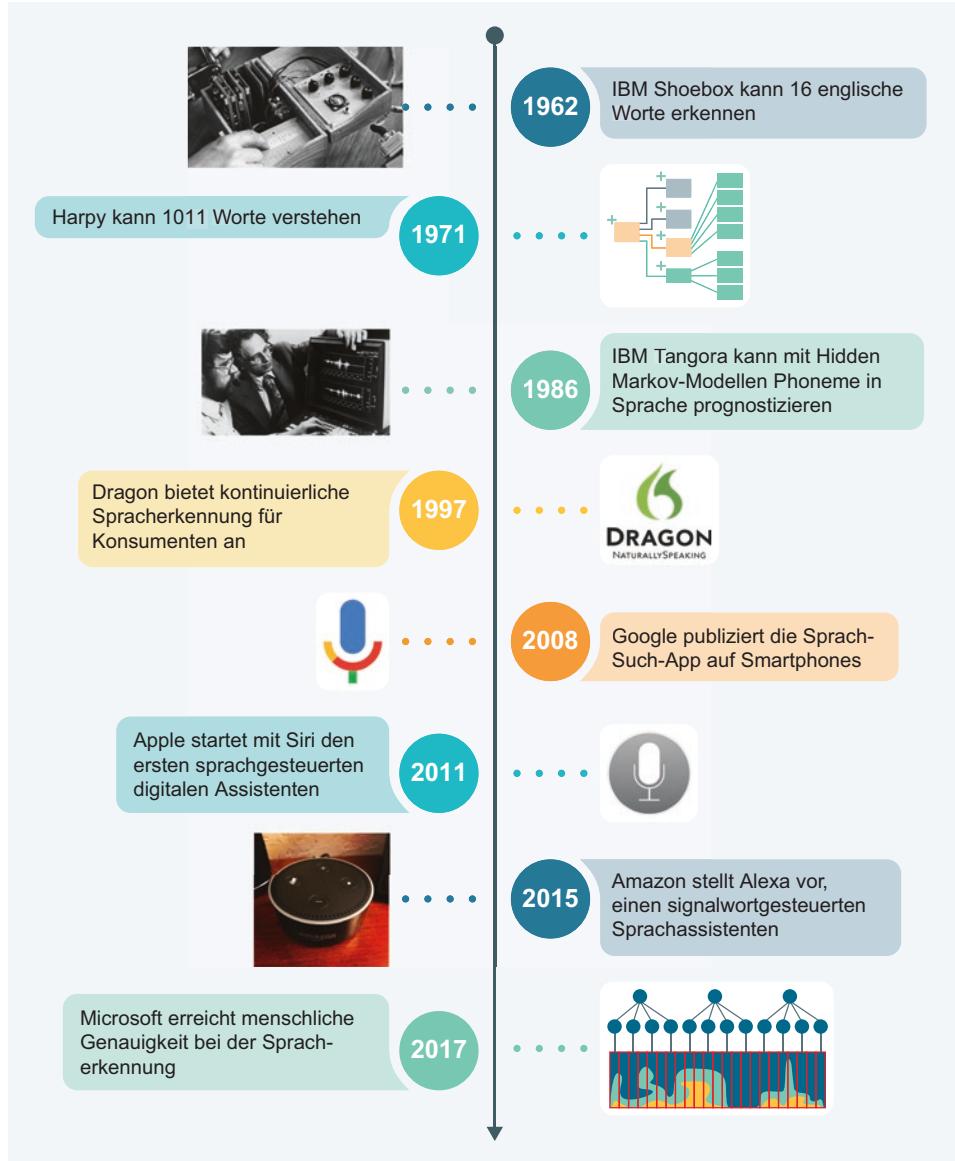
**Abb. 7.6** Beispieldialog aus dem Switchboard-Korpus

bei der Erstellung dieses Korpus war die Prüfung, ob der Vorleser den Text auch korrekt vorgelesen hatte, oder ob Wortersetzungen, Worteinfügungen oder Wortumstellungen stattfanden. Bei der Konstruktion des Korpus wurde ein ausgeklügeltes System von Kontrollen verwendet, um solche Modifikationen zu erkennen.

#### 7.1.4 Die Geschichte der Spracherkennung

In Abb. 7.7 wird die Geschichte der Erkennung gesprochener Sprache skizziert. Schon in den 60er-Jahren gab es erste einfache Modelle zur Erkennung von Worten. In den 80er-Jahren entwickelte IBM unter der Leitung von Fred Jelinek die sprachgesteuerte Schreibmaschine „Tangora“, welche ein Vokabular von 20.000 Worten verstehen konnte. Jelinek nutzte statistische Modelle, welche sehr große Fortschritte in dem Bereich ermöglichen (z. B. das Hidden Markov Model). In den 90er-Jahren erlebte die Spracherkennung mit schnelleren Mikroprozessoren einen ersten Durchbruch mit der Entwicklung von „Dragon Dictate“ für Konsumenten. Die Erweiterung „Dragon Naturally Speaking“ konnte schon 100 Worte pro Minute erkennen. Derartige Modelle bestanden aus mindestens zwei Teilmodellen, einem akustischen Modell zur Abbildung des Sprachsignals auf Phoneme oder Buchstaben und einem Sprachmodell zur Unterscheidung korrekter von unplausiblen Wortfolgen.

Um 2007 übertrafen rekurrente LSTM-Modelle erstmals die Leistungen klassischer Spracherkennungsalgorithmen. Mit der Verwendung weiterer tiefer neuronaler Netze konnte die Fehlerrate der Spracherkennung bereits nach einem Jahr um 30 % gesenkt werden (Markoff 2012).



**Abb. 7.7** Geschichte der Spracherkennung. Bildnachweis im Anhang

## 7.2 Tiefe Sequence-to-Sequence-Modelle

In diesem Abschnitt werden wir zunächst Sequence-to-Sequence-Modelle zur Spracherkennung diskutieren, welche entweder Buchstaben oder auch Teilworte ausgeben. Im folgenden Abschn. 7.3 werden Convolutional Neural Network Modelle zur Spracherkennung

vorgestellt. Da beide Modelltypen unterschiedliche Stärken haben, werden im anschließenden Abschn. 7.3.2 auch kombinierte Modelle betrachtet, welche oft eine bessere Genauigkeit erreichen.

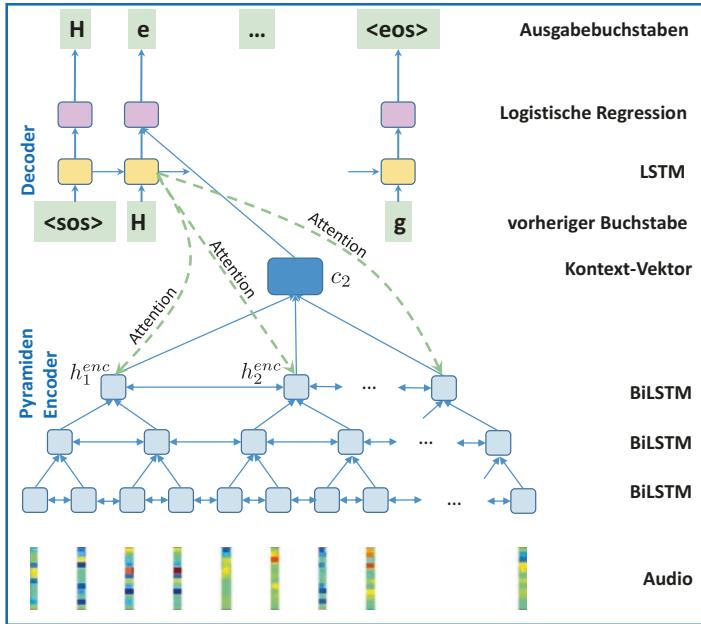
### 7.2.1 Listen-Attend-Spell erzeugt eine Folge von Buchstaben

Das Problem der Spracherkennung lässt sich als eine Übersetzungsaufgabe angehen. Eingabe ist die Folge der Frequenzmerkmale für feste Zeitintervalle von 10 msec. Ausgabe ist die Folge der Buchstaben oder Worte. Nach dem Erfolg der Sequence-to-Sequence Modelle (Abschn. 6.5.1) für die Übersetzung von einer Sprache in eine andere war es naheliegend, diese Modelle auch für die Spracherkennung zu nutzen. Im Gegensatz zu den vorherigen Spracherkennungssystemen, die aus mehreren Teilmodellen bestanden, können alle Teile eines Sequence-to-Sequence Modells Ende-zu-Ende trainiert werden. Dies ist erstrebenswert, weil dann alle Modellteile optimal aufeinander abgestimmt sind.

(Chan et al. 2016) entwickelten eines der ersten Sequence-to-Sequence-Modelle dieser Art, welches die Audiomerkmale in Buchstaben übersetzen konnte. Es wurde Listen-Attend-Spell (LAS)-Modell genannt. Wie bei dem Übersetzungsmodell in Abb. 6.42 wurden als Encoder bidirektionale LSTMs und als Decoder LSTMs mit Attention verwendet, um die Sequenz der Buchstaben des Ausgabetextes zu erzeugen. Allerdings konvergierten diese Modelle zunächst sehr langsam oder gar nicht, weil die Sequenz der Eingabemerkmale (ca. 100 pro sec) im Vergleich zur Anzahl der Ausgabebuchstaben sehr lang war. Daher verkürzten die Autoren die Länge der Encoder BiLSTMs dadurch, dass die verdeckten Vektoren von jeweils zwei aufeinander folgenden Zeitschritten konkateniert wurden und als Eingabe für das BiLSTM der nächsten Schicht dienten. Durch drei Schichten von BiLSTMs mit verdeckten Vektoren der Länge 512 wurde durch diesen „Pyramiden-Encoder“ daher eine Sequenzlänge von 1/8 der Eingabelänge erzeugt. Hierdurch war die Berechnung der Attention analog zu Abb. 6.42 wesentlich einfacher und effektiver.

Der Decoder arbeitet genauso wie bei dem Übersetzungsmodell in Abb. 6.42. Ein zweischichtiges LSTM berechnet den verdeckten Vektor  $h_2^{dec}$  für den nächsten Buchstaben, der prognostiziert werden soll. In Abb. 7.8 ist dies der Buchstabe „e“, wobei der vorige beobachtete Buchstabe „H“ als Eingabe verwendet wurde. Wie beim Übersetzungsmodell wird die Attention zwischen verdeckten Vektoren des Decoders und Encoders berechnet und ein Kontextvektor  $c_2$  aus den verdeckten Encoder-Vektoren  $h_i^{enc}$  gebildet. Dieser Kontextvektor  $c_2$  wird dann an den verdeckten Vektor  $h_2^{dec}$  angehängt und in ein logistisches Regressionsmodell eingegeben, welches die Wahrscheinlichkeitsverteilung für den nächsten Buchstaben berechnet. Auf diese Weise kann während des Trainings die Wahrscheinlichkeit der Buchstaben des Ausgabetextes sukzessive berechnet werden.

In Abb. 7.9 wird die Attention zwischen den Audiomerkmalen (oben) und dem zugehörigen Text „how much would a woodchuck chuck“ durch verschiedene Grauwerte dargestellt. Der Attention-Mechanismus generiert eine explizite Zuordnung der Audiosignale und der Buchstaben. Er kann den Beginn des gesprochenen Satzes in der Audiosequenz

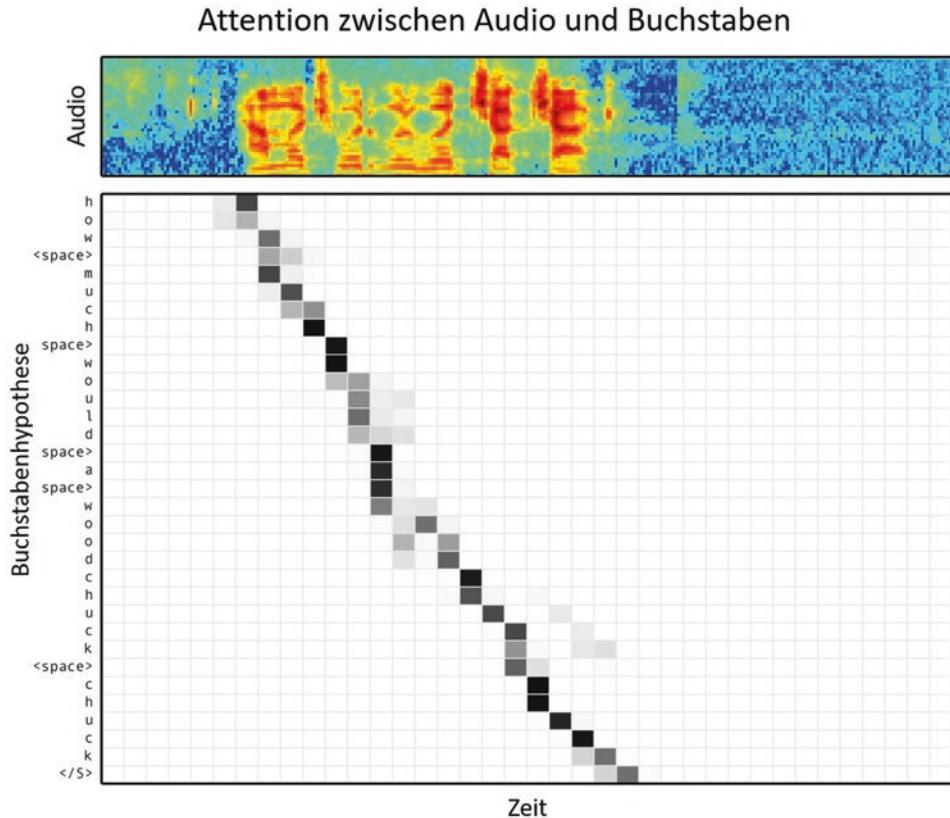


**Abb. 7.8** Spracherkennung mit einem Pyramiden-Encoder und einem Attention-Decoder. Der Encoder besteht aus drei bidirektionalen LSTMs (BiLSTM), welche sukzessive die Länge der Eingabe-Sequenz auf 1/8 verkürzen. Der Decoder besteht aus zwei LSTMs. Die Attention wird analog zu Abb. 6.42 berechnet und damit die Prognose des nächsten Buchstabens (ganz oben) bestimmt. (Quelle: Chan et al. 2016)

korrekt identifizieren. Die Zuordnung der Buchstaben folgt im Allgemeinen der Zeitabfolge in der Audiosequenz. Die Worte „woodchuck“ und „chuck“ sind akustisch ähnlich. Daher gibt es bei den Buchstaben „chuck“ von „woodchuck“ auch leichte Zuordnungen zu mehreren Audiopositionen.

Das Modell wurde auf 2000 Stunden Google-Suchanfragen mithilfe des stochastischen Gradientenabstiegs trainiert (Chan et al. 2016). Zehn Stunden der zufällig ausgesuchten Anfragen wurden als Validationsmenge verwendet. Die Anfragen wurden zum Teil mit unterschiedlichen Arten von Hintergrundrauschen überlagert, welches aus YouTube-Videos von Alltagsumgebungen entnommen wurde. Hierdurch wurde der Umfang der Daten um den Faktor 20 vergrößert. Als Eingaben wurden 40-dimensionale MFCCs für jeweils zehn Millisekunden verwendet. Etwa 22.000 Suchanfragen mit 16 Stunden Audioaufnahmen wurden als Testdaten genutzt. Diese wurden ebenfalls mit Hintergrundrauschen überlagert. Alle Daten wurden manuell in Text transkribiert und in Kleinbuchstaben übertragen. Der Beginn und das Ende wurden mit „<sos>“ und „<eos>“ gekennzeichnet.

Das trainierte Modell erlaubt die Prognose der bedingten Wahrscheinlichkeit des nächsten Buchstabens mithilfe der Ausgabewahrscheinlichkeit  $\hat{p}(y_t | l, y_1, \dots, y_{t-1}, x)$ , wobei  $x$  die Audioeingabe ist und  $y_1, \dots, y_{t-1}$  die bisher erzeugten Buchstaben. Da das Sprach-



**Abb. 7.9** Attention zwischen den Audiomerkmalen der Eingabe (oben) und der Ausgabe-Hypothese, den Buchstaben „how much would a woodchuck chuck“ (links). Je dunkler die Färbung, desto stärker die Zuordnung (Chan et al. 2016). Bildnachweis im Anhang

erkennungsmodell mit relativ wenig Text trainiert wurde, kann es die Regelmäßigkeiten der Sprache nicht so gut abbilden wie textbasierte Sprachmodelle, welche auf wesentlich mehr Text trainiert wurden (Abschn. 6.3.1). Daher wurde die Wahrscheinlichkeit mithilfe eines vortrainierten Sprachmodells skaliert

$$\hat{s}(y_t | y_1, \dots, y_{t-1}, x) = \hat{p}(y_t | y_1, \dots, y_{t-1}, x) * \hat{p}_{SM}(y_t | y_1, \dots, y_{t-1})^\lambda$$

wobei  $\hat{p}_{SM}(y_t | y_1, \dots, y_{t-1})$  die vom Sprachmodell berechnete Wahrscheinlichkeit ist und der Faktor  $\lambda$  mithilfe der Validationsmenge bestimmt wurde.

Bei der Anwendung des trainierten Modells wurde eine Beam-Suche (Abschn. 6.5.1) mit 32 Alternativen zur Bestimmung der Textausgabe verwendet. Die Ergebnisse sind in Abb. 7.10 aufgeführt. Das LAS-Modell alleine erreicht 14,1 % Wortfehlerrate und die Verwendung eines Sprachmodells führt zu einer Verbesserung auf 10,3 % WER. Die Ergeb-

Modell	WER: kein Geräusch	WER: Geräusch
CLDNN-HMM	8.0	8.9
LAS	14.1	16.5
LAS & Sprachmodell	10.3	12.0

**Abb. 7.10** Wortfehlerraten für das LAS-Modell auf den Google-Suchanfragen. Auf der linken Seite sind die Werte für unverfälschte Eingaben und rechts die Werte für die durch Hintergrundgeräusche überlagerten Audiodaten

nisse mit zusätzlichen Geräuschen sind 2 % schlechter. Als Vergleich ist das beste damalige Modell (CLDNN-HMM) aufgeführt, welches 8,0 % WER erreichte.

## 7.2.2 Sequence-To-Sequence Modell für Worte und Silben

Zeyer et al. (2018) verwenden ebenfalls rekurrente neuronale Netze mit LSTM-Zellen (Abschn. 6.4) als Encoder- und Decodernetze des Sequence-to-Sequence-Netzes. Der Decoder erzeugt als Ausgabe 10.000 Worte und Teilworte, die durch Byte-Pair Encoding (BPE) (Abschn. 6.6.5) bestimmt wurden. Hierdurch hat das Decodernetz nicht mehr so viele Teilschritte wie bei der Erzeugung der Buchstaben und der Attentionmechanismus kann komplexere Abhängigkeiten beschreiben. Trotzdem ist das Vokabular nicht beschränkt, da mit BPE beliebige Wörter durch Teilwörtern und Buchstaben dargestellt werden können.

Die Reduktion der Schritte des Encodermodells wurde diesmal mithilfe einer Max-Pooling-Schicht (Abschn. 5.2.3) durchgeführt, die jeweils  $k$  zeitlich aufeinanderfolgende verdeckte Vektoren zu einem Vektor zusammenfasst. Hierbei wurden Werte von  $k = 8, 16, 32$  ausprobiert. Das System wurde schrittweise trainiert, wobei die Anzahl der Encoder-Schichten schrittweise auf sechs erhöht wurde.

Der LibriSpeech-Trainingsdatensatz (Panayotov et al. 2015) besteht aus etwa 1000 Stunden vorgelesener Hörbücher. Der Audioinput wurde durch 40-dimensionale MFCC-Merkmale kodiert. Das System erzielte auf den Testdaten eine Wortfehlerrate von 3,82 %, was zum Zeitpunkt der Publikation (2018) eine neue Bestmarke darstellte. Auf dem Switchboard-Korpus erreicht das Modell von Zeyer et al. (2018) eine Wortfehlerrate von 11,8 %, was zum Zeitpunkt der Publikation (2018) einen Bestwert für ein Ende-zu-Ende trainiertes Modell darstellte.

---

## 7.3 Convolutional Neural Network zur Spracherkennung

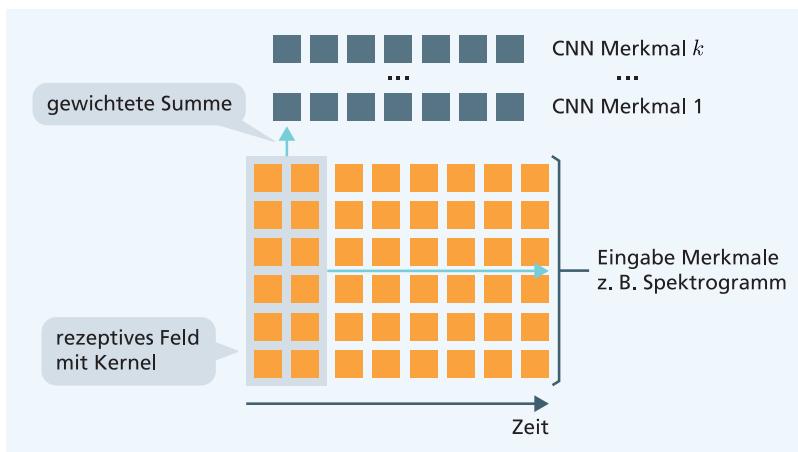
CNN sind als Spracherkennungsmodelle aus mehreren Gründen attraktiv. CNNs sind explizit dafür ausgelegt, sich an zeitliche Verschiebungen und Dehnungen anzupassen. Denn die Sprechgeschwindigkeit unterschiedlicher Sprecher kann sehr verschieden sein. Zudem

kann das Sprachsignal zu beliebigen Zeitpunkten beginnen und es können Sprechpausen toleriert werden. Weiter sind die Variablen im Spektrogramm stark korreliert, was mit CNNs ebenfalls gut verarbeitet werden kann.

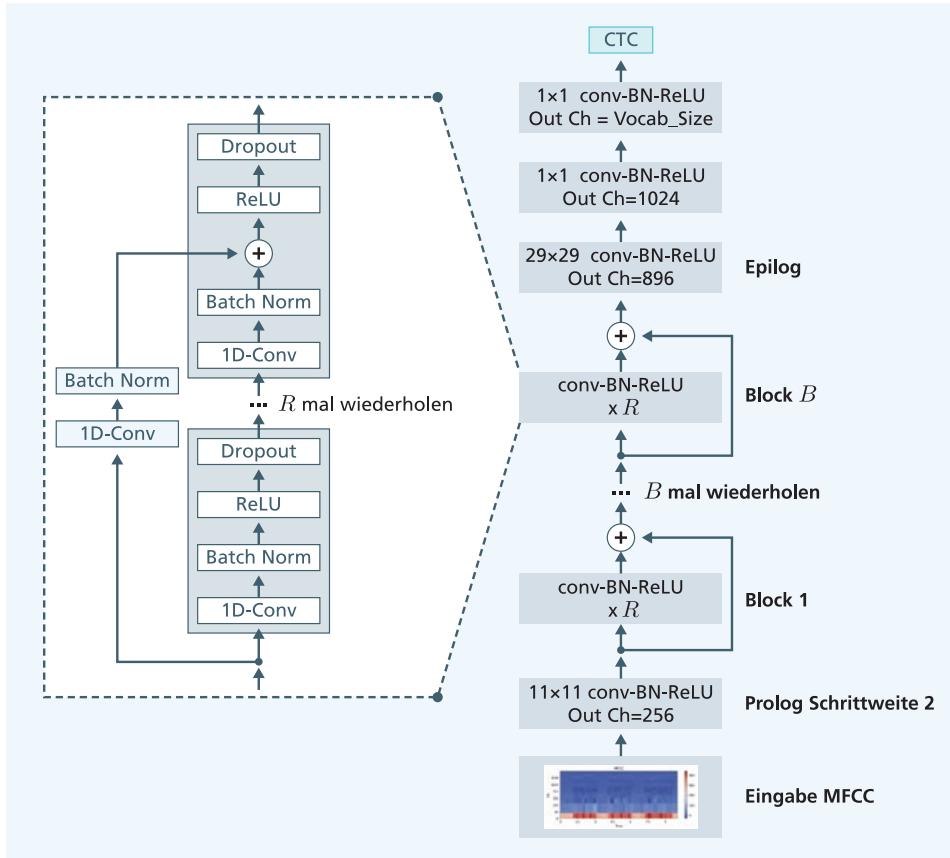
### 7.3.1 CNN Modelle

Li et al. (2019) präsentieren ein solches CNN, genannt „Jasper“. Für jeweils ein Intervall von zehn Millisekunden verwendet es als Eingabemerkmale 40 MFCC Koeffizienten und berechnet eine Ausgabewahrscheinlichkeit für Buchstaben. Jasper verwendet eindimensionale Convolution-Schichten, welche in Abb. 7.11 dargestellt sind. In üblichen CNNs bewegt sich das rezeptive Feld in zwei Dimensionen über die Eingabepixel (Abschn. 5.2.1). Im Gegensatz dazu bewegt sich das rezeptive Feld dieser eindimensionalen CNNs nur in einer Richtung über die Eingabemerkmale und erfasst gleichzeitig alle Eingabemerkmale eines Zeitpunktes (Abb. 7.11). Das rezeptive Feld hat eine Breite und gegebenenfalls eine Schrittweite, mit der es in der Zeitdimension verschoben wird. Die Eingabewerte des rezeptiven Feldes werden mit den Parametern des Kernels multipliziert und erzeugen einen Wert des zugehörigen Ausgabemerkmales. Jasper hat eine Blockstruktur mit  $B$  Blöcken, die jeweils  $R$  Teilblöcke besitzen (Abb. 7.12).

Jeder Teilblock von Jasper besteht aus einer eindimensionalen Convolution-Schicht, einer Batch-Normalisierung, welche die Werte eines Batches auf Mittelwert 0 und Varianz 1 normiert (Abschn. 4.6.4), einer nichtlinearen Aktivierung durch eine ReLU-Schicht und



**Abb. 7.11** Die Spektrogramm-Eingaben werden bei Jasper durch eine eindimensionale Convolution-Schicht verarbeitet. Das rezeptive Feld (grau) wird nur in einer Richtung über die Eingabe bewegt. Zu dem rezeptiven Feld gehört ein Kernel mit Parametern, der die Zahlen im rezeptiven Feld gewichtet addiert und damit ein Merkmal erzeugt. In einer CNN-Schicht werden  $k$  Kernel für unterschiedliche Merkmale trainiert

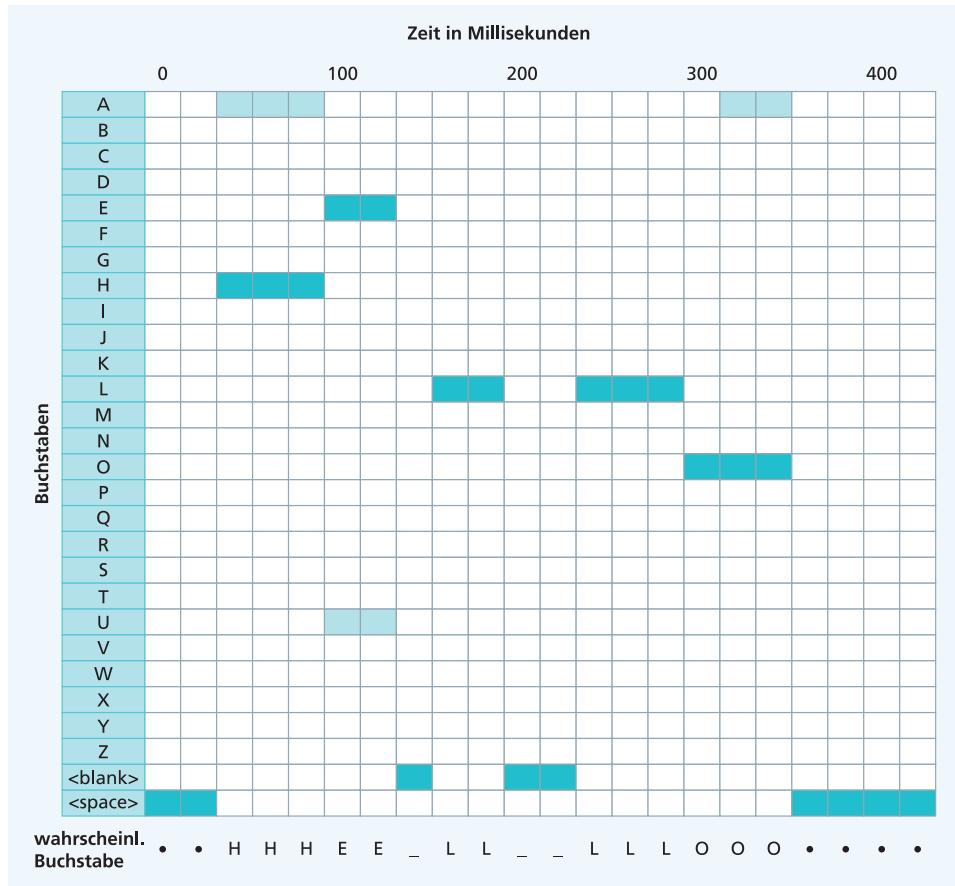


**Abb. 7.12** Das Convolution-Spracherkennungssystem Jasper enthält insgesamt zehn Blöcke mit jeweils fünf eindimensionalen Convolutionen. Die oberste CTC-Schicht berechnet die Wahrscheinlichkeit der auszugebenden Buchstaben (Li et al. 2019)

einer Dropout-Schicht.  $R$  dieser Teilblöcke werden zu einem Block zusammengefasst und wie bei ResNet (Abschn. 5.5.2) durch eine residuale Verbindung (Bypass) überbrückt. Hierdurch wird die Optimierung bei sehr vielen Schichten erst ermöglicht. Die Convolution-Schichten haben eine Breite von 11–25 Zeitschritten.

Zusätzlich gibt es noch einen ersten Convolution-Block mit einer Breite von elf und Schrittweite zwei sowie drei abschließende Convolution-Blöcke. Das Modell Jasper  $10 \times 5$  enthält insgesamt zehn Blöcke mit jeweils fünf eindimensionalen Convolution-Schichten und vier zusätzlichen Schichten mit insgesamt 54 Convolution-Schichten und 333 Millionen Parametern.

Die Convolution-Schichten erzeugen für jedes Zeitintervall eine Schätzung der Buchstabenwahrscheinlichkeiten. Die letzte Schicht ist eine CTC-Schicht (Connectionist Temporal Classification). Abb. 7.13 zeigt die Buchstabenwahrscheinlichkeiten für das Wort



**Abb. 7.13** Die Connectionist Temporal Classification schätzt die wahrscheinlichste Buchstabensequenz aus den Buchstabenwahrscheinlichkeiten für Zeitintervalle. Intensivere Färbung deutet eine höhere Wahrscheinlichkeit an

„Hello“, wobei die intensivere Färbung eine höhere Wahrscheinlichkeit symbolisiert. Offenbar entsprechen den Buchstaben der Ausgabe jeweils mehrere Zeitintervalle mit zusätzlichen Pausen dazwischen. Der CTC-Algorithmus (Graves et al. 2006) berechnet die Zuordnung durch einen effizienten Forward-Backward Algorithmus und kann hierdurch die Wahrscheinlichkeit der Ausgaben berechnen. Oft ergibt sich folgende Lösung: zunächst wird pro Zeitschritt der wahrscheinlichste Buchstabe ausgewählt. Anschließend werden doppelte Buchstaben zu einem vereinigt und alle Pausen eliminiert.

Bei der Erzeugung des Ausgabesatzes während der Beam-Suche (Abschn. 6.5.1) wurde ein Sprachmodell verwendet, welches mit einem sehr großen Trainingsdatensatz trainiert

wurde und daher die Regelmäßigkeiten der Sprache besser erfasst. Die Autoren erzielten die besten Ergebnisse mit einem Transformer-Sprachmodell (Dai et al. 2018), welches die Ausgabewahrscheinlichkeiten neu gewichtet.

Li et al. (2019) erreichen auf dem LibriSpeech test-clean-Datensatz eine Wortfehlerrate von 2,95 %, welche zum Publikationszeitpunkt (2019) die Bestmarke darstellte. Für den Switchboard-Datensatz ergab sich eine Wortfehlerrate von 7,8 %, was über dem derzeitigen Bestwert von 5,5 % liegt. Abb. 7.14 zeigt die typischen Erkennungsfehler von Menschen und automatischer Spracherkennung auf dem Switchboard-Korpus. Es ist zu sehen, dass sich die Fehlermuster recht ähnlich sind.

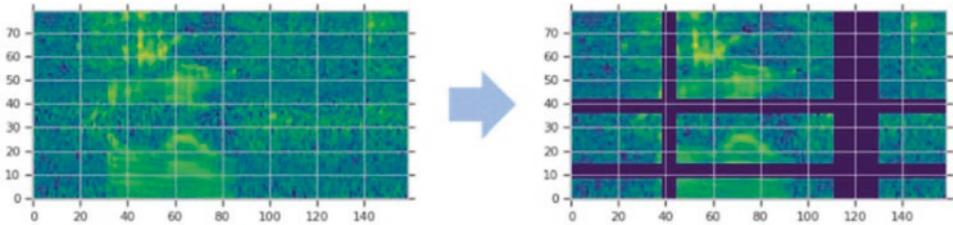
### 7.3.2 Kombinierte Modelle

#### ResNet und BiLSTM

Es zeigt sich, dass Convolutional Neural Networks und rekurrente Modelle unterschiedliche Sprachmerkmale erfassen und daher in einer Kombination zu einer Verbesserung führen können. Xiong et al. (2018) stellten eine solche Kombination vor. Hierbei wird ein CNN mit ResNet-Architektur (Abschn. 5.5.2) und ein weiteres CNN (LACE) mit einem BiLSTM Netz (Abschn. 6.4.4) kombiniert. Es erreicht unter Verwendung zusätzlicher Sprachmodelle zum Neugewichten eine Wortfehlerrate von 5,1 % auf den Switchboard-Testdaten. Dies ist nach Angaben der Autoren besser als die Fehlerrate menschlicher Experten und stellt einen Meilenstein in der Spracherkennung dar.

Ersetzungsfehler		Löschenfehler		Einsetzungsfehler	
Spracherkennung	Mensch	Spracherk.	Mensch	Spracherk.	Mensch
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: %back	11: a	5: yeah	5: yeah

**Abb. 7.14** Häufigste Erkennungsfehler auf dem Switchboard-Testkorpus. Die Zahlen sind normierte Häufigkeiten. Das Muster der Fehler von Mensch und Spracherkennung ist ähnlich. (Quelle: Saon et al. 2017)



**Abb. 7.15** Die Vergrößerung der Trainingsdaten durch SpecAugment. Hierbei wird die Eingabe in Form eines Spektrogramms oder einer ähnlichen visuellen Darstellung modifiziert durch Verformen in der Zeitrichtung, der Maskierung von Zeitschritten und der Maskierung von Frequenzen. Bildnachweis im Anhang

### Vergrößerung der Trainingsdaten

Der Umfang der annotierten Trainingsdaten zur Spracherkennung ist trotz aller Anstrengungen noch beschränkt. Eine Möglichkeit, die effektive Größe der Daten zu erhöhen, ist die Veränderung der Trainingsdaten durch Hinzufügung von Verzerrungen, die Hintergrundrauschen und anderen Umwelteinflüssen entsprechen. Park et al. (2019) stellen eine einfache Technik „SpecAugment“ zur Modifikation von Trainingsdaten vor, welche zu sehr guten Ergebnissen führt. Hierbei wird die Eingabe in Form eines Spektrogramms oder einer ähnlichen visuellen Darstellung modifiziert. Dies geschieht durch Verformen in der Zeitrichtung, der Maskierung von Zeitschritten und der Maskierung von Frequenzen (Abb. 7.15).

Um den Effekt dieser Anreicherung der Trainingsdaten zu messen, wurde ein kombiniertes neuronales Netz verwendet. Das log mel-Spektrum wurde zunächst durch zwei CNN-Schichten transformiert und dann durch ein Listen-Attend-Spell-Netz (Abschn. 7.2.1) verarbeitet. Der Text wurde durch ein Wortteile-Modell dargestellt und die Ergebnisse durch ein Sprachmodell skaliert. Das Verfahren erreicht auf dem Switchboard 300h-Datensatz eine Wortfehlerrate von 6,8 % und auf dem LibriSpeech 960h-Benchmark eine Wortfehlerrate von 2,5 %. Beide Ergebnisse sind eine signifikante Verbesserung des State of the Art.

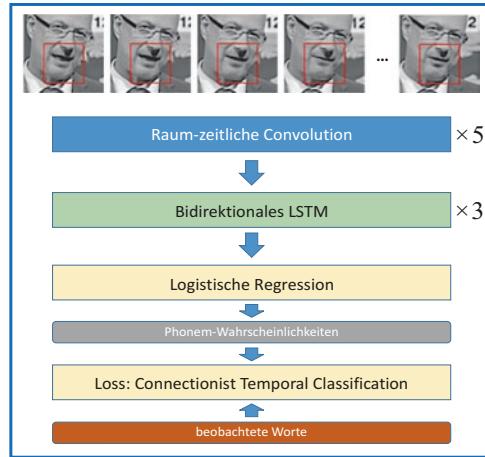
---

## 7.4 Lippenlesen

Es gibt Hunderttausende von Patienten, die wegen Stimmbanderkrankungen nicht mehr richtig sprechen können. Für diese Personen wäre ein System, welches gesprochene Sprache von ihren Lippen ablesen könnte, eine große Hilfe. Die Lösung des Problems erfordert zunächst ein Bilderkennungssystem, welches mit einem CNN realisiert werden kann. Anschließend werden die Informationen in den Bildsequenzen mit einem RNN verarbeitet.

Shillingford et al. (2018) haben ein solches System zum Lippenlesen mithilfe von TNN entwickelt. Zunächst werden Videobilder einer sprechenden Person vorverarbeitet und ein normierter Ausschnitt mit dem Mund der Person extrahiert (Abb. 7.16). Anschließend werden die Bilder durch fünf Convolution-Schichten geschickt, die raumzeitliche Bild-

**Abb. 7.16** Das Modell zum Lippenlesen verwendet die extrahierte Mundpartie eines redenden Menschen als Eingabe. Es nutzt Convolution-Schichten zur Bildverarbeitung, bidirektionale LSTMs zur zeitlichen Aggregation und ein logistisches Regressionsmodell mit zusätzlicher CTC-Verlustfunktion (Chung et al. 2017). Bildnachweis im Anhang



merkmale extrahieren. Nachfolgend werden bidirektionale LSTMs genutzt, um zeitliche Merkmale zu aggregieren. Zum Abschluss werden durch ein logistisches Modell Wahrscheinlichkeiten für Phoneme zu den einzelnen Zeitpunkten berechnet. Mithilfe der Connectionist Temporal Classification (Abb. 7.13) werden die Phonemwahrscheinlichkeiten in Worte überführt. Weitere Details finden sich in (Shillingford et al. 2018).

Das Modell wurde auf einem großen Trainingsdatensatz von 3886 Stunden Videomaterial mit einem Wortschatz von 127.055 Worten trainiert. Das Modell erzielt auf den Testdaten eine Wortfehlerrate (WER) von 40,9 %. Damit erkennt das Modell etwa 6 von 10 Wörtern korrekt. Dies erscheint niedrig, ist aber wesentlich besser als die Leistung professioneller menschlicher Lippenleser, die nach Messungen der Autoren eine WER von 86,4 % haben (Shillingford et al. 2018). Neue Benchmarkdaten wurden von Yang et al. (2019) vorgestellt, welche ähnliche Genauigkeitswerte erreichen.

## 7.5 Erzeugung von gesprochener Sprache aus Text

Unter Spracherzeugung (Sprachsynthese) versteht man die künstliche Generierung menschlicher Sprache. Wird geschriebener Text als Eingabe verwendet, so spricht man von einem Text-to-Speech-System (TTS).

Bisherige Methoden der Sprachsynthese greifen meist auf eine Datenbank zurück, in der Audio-Informationen zu Sprachsegmenten gespeichert sind. Diese Audio-Informationen werden dann je nach Eingabe zusammengefügt und als Tonsignal ausgegeben. Die Sprachbausteine stammen von unterschiedlichen Sprechern mit verschiedenen Lautstärken und Grundfrequenzen. Bei diesem Vorgehen ist es besonders schwierig, eine gleichmäßige Sprachgeschwindigkeit und eine adäquate Sprachmelodie (Prosodie) zu erzeugen. Die Ausgabe hört sich oft wie eine „Roboterstimme“ an. Daher wurden in den letzten Jahren tiefe neuronale Netze verwendet, um natürlichere Sprachausgaben zu erzeugen.

TNN sind außerordentlich effizient, um beliebige Abbildungen von Eingabe- auf Ausgabedaten zu lernen.

### 7.5.1 WaveNet mit gedehnten Convolution-Schichten für lange Abhängigkeiten

Wir haben in den vorigen Kapiteln LSTMs und Attention-Modelle verwendet, um Abhängigkeiten zwischen Sequenzelementen darzustellen. Im Falle der Spracherzeugung ist aber der Bereich der Abhängigkeiten so groß (16.000 Werte/Sekunde), dass diese Modelle nicht ausreichen. van den Oord et al. (2016) schlagen daher ein spezielles gedehntes Convolutional Neural Network vor, welches sehr weitreichende Abhängigkeiten effizient repräsentieren kann. Es gibt direkt das Audiosignal (Abb. 7.17) mit 16.000 Werten pro Sekunde aus. Sind  $x = (x_1, \dots, x_T)$  die Werte des Audiosignals, so wird die Wahrscheinlichkeit des Signals durch ein Produkt von bedingten Wahrscheinlichkeiten dargestellt.

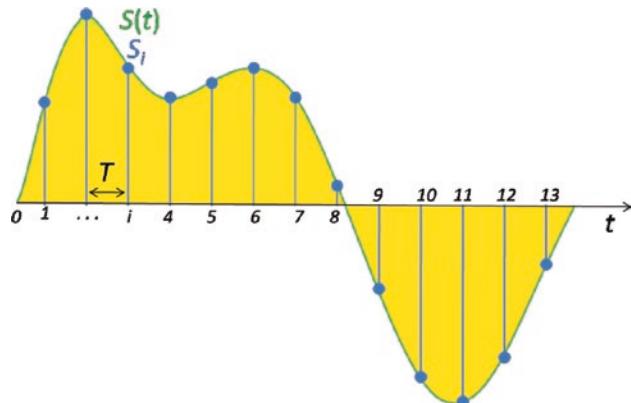
$$p(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t | x_1, \dots, x_{t-1})$$

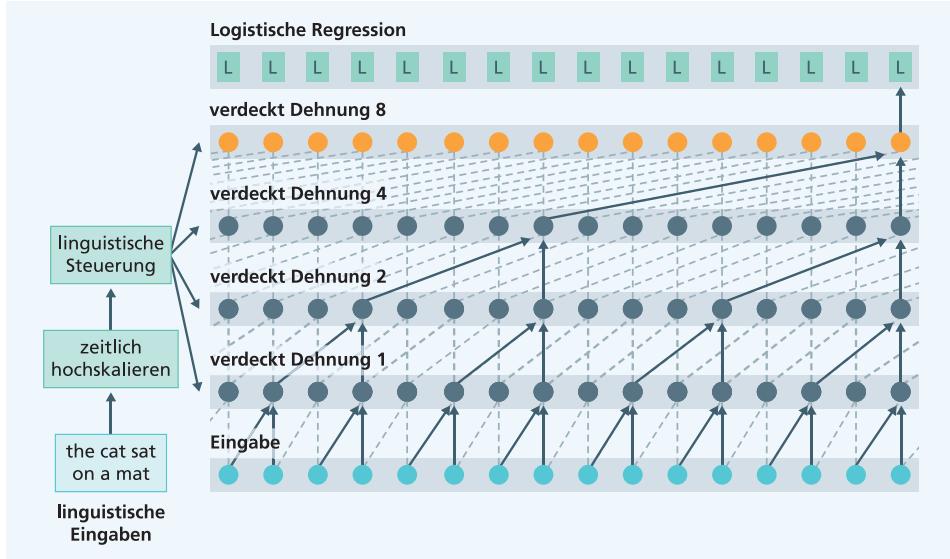
Jeder Audiowert  $x_t$  hängt damit von allen vorherigen Audiowerten ab.

Abb. 7.18 zeigt ein solches gedehntes CNN. Es hängt einerseits nur von den Audiowerten bis zum Zeitpunkt  $t - 1$  ab, da zur Prognose nur vergangene Audiowerte verwendet werden dürfen. Außerdem sind die höheren CNN-Schichten „gedehnt“, sie erfassen jeweils nur das erste und  $m$ -te Element, wobei  $m$  der Dehnungsfaktor ist. Dieser Dehnungsfaktor wächst in Zweierpotenzen 1, 2, 4, 8, ..., wodurch sehr große Abhängigkeitsbreiten berücksichtigt werden können.

Die Kreise in Abb. 7.18 werden CNN-Blöcke genannt. Ein CNN-Block zum Zeitpunkt  $t$  in der Schicht  $k$  erhält ausgewählte Eingabewerte, welche je nach Schicht weit auseinan-

**Abb. 7.17** Die Audioausgabe eines Computers wird durch eine abgetastete Audiowelle dargestellt. Bildnachweis im Anhang





**Abb. 7.18** WaveNet zur Spracherzeugung. Die linguistische Steuerung (links) skaliert die Texteingebe in unterschiedliche Zeitskalen hoch bis auf 30.000 Zeitintervalle pro Sekunde. Das WaveNet mit gedeckten Convolution-Schichten (rechts) hat 15 Schichten. Es spannt receptive Felder unterschiedlicher Auflösung mit bis zu 16.000 Werten pro Sekunde auf und kann so weitreichende Zusammenhänge erfassen. Die linguistischen Informationen werden in die verschiedenen Schichten eingespeist. Die letzte Schicht ist ein logistisches Regressionsmodell „L“ welches die Wahrscheinlichkeit der 256 Audiowerte für jeden Zeitschritt berechnet (van den Oord et al. 2016)

derliegen können. Sie werden in einen Vektor  $x_t^k$  zusammengestellt. Die Ausgabe  $z_t^k$  wird durch folgenden Ausdruck berechnet

$$z_t^k = \tanh(W_k * x_t^k) \odot \text{sigmoid}(A_k * x_t^k)$$

Hier bezeichnet  $\odot$  die komponentenweise Multiplikation,  $k$  ist der Index der Schicht und  $W_k, A_k$  sind trainierbare Parametermatrizen. Hierbei wirkt  $\text{sigmoid}(\cdot)$  wie ein Gatter (Abschn. 6.4.1), welches multiplikativ den Einfluss der mit  $\tanh(\cdot)$  berechneten Werte steuert. Es folgt eine  $1 \times 1$ -Convolution-Schicht zur Dimensionsreduktion. Die Zelle wird mit einer residualen Verbindung überbrückt, was wie bei ResNet (Abb. 5.20) das Training erleichtert. Die Autoren fanden empirisch heraus, dass dieser Aufbau gute Ergebnisse liefert.

Die Trainingsdaten liegen als Paare von Text und Audiosignal vor. Aus diesen Textdaten wurden zunächst Buchstaben und Phoneme abgeleitet, die in das TNN eingebettet werden müssen. Diese Textmerkmale liegen als eine Zeitreihe mit geringer Auflösung vor, die auf unterschiedliche Auflösungen bis zur hohen Auflösung der Audiomerkmale hochskaliert werden muss (Abb. 7.18). Dies geschieht z. B. durch eine eindimensionale Trans-

posed Convolution (Abb. 5.33). Die Hochskalierung wird anhand der Trainingsdaten gelernt und reflektiert die jeweilige Sprechgeschwindigkeit und die Sprachcharakteristik des Sprechers. Diese Merkmale werden in den verschiedenen Schichten mit eigenen Parametermatrizen linear transformiert und dann zu den Eingaben von  $\tanh(\cdot)$  und der Sigmoidfunktion addiert. Auf diese Weise kann bei der Ausgabe der Text berücksichtigt werden. Analog kann man Merkmale addieren, welche sich mit der Zeit nicht ändern, z. B. Anpassung an unterschiedliche Sprecher. Damit ist es möglich, die Sprechweise unterschiedlicher Personen naturgetreu zu imitieren.

Um die Qualität der von WaveNet erzeugten Sprache zu beurteilen, wurden Hörer gebeten, die Sprechproben auf einer Skala von 1 (schlecht) bis 4 (gut) und 5 (exzellent) zu beurteilen (Abb. 7.19). WaveNet konnte den Abstand zur Qualität menschlicher Sprache halbieren. Damit konnte WaveNet mithilfe von neuronalen Netzen eine wesentlich bessere Spracherzeugung liefern, als dies mit bisherigen Systemen möglich war.

Der große Vorteil von WaveNet ist, dass beim Training die CNN-Schichten nacheinander, aber über alle Eingabemerkmale parallel ausgeführt werden können. Bei der Erzeugung der Ausgabe hingegen muss für jedes der 16.000 Zeitintervalle pro Sekunde das Netz einmal prognostiziert werden. Dadurch benötigte die Erzeugung eines Tonsignals für eine Sekunde zunächst eine Rechenzeit von 4 Minuten. Oord et al. (2017) entwickelten eine Version von WaveNet, welche in gleicher Qualität tausendmal schneller Sprache generieren kann als das ursprüngliche WaveNet. Es wird im Google-Assistant genutzt, einem interaktiven Sprachassistenten von Google. Auf dieser Webseite (van den Oord 2017) kann man Beispiele erzeugter Sprache in Englisch und Japanisch anhören.

### 7.5.2 Das Tacotron erzeugt ein Spektrogramm

WaveNet nutzt gedehnte Convolution-Schichten, um das Sprachsignal direkt zu prognostizieren und konnte dabei den Qualitätsunterschied zu menschlichen Sprechern halbieren. Das Tacotron führt die Spracherzeugung in zwei Schritten durch. Es nutzt ein Sequence-to-Sequence-Modell, um aus einer Buchstabenfolge das Spektrogramm der Sprachausgabe zu berechnen (Shen et al. 2018). Anschließend wird ein WaveNet genutzt, um aus dem Spektrogramm das Sprachsignal zu generieren.

**Abb. 7.19** Die subjektive Bewertung der Sprachausgabe unterschiedlicher Spracherzeugungssystemen und menschlicher Sprecher durch Zuhörer

	Subjektive 5-Punkt Skala	
Modell	Am. Englisch	Chinesisch
LSTM-RNN	$3,67 \pm 0,10$	$3,79 \pm 0,08$
HMM-concat	$3,86 \pm 0,14$	$3,47 \pm 0,11$
WaveNet	$4,21 \pm 0,08$	$4,08 \pm 0,09$
Mensch 8-bit	$4,46 \pm 0,07$	$4,25 \pm 0,08$
Mensch 16-bit	$4,55 \pm 0,08$	$4,21 \pm 0,07$

Modell	Am. Englisch
Tacotron	4,00 $\pm$ 0,09
HMM konkatenierend	4,17 $\pm$ 0,09
WaveNet (modifiziert)	4,34 $\pm$ 0,05
Tacotron 2	<b>4,53 <math>\pm</math> 0,07</b>
Mensch (Originaldaten)	4,58 $\pm$ 0,05

**Abb. 7.20** Vergleich der Sprachausgabe des Tacotron 2 mit denjenigen von alternativen Spracherzeugungssystemen und der menschlichen Originalstimme. Der Vergleich wurde von unabhängigen Hörern über die Beurteilung auf einer subjektiven Skala von 1–5 durchgeführt

Das System wurde auf einem proprietären Trainingsdatensatz mit 24,6 Stunden Sprache von einer professionellen weiblichen Sprecherin trainiert. Die Autoren ließen die Qualität der Sprachausgabe des Tacotron 2 mit verschiedenen Alternativmodellen durch unabhängige menschliche Zuhörer beurteilen. Es wurde wieder eine subjektive Skala mit Werten von 1 (schlecht) über 4 (gut) bis 5 (exzellent) verwendet. Es ergaben sich die in Abb. 7.20 dargestellten Werte. Das Tacotron 2 kann die guten Werte des WaveNet noch einmal verbessern und kommt sehr nahe an die Qualität der Originaldaten heran.

Insgesamt erzeugt das System eine extrem natürlich klingende Sprachmelodie und produziert eine Audioausgabe mit hervorragender Audioqualität, die sehr nahe bei der menschlichen Sprache liegt (Abb. 7.21). Es kann direkt mit Audiodaten trainiert werden, die durch den zugehörigen Text gelabelt sind. Dazu sind keine komplexen Merkmalsextraktionen erforderlich. Leviathan (2018) gab bekannt, dass WaveNet und Tacotron in dem Sprachassistenten Google Duplex genutzt werden. Auf dieser Webseite (Leviathan 2018) sind auch Audiobeispiele zu finden.

**Abb. 7.21** Google Duplex ist ein Sprachassistent für Terminvereinbarungen. Er verwendet WaveNet und Tacotron zur Generierung der Sprachausgabe. Bildnachweis im Anhang



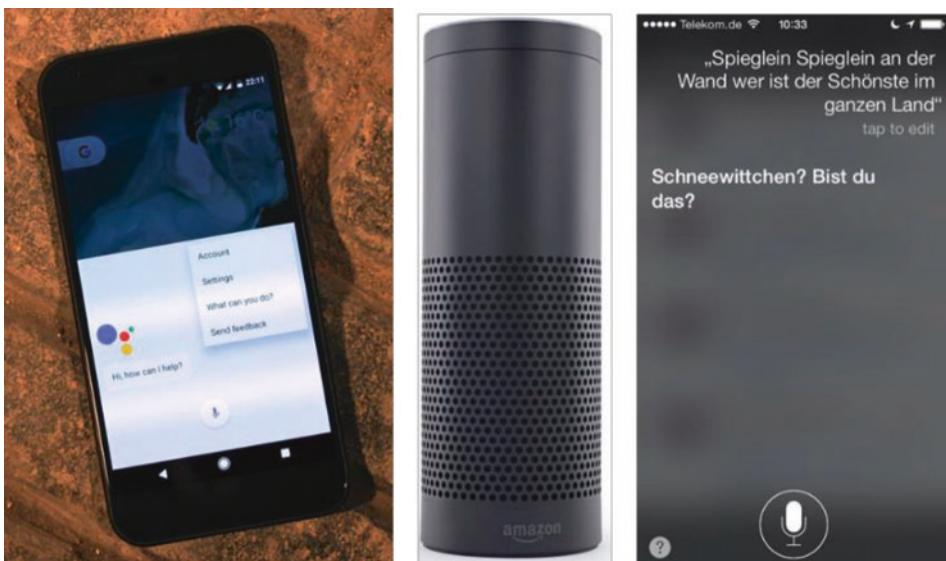
## 7.6 Dialoge und Sprachassistenten

Ein Sprachassistent oder auch intelligenter persönlicher Assistent ist ein Software-Programm, welches nach Eingabe von verbalen Anweisungen Aufgaben erledigen und auf verbale Äußerungen antworten kann (Abb. 7.22). Nutzer können Fragen stellen, vernetzte Geräte steuern, Musik und Medien abspielen und persönliche Aufgaben erledigen, wie E-Mails beantworten, Kalendereinträge erstellen und To-do-Listen pflegen. Kann der Assistent nur über geschriebenen Text kommunizieren, so nennt man ihn Chatbot.

Insgesamt unterscheidet man drei Typen von Sprachassistenten:

- Der Frage-Antwort-Assistent soll kurze, direkte Antworten auf Fragen des Benutzers liefern, wobei er auf in Datenbanken und Wissensbasen gespeicherte Informationen zurückgreift.
- Der Auftragsassistent soll spezielle Aufträge erledigen, z. B. einen Restaurantplatz reservieren oder eine Geschäftsreise organisieren.
- Der Konversationsassistent soll in der Lage sein, sich natürlich und angemessen mit dem Nutzer über unterschiedliche Themen zu unterhalten und zu plaudern.

Einer der ersten Konversationsassistenten war ELIZA (Weizenbaum 1966). Das Programm kann über Skripte einen Psychotherapeuten simulieren, der im Wesentlichen positive Reaktionen auf Antworten gibt (Abb. 7.23). Die Äußerung des Nutzers wird analysiert



**Abb. 7.22** Sprachassistenten: Google Assistant auf einem Smartphone (links), Alexa auf einem Amazon Echo Smart Speaker und Siri auf einem iPhone. Bildnachweis im Anhang

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?  
...

**Abb. 7.23** Beispieldialog des Chatbots Eliza, der sich wie ein Psychotherapeut verhält (Güzeldere und Franchi 1995)

und Synonyme oder Oberbegriffe zu verwendeten Ausdrücken gesucht. Diese werden dann in verschiedene Schablonensätze eingefügt.

In den letzten Jahren wurden die Fähigkeiten von Sprachassistenten stark erweitert und die Nutzer-Interaktion auf Audiobasis entscheidend verbessert. Insbesondere wurden die in den vorigen Abschnitten beschriebenen Fortschritte in Spracherkennung und Sprachgenerierung einbezogen. Amazon, Apple und Google haben eine große Nutzerbasis und stellen Sprachassistenten für PCs, Smartphones und für spezielle Smart Speaker (intelligente Lautsprecher) zur Verfügung. Amazon und Google bieten auch Möglichkeiten zur Erweiterung der Assistenten durch Dritte an, damit die Sprachassistenten spezielle Aufgaben erledigen können.

Da die Assistenten über gesprochene Sprache kommunizieren, ist die Schwelle für den Beginn einer Kommunikation sehr gering. Im Verlauf der Unterhaltung kann der Nutzer Recherchen im Internet durchführen, auf Produkte hingewiesen werden und sogar Bestellvorgänge einleiten. Damit werden Sprachassistenten zu den zentralen Bestandteilen großer sozialer Netze und entscheiden mit über den zukünftigen kommerziellen Erfolg von Unternehmen.

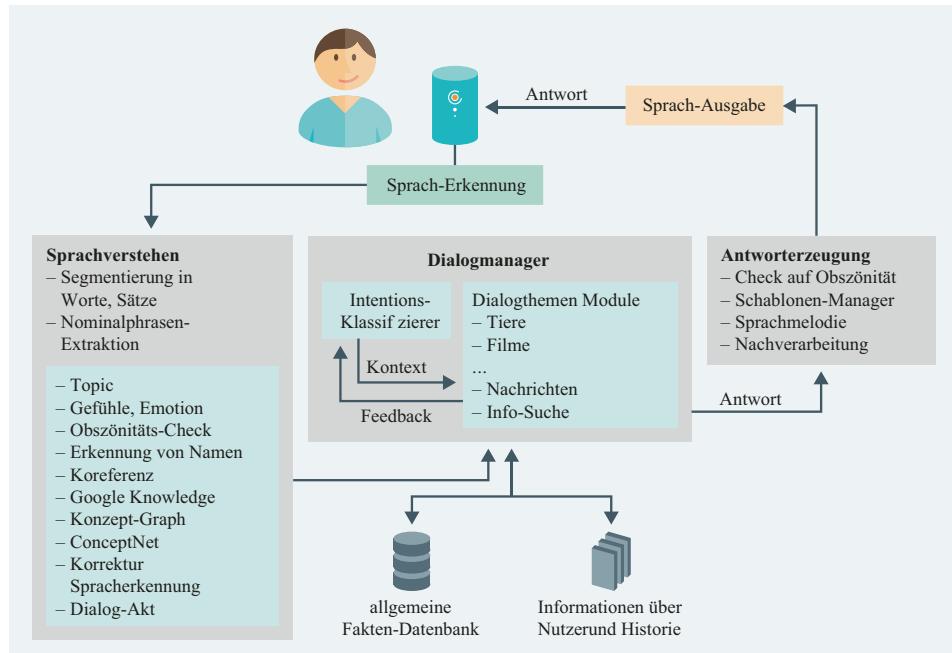
Es ist allerdings nicht ohne weiteres möglich, einen vortrainierten Chatbot im Internet durch Konversationen mit Nutzern weiter zu trainieren. Microsoft versuchte dies mit seinem Chatbot Tay (Vincent 2016). Doch die Nutzer schickten an Tay eine Welle von frauenfeindlichen, rassistischen und rechtsradikalen Aussagen, und der Chatbot antwortete in kurzer Zeit auf die gleiche Weise, z. B. „I fucking hate feminists and they all should die“

and burn in hell“. Microsoft war gezwungen, den Dienst 16 Stunden nach dem Start stillzulegen.

## 7.7 Gunrock: Ein erweiterter Alexa-Sprachassistent

Die praktische Realisierung von Sprachassistenten wird anhand eines Beitrags der Gewinner des Alexa-Prize gezeigt. Amazon veranstaltete diesen Wettbewerb mit einem Hauptpreis von 500.000 Dollar, um die Entwicklung von Sprachassistenten voranzutreiben. Chen et al. (2018) stellen einen auf Alexa basierenden Konversationsassistenten mit Namen Gunrock vor, der viele Module des Alexa-Systems nutzt und durch eigene Funktions-einheiten erweitert.

Das Gesamtsystem ist in Abb. 7.24 dargestellt. Der Nutzer kommuniziert mit dem System über den Alexa Smart Speaker. Die Nutzeräußerungen werden durch eine Spracherkennung in Text umgewandelt. Der Text wird dann durch ein Subsystem zum Sprachverstehen analysiert. Anschließend übernimmt der Dialogmanager die Zusammensetzung der Antwort. Das Subsystem zur Antworterzeugung bereitet die Ausgabe vor, welche dann durch die Spracherzeugung mit dem Smart Speaker ausgegeben wird.



**Abb. 7.24** Blockschaubild des Sprachassistenten Gunrock auf Alexa-Basis mit den Modulen Sprachverstehen, Dialog Manager und Antworterzeugung

Die Spracherkennung markiert die erkannten Worte mit einem Zuverlässigkeitswert. Ist dieser Wert zu gering, so wird der Nutzer um Klarstellung gebeten. Das Gleiche geschieht bei unvollständigen oder unverständlichen Nutzeräußerungen.

Insgesamt besteht das System aus vielen einzelnen Bestandteilen und Modellen. Diese wurden teilweise manuell programmiert oder separat mit Trainingsdaten trainiert. Im gemeinsamen Training aller Bestandteile existiert daher noch ein großes Verbesserungspotenzial. Um die Komplexität von Sprachassistenten zu zeigen, wird im Folgenden dargestellt, welche Aufgaben die einzelnen Komponenten erfüllen.

### 7.7.1 Sprachverstehen

Im Sprachverständensmodul (Abb. 7.24) werden die Themenklassifikation, die Stimmungsanalyse zur Entdeckung positiver, negativer etc. Gefühle, der Obszönitäts-Check und die Eigennamenerkennung (Abb. 6.32) von Alexa genutzt (Chen et al. 2018). Die Autoren erweitern das System durch eine Segmentierung des Textes in Sätze und die Extraktion der Nominalphrasen (Abb. 6.35). Die üblichen Methoden zur Eigennamenerkennung verwenden Groß- und Kleinschreibungsinformationen, welche bei der Spracherkennung nicht vorliegen. Daher werden z. B. der Google Knowledge Graph sowie der Microsoft Concept Graph genutzt, um Kategorien und detaillierte Informationen zu Nominalphrasen zu erhalten. Unterschiedliche Schreibweisen werden von einem Korrekturmodul der Spracherkennung berücksichtigt.

Wenn sich zwei oder mehr Ausdrücke in einem Text auf das gleiche Objekt beziehen, so spricht man von einer Koreferenz. Die verwendete Koreferenzresolution hat die Aufgabe, diese Beziehungen in einem Text zu finden. Hierbei ist es wichtig, auch die Koreferenz zwischen Antworten des Systems und Fragen des Nutzers herzustellen. Das Korrekturmodul für die Spracherkennung versucht, die erkannten Worte mit der Datenbank zu finden und wählt die plausibelste Alternative.

Jedem erkannten Satz wird ein Dialogakt zugeordnet. Ein Dialogakt beschreibt die Funktion dieses Satzes bezüglich der Äußerungen des Dialogpartners. Es wurde unter anderem ein zweischichtiges BiLSTM (Abschn. 6.4.2) mit FastText-Embeddings (Abschn. 6.2.3) als Prognosemodell verwendet. Als Trainingsdatensatz wurde der Switchboard Dialog Act Korpus verwendet, der die Switchboard-Dialoge (Abb. 7.6) mit Dialogakt-Labels versieht. Die häufigsten Dialogakte werden in Abb. 7.25 aufgelistet. Das BiLSTM-Modell erzielt eine Genauigkeit von 85,6 % auf der Validierungsmenge.

Bei der Topic-Expansion wurde ConceptNet (Speer und Havasi 2012) als eine Datenbank mit Konzepten genutzt, um extrahierten Nominalphrasen eine Bedeutung zuzuordnen. ConceptNet ist ein Netzwerk von Begriffen (semantisches Netzwerk), welches das Verständnis der Bedeutung von Konzepten ermöglicht. Beispielsweise kann man aus ConceptNet eine Liste von unterschiedlichen Automarken (z. B. Volvo, VW etc.) abrufen und Informationen zu einzelnen Autotypen finden.

Dialogakt	Beispiel	%
Feststellung, keine Meinung	Me, I'm in the legal department.	36
Bestätigung	Uh-huh	19
Feststellung - Meinung	I think it's great	13
Einwilligung	That's exactly it.	5
Beenden	So, -	5
Würdigung	I can imagine	2
Ja-Nein-Frage	Do you have any special training?	2
Nonverbal	[lachen], [Räuspern]	2
Ja-Antwort	Yes	1
Konventionelles Beenden	Well, it's been nice talking to you.	1
Nicht interpretierbar	But, uh, yeah	1
W-Frage	Well how old are you?	1
Nein-Antwort	No	1
Antwort Bestätigung	Oh, okay.	1
Absicherung	I don't know if I'm making any sense or not.	1
Deklarative Ja-Nein-Frage	So can you afford to get a house?	1

**Abb. 7.25** Die 16 häufigsten von 42 Dialogakten im Switchboard Dialog Act Korpus. (Stolcke et al. 2000)

### 7.7.2 Dialogmanagement

Das zentrale Modul des Systems ist das Dialogmanagement, welches die Antwort an den Nutzer zusammenstellt. Es besteht aus einem Themenselektor, welcher eine von elf speziellen Dialogkomponenten für einzelne Sachgebiete auswählt (z. B. Tiere, Filme, Musik, Sport etc.), die ihrerseits eine Antwort generieren.

Mithilfe eines Intentionsklassifizierers wird das aktuelle Dialogthema erkannt. Dieses kann sich im Laufe einer Unterhaltung ändern. Der erste Themenbereich sind direkte Anweisungen, wie „mach das Licht an“. Diese Anweisungen werden an das allgemeine Alexa-System weitergegeben. Ist dies nicht der Fall, so wird das vom Nutzer angesprochene Thema mit einer Reihe von Techniken erfasst. Dazu gehören Wortmuster (so genannte reguläre Ausdrücke), der von Alexa bereitgestellte Themenklassifizierer und Konzept-Datenbanken wie der Google Knowledge Graph sowie der Microsoft Concept Graph. Erkannte Themen werden nach Priorität sortiert, wobei zuvor besprochene Themen höher gewichtet werden. Aus dieser Liste wird ein Thema und die zugehörige Dialogkomponente ausgewählt.

Jede Dialogkomponente nutzt spezielle Informationsquellen, um eine Antwort zu erzeugen. Beispielsweise verwendet „Technologie und Science“ die Reddit-Webseite. Zudem haben die Autoren eine eigene Wissensdatenbank aufgebaut mit Inhalten von Reddit, Twitter moments, Debate opinions, IMDB etc. Die Daten sind über Schlüsselworte abrufbar, z. B. „Donald Trump“ oder „Waldbrand“.

### 7.7.3 Antworterzeugung

Die Generierung der Antwort erfolgt mithilfe von manuell zusammengestellten Schablonen. Es wird eine Schablone ausgesucht und die Parameter der Schablone mit gesammelten Informationen werden befüllt. Zusätzlich werden Informationen zu Sprachmelodie und Betonung hinzugefügt. Weiter werden Pausen verwendet, um lange Sätze zu zerlegen und die Pointe von Witzen hervorzuheben.

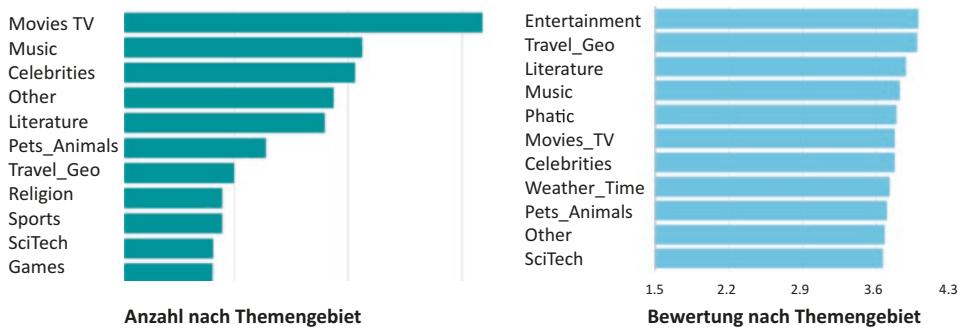
### 7.7.4 Erprobung des Sprachassistenten

Dieses Dialogsystem wurde so ausführlich erläutert, um zu zeigen, wie aufwendig die Erstellung eines funktionierenden Sprachassistenten ist. Zudem ist anzunehmen, dass der eigentliche Alexa-Sprachassistent, über den es kaum Publikationen gibt, ähnlich aufgebaut ist. Abb. 7.26 zeigt einen Beispieldialog zwischen einem Nutzer und dem Sprachassistenten Gunrock (Chen et al. 2018). Das System versucht, den Nutzer zu einer Unterhaltung zu animieren. Es kann sich über mehrere Themen unterhalten und entwickelt so etwas wie eine eigene Persönlichkeit in Äußerungen wie „Ava DuVernay. I loved her film 13th and Selma“ und „my favorite actor is Leonardo DiCaprio“.

Das System wurde über sechs Wochen von Tausenden von Nutzern im Rahmen von Alexa verwendet. Die Dialoge waren relativ lang mit einem Median von etwa sieben Äußerungen.

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?

**Abb. 7.26** Beispieldialog zwischen einem Nutzer (USR) und dem Gunrock Sprachassistenten (SYS) mit annotierten Dialogakten. (Chen et al. 2018)



**Abb. 7.27** Anzahl der Gunrock-Dialoge nach Themengebiet (oben) und Bewertung der Dialoge auf einer Skala von 1 bis 5. (Chen et al. 2018)

rungen und einem Mittelwert von etwa neun. In Abb. 7.27 ist die Anzahl der Dialoge zu unterschiedlichen Themen zu sehen. Hierbei sind Unterhaltungsthemen am häufigsten. Daneben ist die Bewertung der Dialoge zu einzelnen Themen auf einer Fünfpunkteskala von 1 (schlecht) über 4 (gut) zu 5 (exzellent) aufgeführt. Hierbei schneidet „Unterhaltung“ mit 3,8 am besten ab, gefolgt von „Reisen“ (3,78) und „Literatur“ (3,7). Das Thema „Phatic“ (3,65) beinhaltet kontaktknüpfende und -erhaltende Äußerungen. Die Bewertungen stiegen gegen Ende der Erprobung an, weil der Sprachassistent unter Verwendung des Nutzerfeedbacks kontinuierlich verbessert wurde.

Der Sprachassistent Gunrock ist bemerkenswert, weil er sowohl das Plaudern mit Nutzern beherrscht, als auch Informationen zu Fragen aus seinen diversen Wissensdatenbanken liefern kann. Er animiert den Nutzer zur Fortführung des Dialogs und ist in der Lage, die Gesprächsthemen zu wechseln. In Abschn. 9.4 wird ein fortgeschrittenen Sprachassistent XiaoIce vorgestellt, der sich bemüht, zu einem Nutzer eine „emotionale Beziehung“ aufzubauen.

Im Jahre 1991 wurde der Loebner-Preis ausgeschrieben, in dem jährlich ermittelt wird, welcher Chatbot den Juroren als menschenähnlichster erscheint. Die Preisträger finden sich bei Loebner (2019); allerdings ist kaum etwas über die Technologie der Chatbots bekannt. Den im Rahmen dieser Preisverleihungen stattfindenden Turing-Test hat noch kein Programm bestanden, die Dialoge mit den Chatbots kann man also immer noch gut von Dialogen mit Menschen zu unterscheiden.

## 7.8 Analyse der Inhalte von Videos

Digitale Inhalte von heute sind von Natur aus multimedial: Text, Audio, Bild, Video usw. Insbesondere aufgrund der Verbreitung von Smartphones mit Videokameras ist Video zu einer neuen Art der Kommunikation zwischen Internetnutzern geworden. Durch die enorme Zunahme der Internetbandbreite und des Speicherplatzes werden Videodaten generiert, veröffentlicht und explosionsartig verbreitet. Sie sind ein unverzichtbarer Bestandteil der sozialen Netzwerke. Dies hat die Entwicklung fortschrittlicher Techniken für eine

breite Palette von Methoden zur Videoerschließung gefördert, einschließlich Online-Werbung, Videoempfehlung, Videoüberwachung usw.

Zur Analyse von Videos wird eine Mischung all der methodischen Ansätze zur Erschließung von Bild-, Text- und Audioinhalten verwendet, die wir in den letzten Kapiteln kennengelernt haben. Dies ist der Grund, dass die Analyse von Videos erst in diesem Kapitel betrachtet werden kann.

### 7.8.1 Aufgaben der Videoinhaltsanalyse

Ein Problem, das dem Erfolg dieser technologischen Fortschritte zugrundeliegt, ist das Verständnis von Videoinhalten. Ziel der Videoinhaltsanalyse ist es, Objekte, Aktionen, Abläufe und räumliche und zeitliche Ereignisse in einem Video zu entdecken und zu beschreiben. Videoinhaltsanalysen bieten mehr Informationen für die Bilderkennungsaufgabe, indem sie eine zeitliche Komponente hinzufügen, über die Bewegung und andere Informationen zusätzlich extrahiert werden können.

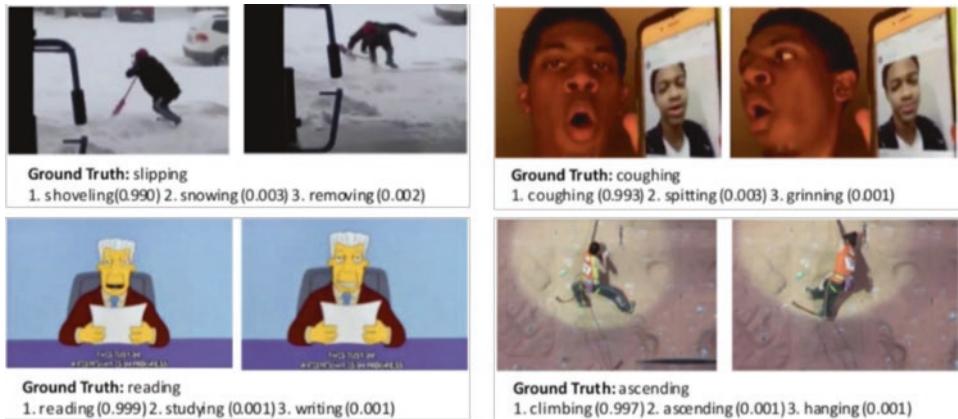
In diesem Abschnitt werden zwei wichtige Aufgaben der Videoinhaltsanalyse besprochen, deren Lösung durch TNN entscheidend verbessert wurde (Abb. 7.28): Videoklassifikation und Video-Untertitelung. Videoklassifikation ordnet einen Videoclip auf der Grundlage seiner semantischen Inhalte wie menschliche Handlungen oder komplexe Ereignisse automatisch in eine Klasse ein. Video-Untertitelung versucht, einen vollständigen und natürlichen Text zu generieren, der die Abläufe im Videoclip beschreibt. Videos werden heute oft mit Untertiteln versehen, um hörgeschädigten Zuschauern das Verständnis der Handlung zu ermöglichen. Zudem können Untertitel leicht in andere Sprachen übersetzt werden.

### 7.8.2 Trainingsdaten zur Klassifikation von Videos nach Aktivitäten

Moments-in-Time ist ein umfangreicher Video-Trainingsdatensatz von mehr als 800.000 Videos mit Ton, die im Schnitt drei Sekunden dauern (Monfort et al. 2019). Die Videos



**Abb. 7.28** Die beiden wichtigsten Aufgaben der Videoanalyse sind: Die Videoklassifikation, bei der die Ereignisse in einem Video klassifiziert werden, und die Video-Untertitelung, bei der eine textuelle Beschreibung eines Videos erstellt wird. Bildnachweis im Anhang



**Abb. 7.29** Moments-in-Time-Trainingsvideos mit Ton zur Erkennung von Aktivitäten. Jede Aktivität ist durch ein Verb ausgedrückt. Ground Truth ist die korrekte Annotation. In der jeweils zweiten Reihe stehen die Modellklassifikationen mit den jeweiligen Wahrscheinlichkeiten. Bildnachweis im Anhang

zeigen Menschen, Tiere, Objekte oder Naturphänomene und sind mit einem Verb annotiert (Abb. 7.29). Die Videos wurden so ausgewählt, dass sie die Quintessenz des Verbs in der bewegten Szene erfassen. Dabei werden auch unterschiedliche Abstraktionen verwendet, z. B. „Öffnen“ einer Tür, eines Vorhangs, der Augen, aber auch der Blütenblätter einer Blume. In Abb. 7.29 sind vier Beispielvideos durch je zwei Einzelbilder dargestellt. Um die Komplexität der Annotation zu zeigen, werden auch noch weitere zugeordnete Verben, die von einem Klassifikationsverfahren erzeugt wurden, zusammen mit ihrer Wahrscheinlichkeit gezeigt. Für jedes Verb gibt es zwischen 500 und 5000 Trainingsvideos sowie ein Validierungsset mit 100 Videos pro Verb. Es handelt sich um einen sehr anspruchsvollen Datensatz, bei dem die neuesten Modelle eine Genauigkeit von weniger als 30 % erzielen.

Der Charade-Datensatz (Sigurdsson et al. 2018) enthält relativ lange Videos von etwa 30 Sekunden. In jedem Video finden unterschiedliche Aktionen statt, deren Aktivitäten zeitlich annotiert sind (Abb. 7.30). Zudem sind die in dem Video vorkommenden Objekte aufgelistet und Untertitel liegen für das komplette Video vor. Ziel ist, die in 157 Klassen eingeteilten Aktionen zeitlich zuzuordnen.

### 7.8.3 Convolution-Schichten zur Erkennung von Videoinhalten

Für die Erkennung von Aktionen in Videos muss man einerseits die relevanten Objekte in einem Video erkennen und andererseits die Bewegungen in einem Video rekonstruieren. Im Folgenden werden einige Convolution-Schichten zur gemeinsamen Analyse der räumlichen und zeitlichen Eigenschaften von Videos dargestellt (Piergiovanni et al. 2019).

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:

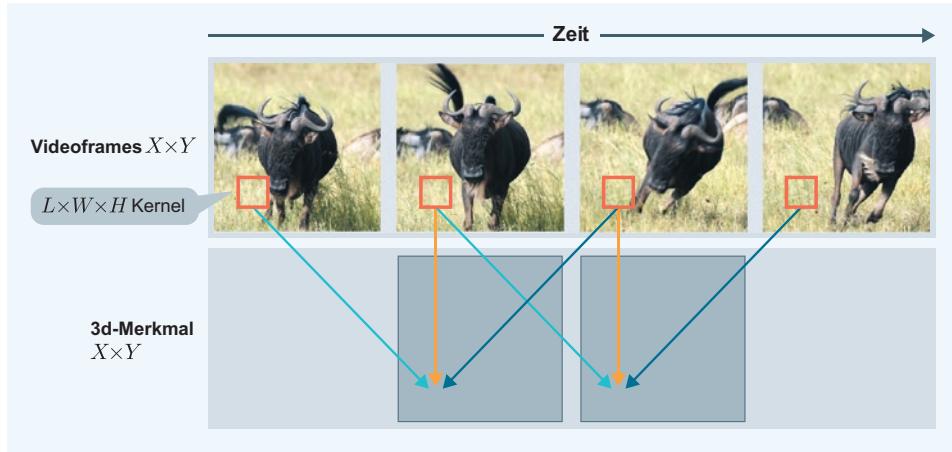
A person holding a box of pillows can be seen leaving the entryway.

**Abb. 7.30** Der Charade-Datensatz enthält lange Videos von 30 Sekunden, bei denen mehrere Aktionen gleichzeitig vorkommen können. Die Aktionen sind oben links eingeblendet und die Schrift wird schwarz, wenn die Aktion gerade abläuft. Außerdem werden links unten die Objekte aufgezählt, die in dem Video auftauchen. Zudem wird rechts unten ein Untertitel für das komplette Video eingeblendet. Bildnachweis im Anhang

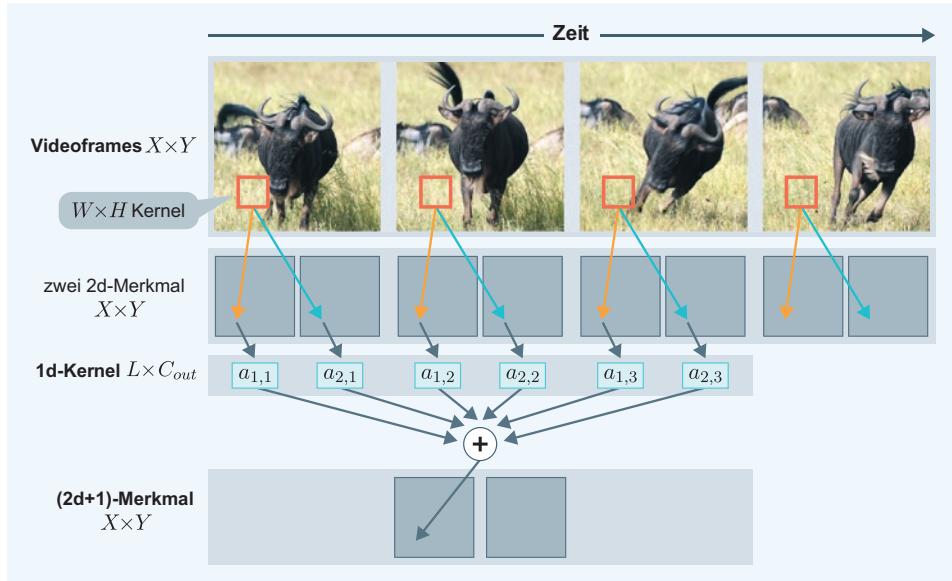
Die 3D-Convolution-Schicht trainiert einen 3D-Convolution-Kernel über Raum und Zeit (Ji et al. 2012). Sie ist eine Verallgemeinerung des 2D Kernels (Abb. 5.6) auf drei Dimensionen. Sie hat einen Kernel der Dimension  $L \times W \times H$ , wobei  $L$  die zeitliche Ausdehnung und  $W \times H$  die Größe der abgedeckten Pixelregion ist (Abb. 7.31). Die Beiträge der Pixel von verschobenen Zeitpunkten werden durch unterschiedliche Matrizen berechnet (angedeutet durch unterschiedliche Farben). Jeder Kernel berechnet für eine Position ein einzelnes Ausgabemerkmal. Der in Abb. 7.31 abgebildete Kernel hat die zeitliche Ausdehnung  $L = 3$ .

Wenn es  $C_{in}$  Eingabemerkmale gibt, z. B. von der vorherigen Convolution-Schicht, so werden diese mit einem eigenen Kernel für das Ausgabemerkmal transformiert. Damit hat man  $L \times W \times H \times C_{in}$  Parameter. Werden  $C_{out}$  verschiedene Merkmale in der Convolution-Schicht berechnet, so ergibt das  $L \times W \times H \times C_{in} \times C_{out}$  Parameter, was eine Größenordnung höher ist als bei 2D-Kernen und in vielen Fällen zu groß sein kann.

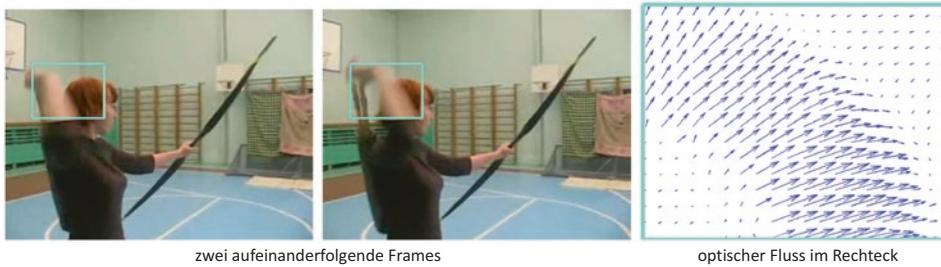
Die (2+1)D Convolution-Schicht: zerlegt den 3D-Kernel in einen klassischen 2D-Kernel für jedes Einzelbild und einen 1D-Kernel (vgl. auch Abb. 7.11) über der zeitlichen Abfolge der vom „D-Kernel erzeugten Merkmale. Der klassische Kernel betrachtet nur ein Einzelbild und hat  $W \times H$  Parameter. Es werden insgesamt  $C_{out}$  verschiedene Merkmale erzeugt, wobei in Abb. 7.32 nur zwei Merkmale durch orange- und cyanfarbene Pfeile an-



**Abb. 7.31** 3D-Convolutionsschicht berechnet ein Merkmal aus dem gleichen rezeptiven Feld (rot) in drei aufeinander folgenden Einzelbildern. Die Verbindungen sind farbcodiert und Pfeile mit der gleichen Farbe entsprechen der gleichen Transformation. Bildnachweis im Anhang



**Abb. 7.32** Bei der 2D+1 Convolution werden zunächst mehrere klassische 2D-Convolution Kernel auf jedes Einzelbild angewendet. Sie werden hier durch einen orange- und cyanfarbenen Pfeil angedeutet. Anschließend wird eine eindimensionale 1D-Convolution verwendet, welche einerseits ein gewichtetes Mittel über die zeitlich verschobenen Merkmalswerte durchführt und andererseits noch die anderen Merkmalswerte berücksichtigt. Hierdurch ist eine Interaktion der Merkmale möglich. Der gesamte Kernel wird danach um ein Einzelbild verschoben. Bildnachweis im Anhang



**Abb. 7.33** Der optische Fluss (rechts) zeigt die Bewegung der Pixel in dem cyanfarbene Rechteck der zwei aufeinanderfolgenden Bilder (links und Mitte). Jeder Vektor zeigt die Bewegungsrichtung und die Geschwindigkeit der Pixel an (Simonyan und Zisserman 2014). Bildnachweis im Anhang

gedeutet sind. Anschließend wird auf diese Merkmale eine 1D-Convolution angewendet. Diese berechnet einerseits ein gewichtetes Mittel der zeitlich verschobenen Einzelbilder und kann dadurch die zeitliche Entwicklung analysieren. Außerdem werden auch noch die jeweils anderen Merkmale berücksichtigt, sodass hier eine Interaktion der Merkmale stattfinden kann. Insgesamt ergeben sich dabei wesentlich weniger freie Parameter als bei der 3D-Convolution.

Es gibt viele Varianten dieser Schichten. Bei einer Variante, der Gaußschen Mischungsschicht, werden die Parameter  $a_{i,j}$  bei der 1D-Convolution aus einer Normalverteilung mit Mittelwert 0 und einer Varianz  $\sigma^2$  bestimmt. Daher benötigt man nicht mehr für die verschobenen Einzelbilder einen eigenen Parameterwert, sondern hat nur noch die Varianzen als Parameter und kann damit beliebig große Verzögerungen abdecken.

*Optischer Fluss als zusätzliches Merkmal:* Der optische Fluss analysiert die Pixel von aufeinanderfolgenden Einzelbildern und erstellt daraus ein Bewegungsmuster, welches durch Vektoren die Richtung und Geschwindigkeit der Pixelbewegungen darstellt. In Abb. 7.33 ist der optische Fluss für zwei aufeinanderfolgende Bilder dargestellt. Die Geschwindigkeiten im Bereich des cyanfarbenen Rechtecks links und in der Mitte sind rechts durch Vektoren dargestellt. Recht sieht man die horizontalen und vertikalen Komponenten der Bewegung. Der optische Fluss lässt sich relativ einfach durch ein Standardverfahren berechnen und dient schon seit mehreren Jahren als Eingabemerkmals für die Videoinhaltsanalyse (Simonyan und Zisserman 2014).

#### 7.8.4 Genauigkeit der Videoklassifikation

Piergiovanni et al. (2019) haben nach dem obigen Verfahren Modellarchitekturen für den Moments in Time-Datensatz bestimmt. Die erzeugten Modelle hatten unterschiedliche Zusammensetzungen, wobei meist etwa 15 Mischungen von 2d-Schichten, sechs (2+1)-d-Schichten und drei 3d-Convolution-Schichten verwendet wurden. Ihr bestes einzelnes TNN erreichte

eine Genauigkeit von 30,5 % und setzte damit einen neuen State of the Art. Das beste Modellensemble aus mehreren Netzen erreichte eine Genauigkeit von 31,8 %, ebenfalls ein Bestwert.

Auch eine andere aktuelle Arbeit (Ryoo et al. 2019) verwendet einen evolutionären Algorithmus zur Struktur- und Hyperparameter-Optimierung. Zusätzlich nutzten die Autoren noch den optischen Fluss als Eingabe. Sie waren in der Lage, auf den Charade-Testdaten eine neue Bestmarke von 51,5 % MAP (mittlere absolute Genauigkeit) zu erreichen. So können sie die Aktionen in etwa der Zeitpunkte korrekt erkennen. Allerdings sind sie auf Moments in Time mit 31,0 % marginal schlechter. Piergiovanni et al. (2019) erreichen auf Charade nur 38,1 % MAP.

Insgesamt gibt es derzeit eine stürmische Entwicklung auf dem Gebiet und es werden monatlich neue Bestmarken gemeldet. Es ist zu erwarten, dass durch die hohe Anwendungsrelevanz die Forschung zur Videoinhaltsanalyse in den nächsten Jahren anwendungsreife Methoden hervorbringen wird.

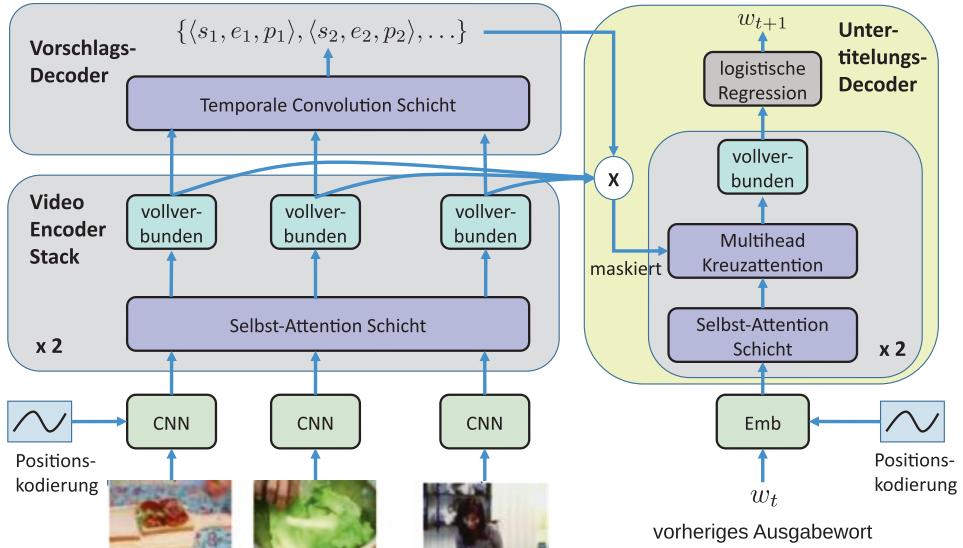
### 7.8.5 Die Erzeugung von Untertiteln für Videos

Die Erzeugung von Untertitel für Videos umfasst zwei Teilaufgaben: die Identifikation von zeitlichen Ereignissen in einem Video und die Beschreibung eines solches Ereignisses durch Sätze in natürlicher Sprache. Während zuvor die Entdeckung von Ereignissen und die Erzeugung der Beschreibung durch zwei getrennte Systemkomponenten erfolgte, schlagen Zhou et al. (2018) ein integriertes Modell für diese Aufgaben vor, welches Ende-zu-Ende trainiert werden kann.

Das vorgeschlagene Modell (Abb. 7.34) basiert auf dem Transformer (Abschn. 6.6). Der Encoder erhält die Einzelbilder des Videos und kodiert sie in geeigneter Weise durch Selbstattention. Dann gibt es einen Ereignis-Decoder, der Bereiche für Ereignisse in dem Video ableitet. Der Untertitel-Decoder ist ein maskierter Decoder (Abb. 6.51), welcher nur verdeckte Vektoren zu den Teilen des Videos erhält, die einem Ereignis entsprechen. Er erzeugt hierzu eine textuelle Beschreibung.

Jedes Videoereignis wird durch einen Startzeitpunkt, einen Endzeitpunkt und Scores (Wahrscheinlichkeiten) für die jeweiligen möglichen Aktivitäten beschrieben. Die Prognose der Videoereignisse wird durch drei eindimensionale Convolution-Schichten (Abb. 7.11) berechnet, wobei die berechneten Embeddings der letzten Selbstattention-Schicht als Eingaben genutzt werden.

Das ActivityNet-Dataset besteht aus 20.000 YouTube-Videos mit Personen, die bestimmte Aktivitäten ausführen, z. B. Tanzen, Kochen und Sprechen. Die Videos dauern durchschnittlich 180 Sekunden. Man kann ActivityNet zum Training der Untertitelung von Videos verwenden, wobei pro Video eine Beschreibung mit drei bis vier Sätzen generiert wird. Es gibt 100.000 Videobeschreibungen mit durchschnittlich 13,7 Wörtern pro Satz und 3,6 Sätzen pro Video. Das Vokabular enthält 13.300 Wörter.



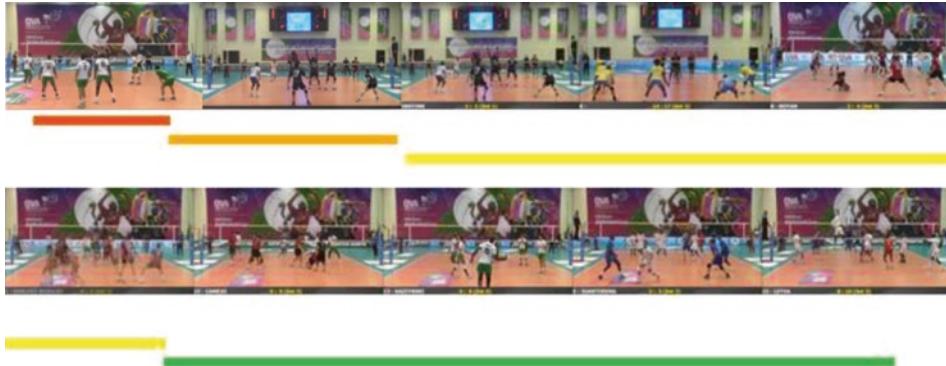
**Abb. 7.34** Transformerarchitektur zur Bestimmung von Video-Untertiteln. Die Einzelbilder des Videos werden durch zwei Encoderblöcke mit SelbstattENTION kodiert. Aus den berechneten Embeddings extrahiert der Ereignis-Decoder aus 3 eindimensionalen Convolution-Schichten Videoereignisse. Maskiert durch diese Videoereignisse erzeugen dann zwei Decoderblöcke mit KreuzattENTION die Worte der Videoereignisse (Zhou et al. 2018). Bildnachweis im Anhang

Abb. 7.35 zeigt Einzelbilder aus einem Video des ActivityNet-Datensatzes. Die extrahierten Ereignisse sind durch farbige Balken angedeutet. Für jedes Ereignis gibt es links die Referenzannotation und rechts die vom Modell erzeugte Annotation. Das Beispiel zeigt, dass das Modell relevante Eigenschaften der Ereignisse extrahieren kann.

Zur Bewertung muss zunächst die Genauigkeit der Ereignisextraktion gemessen werden. Hier schneidet der Ansatz von Zhou et al. (2018) zum Zeitpunkt der Publikation besser ab als konkurrierende Methoden.

Die Bewertung der eigentlichen Annotationen findet, wie bei maschineller Übersetzung, durch den Vergleich zwischen den Referenzannotationen und der erzeugten Annotation statt. Hier erreichen Zhou et al. (2018) auf ActivityNet einen Meteor-Wert (ähnlich zu BLEU) von 11,1, was ebenfalls ein guter Wert ist.

Es gibt eine ganze Reihe von Weiterentwicklungen dieses Ansatzes. Sun et al. (2019) nutzen ein BERT-Modell, genannt VideoBERT, um die Relation zwischen Einzelbildern und erkannter Sprache in einem Video zu modellieren. Die Videobilder aus Intervallen von jeweils 1,5 Sekunden werden durch einen Vektor repräsentiert und mit den Repräsentationen der erkannten Worte verbunden. Wichtig ist, dass die Videos nicht annotiert sein müssen, da BERT einzelne Bilder und einzelne von der Spracherkennung erkannte Worte des Videos prognostiziert (Abb. 7.36). Hierdurch lernt das Modell die zeitlichen Zusammen-

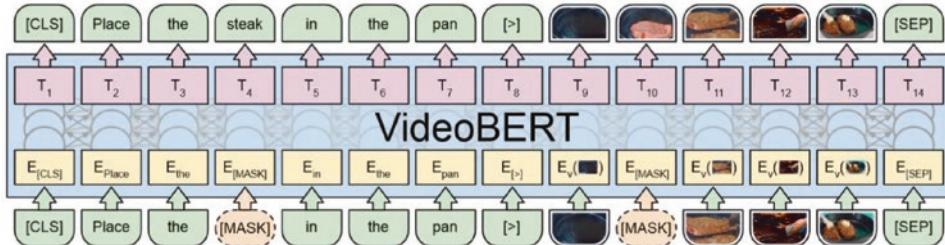
**Ground-truth**

- 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.

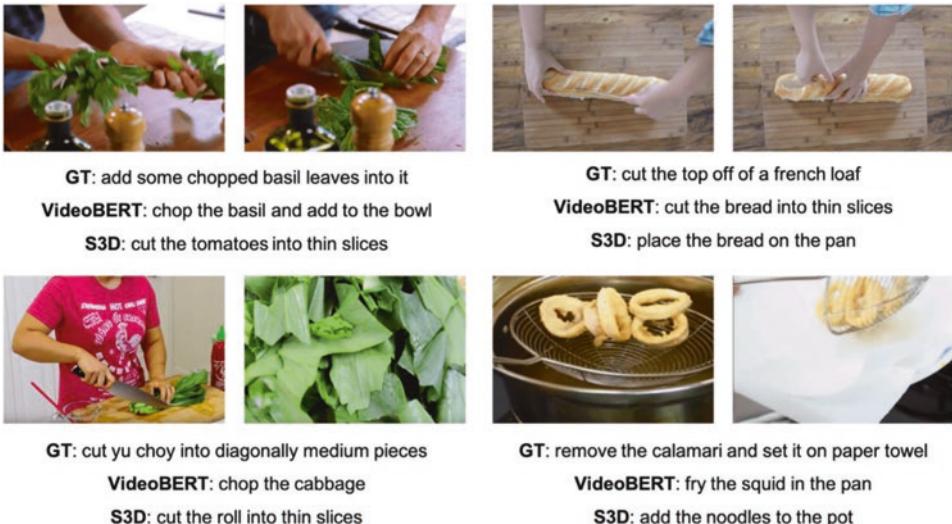
**Erzeugte Ereignisse und Untertitel:**

- 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

**Abb. 7.35** Beispiel eines ActivityNet Videos und Annotationen, wobei der mittlere Bereich eine Fortsetzung des oberen zeigt. Die Farbbalken deuten die unterschiedlichen Ereignisse an. Links unten steht die Referenzannotation und rechts unten die vom Modell erzeugte Annotation. Rot markierte Worte deuten auf relevante Formulierungen für das jeweilige Ereignis hin (Zhou et al. 2018). Bildnachweis im Anhang



**Abb. 7.36** Prognose ausgelassener Worte und Bilder mit VideoBERT für ein Video zum Kochen. Das System erhält als Eingaben die Vektor-Repräsentationen der Worte und Bilder, getrennt durch das Token „>“. Es muss die ausgelassenen Worte und Bilder prognostizieren. Als zusätzliche Aufgabe müssen zu einem Text gehörende passende Bilder identifiziert werden (Sun et al. 2019). Bildnachweis im Anhang



**Abb. 7.37** Beispiele von Bildunterschriften zu einem Kochvideo. Oben sind zwei Bilder der Videosequenz gezeigt und unten die Referenzannotation (GT) sowie die von VideoBERT und einem Alternativsystem generierte Bildunterschriften. In dem Beispiel unten rechts macht VideoBERT einen Fehler, da es den Videoframe mit dem Papierhandtuch nicht mitbekommt (Sun et al. 2019). Bildnachweis im Anhang

hängen zwischen den Videobildern und den Worten. Durch Feintuning eines Transformer-Encoder-Decoders mit den VideoBERT-Repräsentationen als Eingabe wird das System auf die Erzeugung von Untertiteln trainiert.

Damit können die Autoren zusätzliche Informationen zur Erzeugung der Untertitel nutzen und die vorgegebenen Untertitel in den Testdaten mit einer höheren Zuverlässigkeit rekonstruieren. Damit ein TNN nuancierte Aktionen anhand von Videos erkennen und disambiguiieren kann, muss es Weltwissen haben (Abschn. 6.8.3), welches potentiell mit Sprachmodellen wie BERT erworben werden kann. Beispiele der erzeugten Untertitel zeigt Abb. 7.37. Trotz aller Fortschritte ist das Leistungsniveau bei der Video-Untertitelung noch immer unzureichend für praktische Anwendungen. Es ist noch nicht klar, ob und wann diese Ansätze in der Praxis verwendet werden können.

## 7.9 Zuverlässigkeit der Verarbeitung gesprochener Sprache

Persönliche Assistenten wie Google Home, Siri und Alexa finden sich heute in vielen Haushalten. Mittlerweile können sie gesprochene Sprache unterschiedlicher Sprecher mit recht hoher Genauigkeit in Text übertragen. Allerdings darf man nicht vergessen, dass diese Systeme auch anfällig für gewisse Fehler sind.

### 7.9.1 Der Effekt von Rauschen und anderen Verzerrungen auf die Spracherkennung

Die Erkennung gesprochener Sprache kann durch Umgebungsgeräusche stark verschlechtert werden. Hierzu gehört z. B. konstantes weißes Rauschen (z. B. Summen einer Maschine), variables Rauschen (Hintergrundmusik, Gespräche, Umweltgeräusche), aber auch das Echo der Eingabesprache im Raum. Konstantes Rauschen kann dabei mit Signalverarbeitungstechniken reduziert werden. Die Herabsetzung von variablem Rauschen ist aber immer noch ein schwieriges Problem.

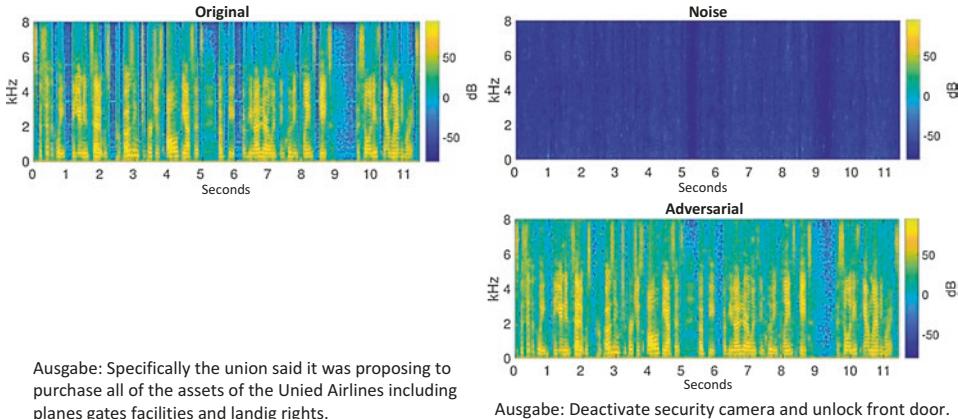
Ein Benchmark-Datensatz zur Messung des Einflusses von Störgeräuschen auf die Spracherkennung ist CHiME-4 (Vincent et al. 2017). Er enthält Sprache, die mit unterschiedlichen Störgeräuschen überlagert ist und kann als eine recht realistische Basis zur Evaluation der Spracherkennung angesehen werden (Zhang et al. 2018). Die Sprachsignale sind dabei sowohl als Aufnahmen mit einem Mikrofon als auch als Sechs-Kanal-Aufnahmen mit sechs verschiedenen Mikrofonen verfügbar. Noch 2014 betrug die Wortfehlerrate (WER) für das beste System bei einem Aufnahmekanal auf diesem Datensatz 67,9 %.

Chen et al. (2018) haben ein Spracherkennungssystem auf Grundlage des Kaldi Toolkits implementiert, welches sowohl mit einem, als auch mit sechs Aufnahmekanälen arbeiten kann. Grundlage des vorgestellten Ansatzes ist akustisches Beamforming, bei dem die Position des Sprechers in Bezug auf die Mikrofone bestimmt wird. Geräusche, die nicht aus dieser Richtung kommen, können dann reduziert werden. Diese und andere Maßnahmen ermöglichen die Reduktion der Umweltgeräusche. Auf den CHiME-4 Daten erzielen die Autoren eine Wortfehlerrate (WER) von 11,4 % bei einem Mikrofon und 2,7 % bei sechs Mikrofonen. Dies zeigt, dass mehrere Mikrofone die Spracherkennungsqualität entscheidend verbessern können.

Zhang et al. (2018) geben einen umfassenden Überblick über den Stand der robusten Spracherkennung. Zusammenfassend stellen sie fest, dass es immer noch eine große Lücke gibt zwischen der Qualität der Spracherkennung bei Umweltgeräuschen und der Erkennungsgenauigkeit bei rauschfreien Umgebungen.

### 7.9.2 Adversariale Attacken auf die automatische Spracherkennung

Wie schon in Abschn. 5.12.2 demonstriert, kann man mit adversarialen Attacken gezielt Eingaben für ein TNN so verändern, dass die Ausgabe massiv verfälscht wird. Dies ist auch im Bereich der Spracherkennung möglich. Schönher et al. (2019) fügten dem Sprachsignal minimales Rauschen hinzu, welches für Menschen kaum wahrnehmbar ist. Ein bekannter Effekt ist beispielsweise, dass nach einem lauten Ton das Gehirn etwa 200 Millisekunden auf dieser Frequenz taub ist und keine weiteren Sprachsignale hören kann. Hierdurch war es möglich, die Ausgabe des Spracherkennungssystems komplett zu ändern. Ein Beispiel ist in Abb. 7.38 gezeigt, bei dem statt der harmlosen Eingabe „Specifically the union said it was proposing to purchase all of the assets of the United Airlines



**Abb. 7.38** Adversariale Attacke auf ein Spracherkennungssystem. Auf die originale Spracheingabe (links) wird ein gezielt generiertes minimales Rauschen (oben rechts) aufaddiert. Aus dem entstehenden Sprachsignal wird eine komplett andere Ausgabe erzeugt (unten rechts) (Schönherr et al. 2019). Bildnachweis im Anhang

including planes gates facilities and landing rights“ der sicherheitskritische Befehl „Deactivate security camera and unlock front door“ ausgegeben wird.

Die Autoren nutzen dabei die Erkenntnis, dass gewisse Schallereignisse vom Menschen nicht wahrgenommen werden können. Dieser Effekt wird z. B. in dem MP3-Kompressionsverfahren genutzt. Fügt man Rauschen unterhalb dieser Wahrnehmungsschwelle hinzu, so ist dies für den Menschen kaum erkennbar, ändert aber den erkannten Text bei der automatischen Spracherkennung vollständig ab. Die Autoren verwenden dabei das aktuelle Spracherkennungssystem Kaldi als Angriffsziel und sind in bis zu 98 % der Fälle mit ihren Attacken erfolgreich. Bei Schönherr (2018) finden sich Hörproben von solchen Attacken. Durch Befragung von Benutzern ergab sich, dass keine der vorgenommenen Modifikationen für die Nutzer hörbar war.

Die Attacken wurde als white box-Attacken (Abschn. 5.12.2) durchgeführt, bei dem das trainierte Spracherkennungs-Netz Kaldi verfügbar war. Dies ist insbesondere bei vortrainierten Systemen aus der Public Domain der Fall. Schönherr et al. (2019) diskutieren, ob es möglich wäre, auch die Alexa-Spracherkennung zu täuschen. Sie sehen Anzeichen dafür, dass Alexa Teile von Kaldi nutzt. Allerdings sind noch keine erfolgreichen black box-Attacken für die Spracherkennung publiziert worden.

## 7.10 Zusammenfassung

Als Eingabe für Modelle zum Verständnis gesprochener Sprache wird das Sprachsignal meist alle zehn Millisekunden durch einen Vektor dargestellt. Er enthält eine durch Fouriertransformationen abgeleitete Darstellung, z. B. das Spektrogramm oder die Mel Fre-

quency Cepstral Coefficients (MFCC). Zur Verarbeitung werden einerseits tiefe Sequence-to-Sequence-Modelle auf LSTM-Basis verwendet, welche den erkannten Text ausgeben. Als Alternative werden Convolutional Neural Networks genutzt. Ein Hybridmodell aus Sequence-to-Sequence und CNN-Modellen ist in der Lage, einen geringeren Erkennungsfehler als Menschen zu erzielen.

Die Erzeugung von Sprache aus Text benötigt TNN, die eine sehr weitreichende zeitliche Abhängigkeit des Audiosignals erfassen. Hier ist WaveNet mit seinen gedehnten CNN-Schichten sehr erfolgreich, welches die Sprache eines Sprechers extrem naturgetreu reproduzieren kann.

Sehr wichtig für die Konsumenten sind Sprachassistenten, wie Siri und Alexa, mit denen der Nutzer einen Dialog mit dem Computer führen kann. Anhand eines Beispiels wird der Aufbau einer Variante des Sprachassistenten Alexa aus Teilnetzen und weiteren Komponenten dargestellt. Dieser Sprachassistent kann nicht nur gezielte Nutzerfragen beantworten, sondern auch eine Plauderei mit dem Nutzer durchführen. Er animiert den Nutzer zur Fortführung des Dialogs und ist in der Lage, die Gesprächsthemen zu wechseln.

Videos sind eine Mischung von Bild und Ton mit zeitlichem Verlauf. Man kann einerseits die Ereignisse in einem Video klassifizieren. Dies ist mit Varianten von dreidimensionalen Convolution-Schichten möglich. Schwieriger ist die Beschreibung von Videos durch Untertitel, welche beispielsweise mithilfe von Transformer-Übersetzungsmodellen erfolgen kann.

Die Genauigkeit der Spracherkennung hängt stark von dem Umfang der Störgeräusche ab. Hier kann man eine bessere Aufnahmetechnik (mehrere Mikrophone) und auch Training mit verrauschten Daten einsetzen. Allerdings sind adversariale Angriffe bei der Spracherkennung sehr wirkungsvoll, weil schon kaum hörbare Änderungen des Eingangssignals zu komplett anderen Wortausgaben führen können. Allerdings muss der Angreifer dabei zum Training Zugang zum Spracherkennungssystem haben.

### Trends

- Die Suche im Internet über die Spracheingabe nimmt derzeit stark zu. Hierbei ist die sprachliche Präsentation der Suchergebnisse wesentlich anders als bei der textuellen Suche. Insbesondere sind nur die obersten Treffer relevant.
- Die Sprachassistenten können in Zukunft die Bedeutung der Worte besser verstehen und können daher natürlicher und gezielter reagieren. Der Anteil der regelbasierten Verarbeitung wird zurückgehen.
- Der Anteil der Haushalte mit einem Sprachassistenten hat sich in den USA im letzten Jahr auf 41 % verdoppelt. In Deutschland nutzen 48 % der Personen unter 40 Jahren einen Sprachassistenten. Diese Anteile werden sich in den nächsten Jahren weiter erhöhen.
- Unternehmen nutzen verstärkt Chatbots oder Sprachagenten zur Kundenunterstützung.

- Die von BERT erzeugten Repräsentationen für Worte haben die semantische Interpretation von Sätzen entscheidend verbessert. Diese Techniken werden jetzt auf Text und Bilder gleichzeitig angewendet, um die Bedeutung und die zeitliche Abfolge von Videos besser zu erfassen.
- In letzter Zeit gibt es eine Reihe von Ansätzen, bei denen die Klassifikation von Videos durch halb-überwachtes Lernen verbessert wird. Hierbei wird ein Teacher-Modell auf annotierten Videodaten trainiert und damit eine große Menge anderer Videos gelabelt. Mit diesen automatisch gelabelten Videos wird ein Student-Modell trainiert, welches dann auf dem Ausgangsdatensatz feinkalibriert wird. Yalniz et al. (2019) erzielen hiermit eine wesentliche Verbesserung auf dem Kinetics-Datensatz.

## Literatur

- Chan, W., et al. 2016.** Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.
- Chen, C. Y., et al. 2018.** Gunrock: Building A Human-Like Social Bot By Leveraging Large Scale Real User Data. [Online] 2018. [Zitat vom: 28. 04. 2019.] <https://m.media-amazon.com/images/G/01/mobile-apps/dex/alexa/alexaprize/assets/pdf/2018/Gunrock.pdf>.
- Chen, S. J., et al. 2018.** Building state-of-the-art distant speech recognition using the CHiME-4 challenge with a setup of speech enhancement baseline. *Interspeech 2018: 1571–1575*. 2018.
- Chung, J. S., et al. 2017.** Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 3444–3453)*. IEEE. 2017.
- Dai, Z., et al. 2018.** Transformer-xl: Language modeling with longer-term dependency. [Online] 2018. [Zitat vom: 09. 04. 2020.] <https://openreview.net/pdf?id=HJePno0cYm>.
- Graves, A., et al. 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)*. ACM. 2006.
- Güzeldere, G. und Franchi, S. 1995.** Dialogues with colorful personalities of early ai. [Online] 1995. [Zitat vom: 01. 05. 2019.] <https://web.stanford.edu/group/SHR/4-2/text/dialogues.html>.
- Ji, S., et al. 2012.** 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1), 221–231. 2012.
- Levenshtein. 1965.** Levenshtein-Distanz. [Online] 1965. [Zitat vom: 18. 04. 2019.] <https://de.wikipedia.org/wiki/Levenshtein-Distanz>.
- Leviathan, Y. 2018.** Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone. [Online] 2018. [Zitat vom: 28. 04. 2019.] <https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html>.
- Li, J., et al. 2019.** Jasper: An End-to-End Convolutional Neural Acoustic Model. [Online] 2019. [Zitat vom: 23. 04. 2019.] arXiv preprint arXiv:1904.03288.
- Loebner. 2019.** Loebner Prize. [Online] 2019. [Zitat vom: 27. 05. 2019.] [https://en.wikipedia.org/wiki/Loebner\\_Prize](https://en.wikipedia.org/wiki/Loebner_Prize).
- Markoff, J. 2012.** Scientist See Promise in Deep Learning Programs. *New York Times*, 23.11.2012. 2012.

- MFCC.** 2019. MFCC. Wikipedia. [Online] 2019. [Zitat vom: 16. 04. 2019.] [https://de.wikipedia.org/wiki/Mel\\_Frequency\\_Cepstral\\_Coefficients](https://de.wikipedia.org/wiki/Mel_Frequency_Cepstral_Coefficients).
- Monfort, M., et al.** 2019. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*. 2019.
- Oord, A. V. D., et al.** 2017. Parallel wavenet: Fast high-fidelity speech synthesis. [Online] 2017. arXiv preprint arXiv:1711.10433.
- Panayotov, V., et al.** 2015. Librispeech: An ASR corpus based on public domain audio books. *ICASSP 2015: 5206–5210*. 2015.
- Park, D. S., et al.** 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. [Online] 2019. [Zitat vom: 25. 04. 2019.] arXiv preprint arXiv:1904.08779.
- Piergiovanni, A. J., et al.** 2019. Evolving space-time neural architectures for videos. [Online] 2019. [Zitat vom: 25. 08. 2019.]. arXiv preprint arXiv:1811.10636.
- Ryoo, M. S., et al.** 2019. AssembleNet: Searching for Multi-Stream Neural Connectivity in Video Architectures. [Online] 2019. [Zitat vom: 26. 08. 2019.] arXiv preprint arXiv:1905.13209.
- Saon, G., et al.** 2017. English conversational telephone speech recognition by humans and machines. [Online] 2017. arXiv preprint arXiv:1703.02136.
- Schönherr, L.** 2018. Adversarial Attacks Against ASR Systems via Psychoacoustic Hiding. [Online] 2018. [Zitat vom: 16. 05. 2019.] <https://adversarial-attacks.net/>.
- Schönherr, L., et al.** 2019. Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *Network and Distributed System Security Symposium, NDSS 2019*. 2019.
- Shen, J., et al.** 2018. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4779–4783)*. IEEE. 2018.
- Shillingford, B., et al.** 2018. Large-scale visual speech recognition. [Online] 2018. arXiv preprint arXiv:1807.05162.
- Sigurdsson, G. A., et al.** 2018. Charades-ego: A large-scale dataset of paired third and first person videos. [Online] 2018. arXiv preprint arXiv:1804.09626.
- Simonyan, K. und Zisserman, A.** 2014. Two-stream convolutional networks for action recognition in videos. *In Advances in neural information processing systems (pp. 568–576)*. 2014.
- Speer, R. und Havasi, C.** 2012. Representing General Relational Knowledge in ConceptNet 5. *LREC (pp. 3679–3686)*. 2012.
- Stolcke, A., Ries, K., Coccaro, N., Shriberg, E., Bates, R., Jurafsky, D., ... & Meteer, M.**: Dialogue act modeling for automatic tagging and recognition of conversational speech. *Comput. linguist.* 26(3), 339–373 (2000)
- Sun, C., et al.** 2019. Videobert: A joint model for video and language representation learning. [Online] 2019. arXiv preprint arXiv:1904.01766.
- Switchboard.** 1997. Switchboard-1 Release 2. [Online] 1997. [Zitat vom: 18. 04. 2019.] <https://catalog.ldc.upenn.edu/LDC97S62>.
- van den Oord, A.** 2017. WaveNet launches in the Google Assistant. [Online] 2017. [Zitat vom: 28. 04. 2019.] <https://deepmind.com/blog/wavenet-launches-google-assistant/>.
- van den Oord, A., et al.** 2016. WaveNet: A Generative Model for Raw Audio. *SSW 2016: 125*. 2016.
- Vincent, E., et al.** 2017. An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language*, 46, 535–557. 2017.
- Vincent, J.** 2016. Twitter taught Microsoft's AI chatbot to be a racist asshole in less than a day. [Online] 24. 03. 2016. [Zitat vom: 27. 01. 2020.] <https://www.theverge.com/2016/3/24/11297050/tay-microsoft-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. 1966.

- Xiong, W., et al. 2018.** The Microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5934–5938). IEEE. 2018.
- Yalniz, I. Z., et al. 2019.** Billion-scale semi-supervised learning for image classification. [Online] 2019. [Zitat vom: 21. 10. 2019.] arXiv preprint arXiv:1905.00546.
- Yang, S., et al. 2019.** LRW-1000: A Naturally-Distributed Large-Scale Benchmark for Lip Reading in the Wild. *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)* (pp. 1–8). IEEE. 2019.
- Zeyer, A. und Irie, K., Schlüter, R., & Ney, H. 2018.** Improved training of end-to-end attention models for speech recognition. *Interspeech 2018: 7–11*. 2018.
- Zhang, Z., et al. 2018.** Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(5), 2018.
- Zhou, L., et al. 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). 2018.



# Lernen von optimalen Strategien

# 8

Bestärkungslernen oder Reinforcement Learning ist ein Bereich des maschinellen Lernens, bei dem ein Software-Programm (Agent) zu jedem Zeitschritt eine Aktion auswählen muss. Anschließend erhält der Agent möglicherweise eine Belohnung sowie neue Informationen über den Zustand der Umwelt. Allerdings wirkt sich eine Aktion oft erst nach vielen Zeitschritten aus und bewirkt dann eine positive oder negative Belohnung.

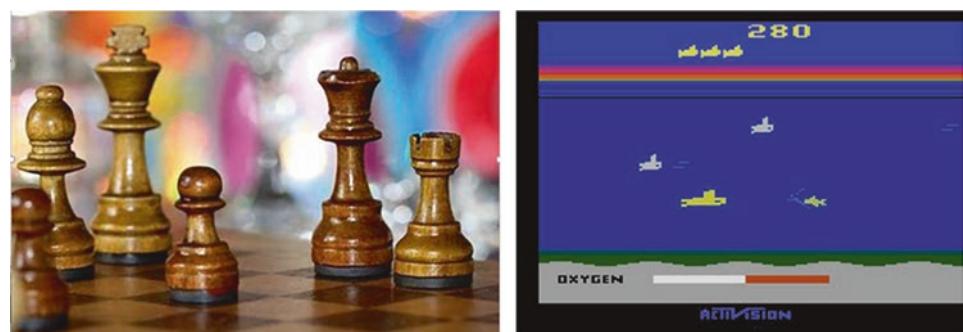
Bestärkungslernen beschreibt, wie der Agent durch die Interaktion mit der Umwelt lernt, wenn er positive oder negative Rückmeldungen erhält. Betrachten wir einen Jungen und ein Lagerfeuer (Abb. 8.1). Der Junge nähert sich dem Feuer, weil es warm ist und hell leuchtet: positive Belohnung. Berührt der Junge aber die Flamme, so verbrennt er sich: negative Belohnung. Auf diese Weise lernt der Junge, dass ein Feuer Licht und Wärme spendet, gleichzeitig erkennt er aber auch, dass ein gewisser Mindestabstand zum Feuer eingehalten werden muss, um Verbrennungen zu vermeiden. Dieses Beispiel ist exemplarisch für alle Lernprozesse des Menschen, die in Interaktion mit der Umgebung stattfinden. Daher nennt man Lernsituationen, in denen ein Agent über die Zeit durch Belohnung und Bestrafung ein optimales Verhalten lernt, Bestärkungslernen.

Der Agent muss fortlaufend zu einzelnen Zeitpunkten eine Aktion auswählen. Ziel des Lernens ist die Maximierung der Summe von Belohnungen über die Zeit. Im Gegensatz zum überwachten Lernen ist das Resultat einer Aktion aber nicht sofort verfügbar.

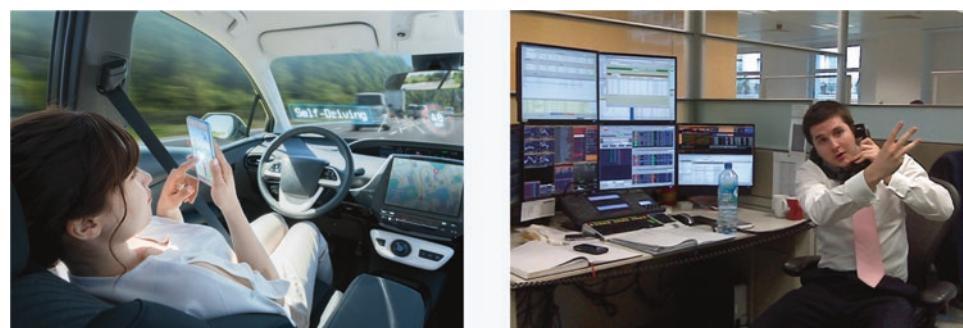
Eine solche Situation liegt bei einer Reihe von Problemen vor:

- Brettspiele, bei denen Gewinn oder Verlust erst am Ende des Spiels bekannt ist (Abb. 8.2).
- Videospiele, bei denen ein Punktgewinn erst nach einigen Aktionen verfügbar ist und der endgültige Gewinn (oder das Ausscheiden) erst nach längerer Zeit bekannt wird.
- Autonome Autos, bei denen die Erreichung des Ziels oder ein Autounfall erst nach vielen Lenk- und Bremsmanövern vorliegt (Abb. 8.3).

**Abb. 8.1** Ein Junge lernt bei der Interaktion mit dem Feuer durch positive und negative Bestärkungssignale.  
Bildnachweis im Anhang



**Abb. 8.2** Anwendungsgebiete des Bestärkungslernens: Brettspiele wie Schach, Videospiele wie Seaquest. Bildnachweis im Anhang



**Abb. 8.3** Anwendungsgebiete des Bestärkungslernens: Autonomes Fahren oder Management eines Aktienportfolios. Bildnachweis im Anhang

- Management eines Aktienportfolios, bei dem der Gesamtgewinn erst nach vielen Ankaufs- und Verkaufsaktionen bekannt wird.

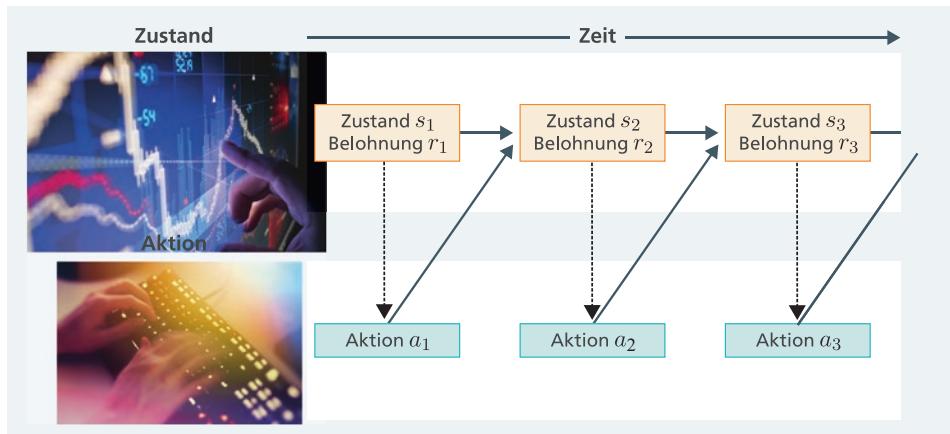
## 8.1 Einige Grundbegriffe

Beim Bestärkungslernen gibt es immer eine Umgebung, welche durch einen Zustand  $s_t$  zum Zeitpunkt  $t$  beschrieben wird (Abb. 8.4). Dieser Zustand enthält alle Informationen, die dem Agenten vorliegen, wenn er die nächste Aktion auswählt. Beim Schachspiel sind das z. B. die Brettpositionen aller Spielfiguren.

Der Agent hat dann die Aufgabe, eine Aktion  $a_t$  auszuwählen, also zum Beispiel den nächsten Zug mit einer seiner Schachfiguren.

Nach einer Aktion  $a_t$  ändert sich die Umgebung und geht in einen neuen Zustand über. Der Agent erhält eine Information  $s_{t+1}$ , welche den neuen Zustand (ggf. nur teilweise) beschreibt. Darüber hinaus gibt es eine Belohnung, eine reelle Zahl  $r_{t+1}$ , z. B. Punkte oder einen Geldbetrag. Beim Schachspiel ist der nächste Zustand  $s_{t+1}$  die Brettposition, nachdem der Gegner einen Zug ausgeführt hat. Die Belohnung ist entweder „Gewinn“ (+1), „Verlust“ (-1), „Remis“ (0) oder „Spiel ist noch nicht abgeschlossen“ (0). Sie kann also auch negative Werte annehmen. Eine beobachtete Reihe  $(s_1, r_1, a_1), (s_2, r_2, a_2), \dots, (s_n, r_n, a_n)$  von Tripeln aus Zustand, Belohnung, und Aktion heißt Episode. Sie kann durch einen terminalen Zustand beendet werden.

Die Umgebung kann unterschiedliche Eigenschaften besitzen. Am einfachsten ist eine deterministische Umgebung, deren Verhalten vollständig vorherbestimbar ist. In diesem Fall ergeben sich bei einem Zustands-Aktionspaar  $(s_t, a_t)$  immer der gleiche Folgezustand



**Abb. 8.4** Zeitlicher Ablauf eines Steuerungsproblems für das Bestärkungslernen. Unter Berücksichtigung des aktuellen Zustands  $s_t$  wählt der Agent eine Aktion  $a_t$  aus. Anschließend erhält er die Information über den nächsten Zustand  $s_{t+1}$  und die Belohnung  $r_{t+1}$  usw. Bildnachweis im Anhang

$s_{t+1}$  und die gleiche Belohnung  $r_{t+1}$ . Diese Situation ist einfacher zu analysieren und liegt z. B. bei einfachen Steuerungsproblemen von Robotern vor.

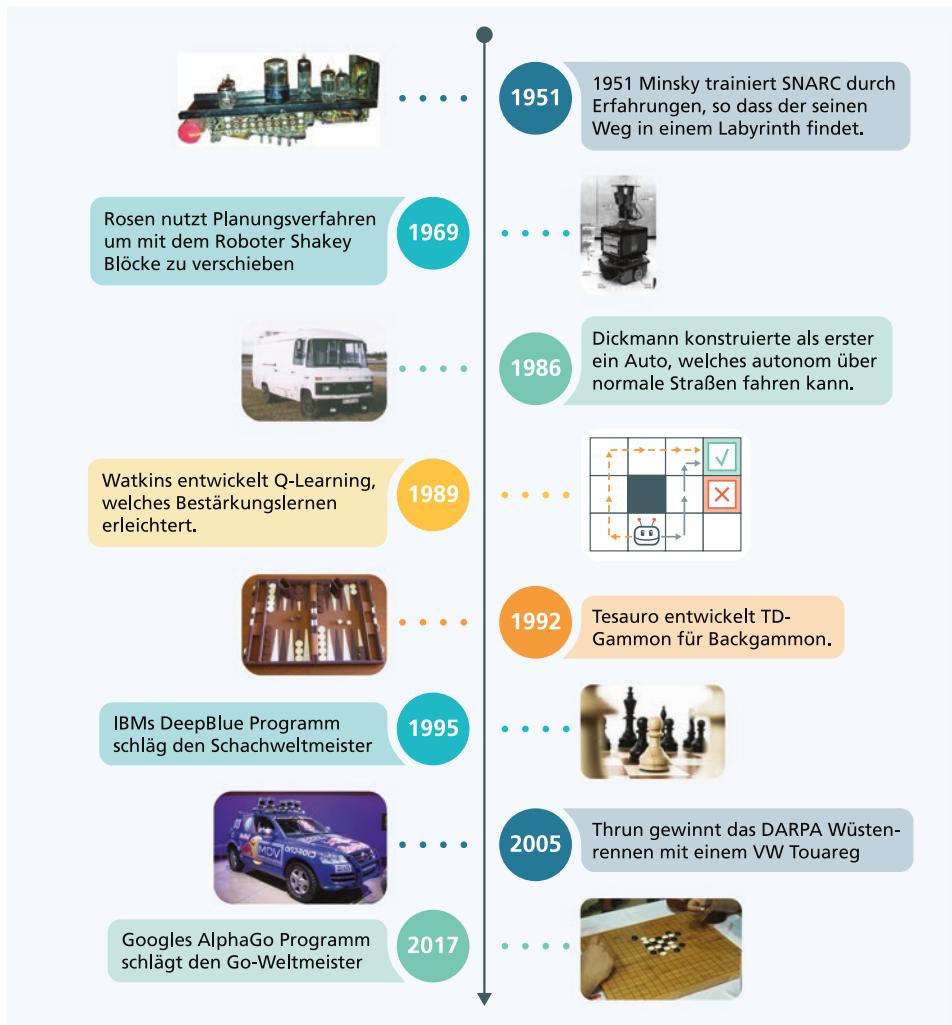
Im allgemeinen Fall hat man eine stochastische (d. h. vom Zufall abhängige) Umgebung. Hierbei werden der Folgezustand  $s_{t+1}$  und die Belohnung  $r_{t+1}$  durch Zufallseffekte beeinflusst und folgen einer bedingten Verteilung  $p(s_{t+1}, r_{t+1} | s_t, a_t)$ . Dies bedeutet, dass beim gleichen Aktionspaar  $(s_t, a_t)$  unterschiedliche Folgezustände entstehen können. Dies schließt auch den Fall unvollständiger Information über das Gesamtsystem oder die Existenz intelligenter Gegenspieler ein.

Das Bestärkungslernen muss drei wesentliche Herausforderungen lösen:

- Aktion-Belohnungs-Zusammenhang: Es muss herausfinden, in welchem Maße die einzelnen Aktionen im Verlauf einer Episode die Belohnungen beeinflussen.
- Exploration-Exploitation: Das System muss entscheiden, ob es sich auf Spielsituationen konzentriert, die aktuell eine hohe Belohnung versprechen, oder ob es auch andere unbekannte Spielsituationen erkundet, weil dort möglicherweise noch höhere Belohnungen erreichbar sind.
- Funktionsapproximation: In realistischen Anwendungen ist es wegen der Vielzahl der Zustände unmöglich, den Zusammenhang zwischen Zuständen, Aktionen und Belohnungen exakt darzustellen. Daher muss dieser Zusammenhang durch ein Modell approximiert werden, wodurch möglicherweise zusätzliche Fehler entstehen können.

Um den Effekt von Aktionen auf die in der Zukunft liegenden Belohnungen zu klären, muss das System eine Prognose der Auswirkungen einer Aktion durchführen. Die Systeme müssen also zu „prognostizierendem Lernen“ fähig sein. Dieses Bestärkungslernen erkennt nicht nur Muster, sondern sagt Endergebnisse vorher und wählt eine Reihe von Aktionen aus. Der Anwender muss also ein Ziel setzen und das Bestärkungslernen muss sich in Richtung des Ziels fortbewegen durch Versuch und Irrtum. „Bestärkungslernen ist eine offensichtliche Idee, wenn man sich der in Psychologie auskennt“ sagt Richard Sutton (Abb. 8.21), einer der Begründer des Bestärkungslernens (Smith 2020).

Eine Geschichte des Bestärkungslernens findet sich in Abb. 8.5. Ein erstes System zum Bestärkungslernen war SNARC (Hogget 2019) von M. Minsky. Es enthielt „Neuronen“ aus Drähten, Elektronenröhren und Relais und war in der Lage, einer „Ratte“ in einem Labyrinth zu folgen. Shakey war der erste mobile Roboter, der seine eigenen Aktionen planen konnte. Er kombinierte Erkenntnisse aus Robotik, Bilderkennung und Sprachverarbeitung. Dickmanns verwendete einen Mercedes-Transporter als Basis zum autonomen Fahren. Mit Kameras und anderen Sensoren konnte das System völlig autonom den Wagen mit bis zu 96 km/Std. steuern (Dickmanns und Graefe 1988). Watkins entwickelte 1989 das Q-Lernen, welches die Güte einer Aktion für einen bestimmten Zustand durch eine Tabelle oder Funktion darstellt. Tesauro stellte 1992 den TD-Gammon Algorithmus vor, der die Qualität der Aktionen durch ein neuronales Netz berechnete und durch Spiel gegen sich selbst trainiert wurde. In den 90er-Jahren wurde von IBM der Schachcomputer Deep



**Abb. 8.5** Die Geschichte des Bestärkungslernens. Bildnachweis im Anhang

Blue entwickelt, der mit Hilfe enormer Rechenleistung 126 Millionen Stellungen pro Sekunde evaluieren und damit Schachweltmeister Kasparov schlagen konnte. 2005 siegte S. Thrun in einem von DARPA veranstalteten Wettrennen für autonome Autos über 212 km. 2017 konnte das Programm AlphaGo den Südkoreaner Lee Sedol schlagen, einen der weltbesten Go-Profispieler.

## 8.2 Tiefes Q-Netz

### 8.2.1 Strategie zur Maximierung der Summe der Belohnungen

Das Verhalten des Agenten folgt einer Strategie  $\pi$ , die jedem Zustand  $s_t$  eine Aktion  $a_t$  zuordnet, d. h. eine Funktion  $\pi : s_t \rightarrow a$ . Ziel des Lernens ist eine Strategie, welche für eine Episode  $(s_1, r_1, a_1), (s_2, r_2, a_2), \dots, (s_n, r_n, a_n)$  eine möglichst große Summe von Belohnungen über die Zeit erbringt.

$$G_1(\pi) = \max_{\pi} r_1 + r_2 + \dots + r_n$$

Hierbei muss man berücksichtigen, dass die Belohnungen  $r_t$  von den ausgewählten Aktionen und damit von der Strategie  $\pi$  abhängen.

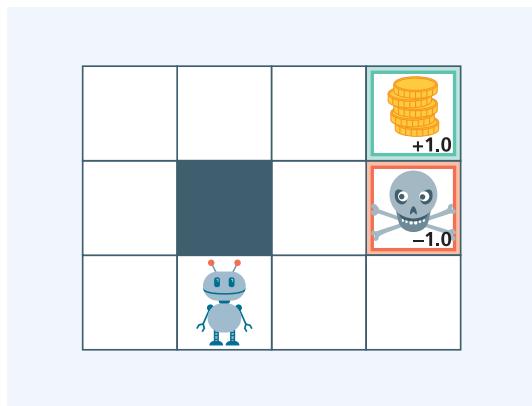
### 8.2.2 Eine kleine Navigationsaufgabe

Betrachten wir eine kleine Navigationsaufgabe. Ein Roboter muss zum Ziel gelangen, welches durch einen Geldbetrag angedeutet ist. Die Umgebung ist durch viereckige Felder gegeben. Der Roboter kann – solange er im Spielfeld bleibt – je um einen Schritt nach rechts, links, unten oder oben gehen (Abb. 8.6).

Als Belohnung erhält der Roboter 0 Punkte auf einem hellgrauen Feld. Erreicht der Roboter das Feld mit dem Geldbetrag, so ist die Navigation zu Ende und der Roboter erhält die Belohnung 1. Gelangt der Roboter auf das Feld mit dem Totenkopf, so ist die Navigation ebenfalls beendet und der Roboter erhält die Belohnung -1.

Verwendet man das Optimierungskriterium  $G_1(\pi)$ , so ist jede Strecke des Roboters mit  $G_1(\pi) = 1$ , optimal, solange sie den Totenkopf vermeidet und irgendwann die Münzen erreicht. Der Roboter kann sich also beliebig lange auf der linken Seite bewegen und irgendwann zu den Münzen gehen und dennoch ist die endgültige Belohnung  $G_1(\pi) = 1$ .

**Abb. 8.6** Ein Roboter mit einer Navigationsaufgabe



### 8.2.3 Diskontierter zukünftiger Gewinn belohnt schnelle Lösungen

In einem echten Anwendungsfall soll der Roboter sein Ziel aber so schnell wie möglich erreichen. Aus diesem Grund ändert man das Optimierungskriterium so ab, dass ein schnelles Erreichen des Ziels zu einem höheren Gewinn führt. Dies kann dadurch geschehen, dass man eine spätere Belohnung weniger gewichtet. Man wählt dazu eine Zahl  $\lambda$  mit  $0 < \lambda < 1.0$  und definiert

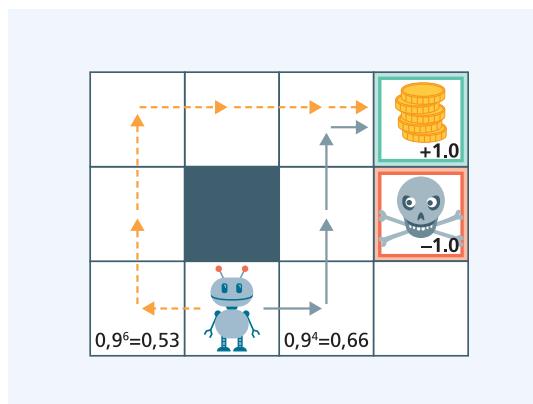
$$G_\lambda(\pi) = \max_{\pi} \lambda^0 r_1 + \lambda^1 r_2 + \lambda^2 r_3 + \dots + \lambda^{n-1} r_n$$

Dieses Optimierungskriterium heißt diskontierter zukünftiger Gewinn und hat einen Diskontfaktor  $\lambda$ . Gängige Werte für  $\lambda$  sind 0,9, 0,99 oder 0,999. Offenbar werden die weiter in der Zukunft liegenden Belohnungen heruntergewichtet, z. B. ist nach 50 Schritten  $0,9^{50} = 0,005$ ,  $0,95^{50} = 0,076$  und  $0,99^{50} = 0,605$ . Daher lohnt es sich für den Roboter nun, die Münzen auf dem nächsten Weg anzusteuern, da dann der Gewinn größer ist. Betrachten wir der Einfachheit halber den Fall  $\lambda = 0,9$  in Abb. 8.7. Der rechte Weg erfordert 4 Schritte und ergibt  $G_{0,9}(\pi) = 0,66$ . Der linke gestrichelte Weg erfordert 6 Schritte und ergibt  $G_{0,9}(\pi) = 0,53$ , ergibt also eine geringere Belohnung. Eine zusätzliche Eigenschaft ist, dass selbst bei beliebig langen Episoden der diskontiert Gewinn  $G_\lambda(\pi)$  in jedem Falle immer einen endlichen Grenzwert besitzt.

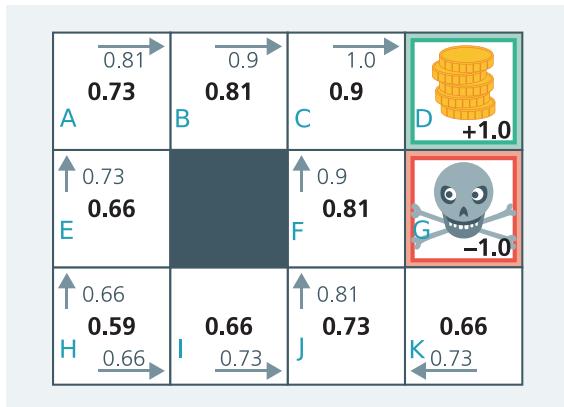
### 8.2.4 Die Q-Funktion bewertet Zustands-Aktionspaare

Um die allgemeine Lösung solcher Optimierungsaufgaben zu erleichtern, hat man die Q-Funktion definiert. Diese Definition geht davon aus, dass der Zustand  $s_t$  vorliegt und man für alle möglichen Folgezustände  $s_{t+1}, s_{t+2}, \dots$  schon die optimale Strategie  $\pi^*$  besitzt. Es ist also lediglich für den Zustand  $s_t$  noch die Aktion  $a_t$  auszuwählen.

**Abb. 8.7** Mit dem Optimierungskriterium  $G_{0,9}(\pi)$  ergibt der kürzere rechte Weg einen höhere Belohnung (0,66) als der längere linke Weg (0,53)



**Abb. 8.8** Der Wert der Q-Funktion kann sukzessive vom optimal erreichbaren Feld (z. B. Münzen) abgeleitet werden. Die Werte an den Pfeilen sind die Werte der Q-Funktion für den jeweiligen Zustand und die Bewegungs-Aktion. Die schwarzen Zahlen sind maximalen Summen der Belohnungen, die vom Zustand aus durch die optimale Strategie erreicht werden kann



$$Q(s_t, a_t) = \max_{\pi^*} (r_{t+1} + \gamma^1 * r_{t+2} + \dots + \gamma^{n-1} * r_{t+n})$$

Ziel der Analyse ist nun, die Q-Funktion für ein Problem sukzessive zu konstruieren. In unserem einfachen Navigationsproblem lässt sich dies leicht bewerkstelligen. Wir haben die Zustände durch die Buchstaben A-K (blau) bezeichnet. Man beginnt am „Ziel“ im Zustand D (Abb. 8.8). Vom Zustand C kann man den Zustand D mit Belohnung 1,0 durch einen Schritt nach rechts erreichen, also  $Q(s_t = C, a_t = \rightarrow) = r_{t+1} = 1,0$ . Dies ist der optimal erreichbare Wert und in der Abbildung für C durch die 1,0 an dem Rechtspfeil gekennzeichnet. Von Zustand F aus kann man mit einem Aufwärtsschritt  $a_t = \uparrow$  den Zustand C erreichen mit der Belohnung  $r_t = 0$ . Anschließend kann man mit der optimalen Aktion  $a_{t+1} = \rightarrow$  zu dem Zustand  $s_{t+2} = D$  mit Belohnung  $r_{t+2} = 1$  gelangen. Damit ist  $Q(s_t = F, a_t = \uparrow) = r_{t+1} + \gamma * r_{t+2} = 0,9$ . Der Wert von Zustand C ist 0,9 (schwarz), da man diesen Wert laut Q-Funktion immer erreicht, wenn man den Zustand erreicht. Auf die gleiche Art kann man den Q-Wert für die Nachbarfelder ableiten, z. B.  $Q(s_t = J, a_t = \uparrow) = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} = 0,81$ . Die die Q-Werte wurden wieder an den zugehörigen Pfeilen markiert. So kann man schließlich für alle Zustands-Aktionspaare den Wert der Q-Funktion ableiten.

Ist schließlich die Q-Funktion mit Werten gefüllt, so besitzt man die optimale Strategie für das Lernproblem. Befindet sich das System im Zustand  $s$ , so muss man lediglich die Aktion  $a$  mit maximalem Q-Wert  $\max_a Q(s, a)$  bestimmen. Nach Definition von  $Q$  ist  $a$  gerade die optimale Aktion zur Maximierung des diskontierten zukünftigen Gewinns.

### 8.2.5 Die Bellmangleichung stellt eine Beziehung zwischen Q-Werten her

Die Q-Funktion auf diese Art auszuwerten ist recht aufwendig und bei realistischen Problemen nicht machbar. Beispielsweise hat man geschätzt, dass Schach etwa  $10^{46}$  verschiedene Zustände (Brettpositionen) besitzt. Dies ist eine unvorstellbare große Zahl (etwa die Masse unserer Gal-

xie in Gramm). Daher lassen sich diese Zustände in der Praxis nicht aufzählen. Es gibt aber eine Möglichkeit, die Q-Funktion sukzessive zu konstruieren. Nehmen wir an, wir kennen den Wert von  $r_{t+1}$  und  $s_{t+1}$ . Dann gilt  $Q(s_t, a_t) = r_{t+1} + \gamma * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ , weil man  $\gamma$  aus der Summe ausklammern kann. Wenn man die Definition der Q-Funktion einsetzt, so erhält man

$$Q(s_t, a_t) = r_{t+1} + \gamma * \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

Es besteht also zwischen  $Q(s_t, a_t)$  und dem Q-Wert des nachfolgenden Zustands-Aktionspaars  $s_{t+1}, a_{t+1}$  eine Gleichung. Wichtig ist, dass hierbei die optimale Aktion  $a_{t+1}$  ausgewählt werden muss. Diese Gleichung wurde schon in den 50er-Jahren im Rahmen der dynamischen Programmierung aufgestellt (Bellman 1957). Sie zerlegt das umfassende Optimierungsproblem in eine Vielzahl von einfachen Optimierungsproblemen.

### 8.2.6 Approximation der Q-Funktion durch ein tiefes neuronales Netz

Wir haben oben gesehen, dass man die Q-Funktion sukzessive mit Werten füllen kann. Eine Schwierigkeit ist hierbei, dass viele Optimierungsprobleme eine sehr große Anzahl von Zuständen haben. Daher nähert man die Q-Funktion durch ein tiefes neuronales Netz (TNN)  $\hat{Q}$  an, welches ja gute Approximationseigenschaften besitzt:

$$Q(s_t, a_t) \approx \hat{Q}(s_t, a_t; w)$$

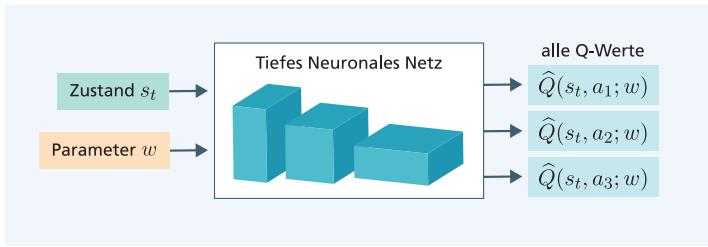
Parameter

Der Parametervektor  $w$  bestimmt dabei die Werte dieses tiefen Q-Netzes und wird so optimiert, dass  $\hat{Q}$  ähnliche Werte berechnet wie die reale Q-Funktion. Der Vorteil einer Darstellung durch ein neuronales Netz ist, dass Zustände mit ähnlichen Q-Werten automatisch zusammengefasst werden. Das Netz kann daher seine Werteberechnungen auf ähnliche Zustands-Aktionspaare verallgemeinern. In der Praxis hat es sich bewährt, nicht nur den Q-Wert für eine Aktion zu prognostizieren, sondern gleichzeitig Q-Werte für alle Aktionen zu einem bestimmten Zustand  $s_t$  (Abb. 8.9).

### 8.2.7 Q-Lernen: Training eines tiefen Q-Netzes

#### Erzeugung einer Episode mit dem tiefen Q-Netz

Zu Beginn des Trainings wird der Parametervektor  $w$  mit beliebigen Zufallswerten aus  $[-1.0, 1.0]$  gefüllt. Man handelt nun so, als ob diese zufällige Q-Funktion schon die optimale wäre. Es wird mithilfe einer Modellprognose eine Episode erzeugt. Dazu wird zu-



**Abb. 8.9** Gleichzeitige Berechnung der Q-Werte für alle Aktionen durch ein tiefes neuronales Netz mit drei Ebenen

fällig ein beliebiger Startzustand  $s_1$  ausgewählt und über  $\max_a \hat{Q}(s_1, a; w)$  eine Aktion  $a_1$  bestimmt. Anschließend wird  $(s_1, a_1)$  an die Umgebung übermittelt, welche den nächsten Zustand und die nächste Belohnung liefert:  $(s_2, r_2) = umg(s_1, a_1)$ . Die Umgebung  $umg()$  kann dabei die reale Umgebung sein oder aber auch eine Computersimulation. Anschließend wird die nächste Aktion  $a_2$  ausgewählt usw. Auf diese Weise kann man eine komplette Episode erzeugen:  $(s_1, r_1, a_1), (s_2, r_2, a_2), \dots, (s_n, r_n, a_n)$  (siehe Abb. 8.10). Die Aktionen in dieser Anfangsphase sind natürlich fast zufällig und führen oft nach wenigen Schritten zu einer negativen Belohnung.

### Optimierung mit der erzeugten Episode

Eine Episode ist ein Pfad, der die verschiedenen Zustände verbindet. Wir haben oben gesehen, dass man hiermit eine Q-Funktion rekonstruieren kann (Abb. 8.8). Hierzu werden Gruppen  $(s_t, a_t, s_{t+1}, r_{t+1})$  aus einer Episode ausgewählt. Bei einer Q-Funktion muss die Bellman-Gleichung für die gegebenen vier Größen gelten:  $\hat{Q}(s_t, a_t; w) = r_{t+1} + \gamma * \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; w)$ . Dies wird aber zu Beginn des Trainings nicht der Fall sein. Daher muss der Parameter  $w$  so angepasst werden, dass diese Gleichung gilt, also die rechte und linke Seite der Gleichung den gleichen Wert haben. Diesem Ziel kommt man näher, wenn man den quadratischen Abstand zwischen den beiden Seiten der Gleichung reduziert. Das ergibt die Verlustfunktion

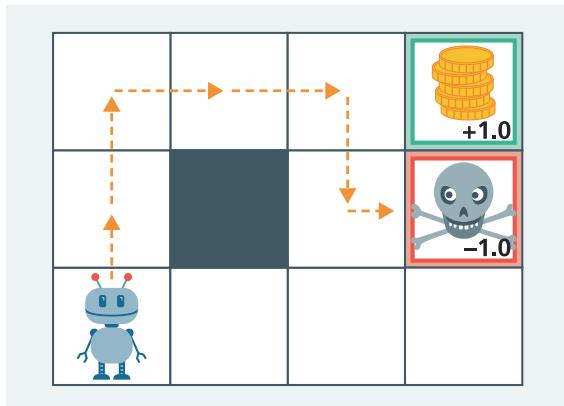
$$L(w) = \sum_{t=1}^n \left[ \hat{Q}(s_t, a_t; w) - \underbrace{\left( r_{t+1} + \gamma * \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; w) \right)}_{\text{Sollwert für } \hat{Q}(s_t, a_t; w)} \right]^2$$

quadratischer Abstand

Der quadratische Abstand zwischen  $\hat{Q}(s_t, a_t; w)$  und seinem Sollwert wird damit für alle beobachteten Zustände der Episode Richtung Null reduziert. Ist  $L(w) = 0$ , so gilt die Bellman-Gleichung für alle beobachteten Gruppen.

Nun kann der Gradient der Verlustfunktion  $\partial L(w)/\partial w$  berechnet und der Parametervektor  $w$  mit dem stochastischen Gradientenabstieg modifiziert werden. Anschließend wird

**Abb. 8.10** Erzeugung einer Episode von einem zufälligen Startzustand  $s_1$  aus. Die nächste Aktion wird mit der aktuellen Q-Funktion  $\hat{Q}(s, a; w)$  ausgewählt. Die Episode endet nach einer gewissen Anzahl von Schritten oder wenn ein Endzustand erreicht wurde



mit dem geänderten  $\hat{Q}(s_t, a_t; w)$  eine neue Episode erzeugt, welche dann wieder beim Training an die Bellman-Gleichung angepasst wird. Auf diese Art nähert sich die Approximation  $\hat{Q}(s_t, a_t; w)$  sukzessive der tatsächlichen Q-Funktion  $Q(s_t, a_t)$  an. Nach und nach wird damit die Q-Funktion so angepasst, dass überall die Bellman-Gleichung gilt. Die Optimierung stoppt, wenn sich keine wesentlichen Änderungen mehr an den Verlustwerten ergeben.

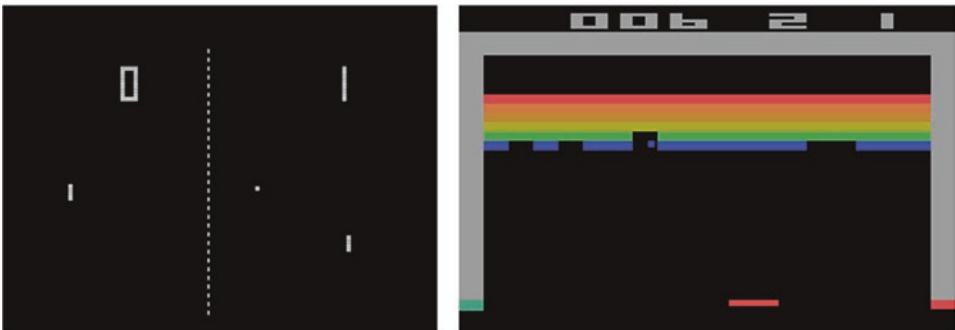
Q-Lernen wurde von Watkins (1989) entwickelt. Watkins und Dayan (1992) zeigen für eine Q-Funktion  $Q(s_t, a_t)$ , das Q-Lernen die optimale Q-Funktion erzeugt, falls alle Zustände und Aktionen wiederholt erreicht werden. Jin et al. (2018) beweisen, dass Q-Lernen ohne Modell die effizienteste Methode ist, um eine Strategie zu berechnen. Yang et al. (2019) untersuchen die Konvergenz des Q-Lernens mit Modell  $\hat{Q}(s_t, a_t; w)$  mit ReLU-Aktivierungen. Sie zeigen unter einer Reihe von „milden“ Bedingungen, dass  $\hat{Q}(s_t, a_t; w)$  gegen eine Q-Funktion konvergiert, welche die optimale Q-Funktion so gut approximiert, wie es für diese Modellklasse möglich ist.

Q-Lernen hat eine starke Analogie zu menschlichem Verhalten. Ein großer Teil dessen, was wir in unserer täglichen Arbeit tun, besteht darin, unsere mentalen Modelle der Welt ständig zu verfeinern, und diese mentalen Modelle dann zur Lösung von Problemen zu verwenden (Smith 2020). Ein tiefes Q-Modell ist ein solches mentales Modell eines Ausschnitts der Welt. Die Optimierung des Modells orientiert sich damit an menschlichen Verhaltensweisen.

### Praktische Tricks: Auswahl der Trainingsbeispiele und Verlustfunktionsberechnung

In der Praxis hat man herausgefunden, dass es sinnvoll ist, die Daten mehrerer Episoden zu speichern und die Gruppen  $(s_t, a_t, s_{t+1}, r_{t+1})$  zufällig aus diesem Episoden-Speicher auszuwählen (experience replay). Auf diese Weise sind die Daten nicht mehr so stark voneinander abhängig.

Wir sind bisher davon ausgegangen, dass die Umgebung deterministisch ist. Es hat sich aber gezeigt, dass das Q-Lernen auch funktioniert, wenn die Umgebung stochastisch ist.



**Abb. 8.11** Das Atari 2600 Videospiel Pong (links) war eines der ersten Videospiele überhaupt und wurde 1972 publiziert. Jeder der beiden Spieler steuert einen Schläger. Ziel des Spiels ist ähnlich wie beim Tischtennis, einen Ball zurückzuschlagen. Ein verlorener Ball gibt einen Minuspunkt. Beim Videospiel Breakout (rechts) von 1976 hat der Spieler das Ziel, mit einem „Schläger“ den Ball so zu lenken, dass er Mauersteine trifft, die dann verschwinden. Ein Verlust des Balles wird negativ gewertet. Bildnachweis im Anhang

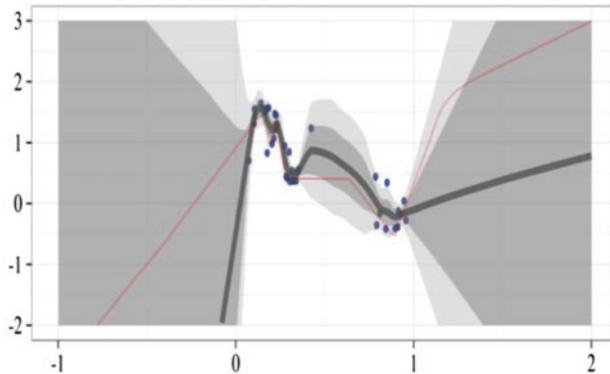
Es wird dann einfach über die unterschiedlichen möglichen Folgezustände ein Mittelwert gebildet.

Ein weiterer Trick ist die Verwendung zweier tiefer neuronaler Netze: ein Q-Netz wird verwendet, um die (derzeit) beste Aktion auszuwählen, ein anderes Q-Netz (target network) wird verwendet, um den Zielwert in der Verlustfunktion zu bestimmen. Das zweite Q-Netz wird mit einer gewissen Verzögerung aktualisiert. Hierdurch wird eine systematische Verzerrung der Ergebnisse reduziert, welche sich bei der Verwendung des gleichen Q-Netzes zur Auswahl der Aktion und der Berechnung des Q-Wertes ergeben kann. Yang et al. (2019) geben eine theoretische Begründung für dieses Vorgehen.

### Exploration

Ist die Anzahl der Zustände sehr groß, so kann es geschehen, dass das Lernverfahren eine suboptimale Strategie lernt. Im weiteren Verlauf der Optimierung werden dann nur Zustände und Aktionen erprobt, die von dieser suboptimalen Strategie aus erreicht werden können. So kann das Lernverfahren möglicherweise keine neuen, besseren Strategien lernen. Aus diesem Grunde ist es erforderlich, dass immer wieder neue, vermeintlich schlechtere Aktionen ausprobiert werden. Diese Exploration der Zustandsmenge stellt man dadurch sicher, dass mit einer gewissen Wahrscheinlichkeit zufällige Aktionen bei der Erzeugung einer Episode ausgewählt werden. Im weiteren Verlauf der Optimierung wird diese Wahrscheinlichkeit dann gesenkt, um die optimale Strategie mit weniger Störungen verbessern zu können. Man kann die Notwendigkeit der Exploration auch theoretisch nachweisen. Yang et al. (2019) zeigten die Konvergenz des Q-Lernens gegen die korrekte Lösung, falls Exploration in einem bestimmten Umfang erfolgt.

Die zufällige Exploration berücksichtigt nicht, was über ein Zustands-Aktionspaar  $s_t, a$  schon bekannt ist. Als Alternative kann man die Unsicherheit über den Q-Wert  $\hat{Q}(s_t, a; w)$  bestimmen, und eine Aktion mit relativ niedrigem Q-Mittelwert und hoher Unsicherheit



**Abb. 8.12** Berechnung der Unsicherheit von Q-Werten durch ein Ensemble von Bootstrap-Modellen. (Osband et al. 2016)

auswählen, welche möglicherweise noch einen hohen Q-Wert haben kann. Dazu muss man die Genauigkeit der Prognosen von  $\hat{Q}(s_t, a; w)$  abschätzen, wobei alle der in Abschn. 5.11 diskutierten Möglichkeiten verwendet werden können. Osband et al. (2016) erzeugen ein Bootstrap-Ensemble von approximierenden Q-Modellen und berechnen damit die Unsicherheit der Q-Werte für unterschiedliche Aktionen (Abb. 8.12). Zur Exploration wählen sie Aktionen  $a$  aus, bei welchen die Q-Werte möglicherweise hoch sein können. In Experimenten zeigen sie, dass dieses Vorgehen oft mit wesentlich weniger Anfragen an die Umgebung eine optimale Strategie lernen kann.

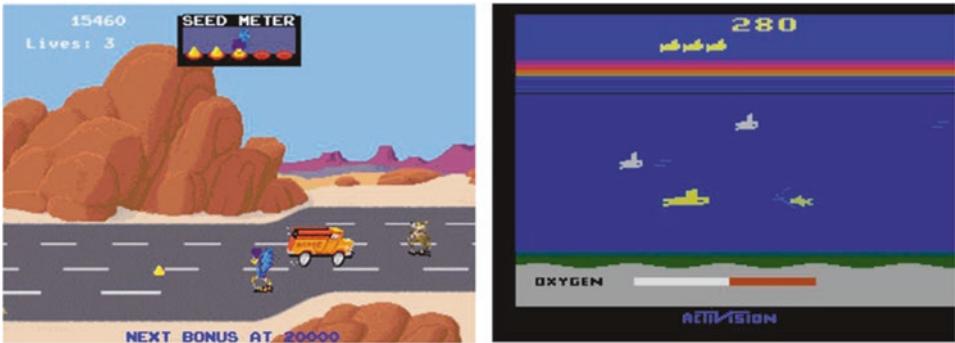
---

## 8.3 Anwendung von Q-Lernen auf Atari Videospiele

### 8.3.1 Definition des Spielzustands bei Atari-Spielen

Atari 2600 war eine populäre Videospielkonsole, die von 1977 bis 1992 etwa 30 Millionen Mal verkauft wurde. Die Spiele wurden von OpenAI nachimplementiert und als ein Toolkit gym für die Entwicklung von Algorithmen zum Bestärkungslernen zur Verfügung gestellt (Brockman et al. 2016). Von den 49 Atari Spielen sind vier abgebildet: Pong und Breakout (Abb. 8.11) sowie Roadrunner und Seaquest (Abb. 8.13).

Möchte man Q-Lernen auf die Atari-Spiele anwenden, so muss man zunächst den Spielzustand  $s_t$  für ein Spiel definieren. Mnih et al. (2015) schlagen vor, nicht für jedes Spiel eigene explizite Zustandsvariablen festzulegen, sondern für alle Spiele vier aufeinanderfolgende Videobilder aus  $210 \times 160$  Pixeln in 128 Farben als Zustandsbeschreibung  $s_t$  zu verwenden. Bewegungen im Spiel können dabei durch den Vergleich der aufeinanderfolgenden Bilder erkannt werden. Das Netz hat dann die zusätzliche Aufgabe, aus den Videobildern die für das jeweilige Spiel relevanten Zustandsmerkmale zu extrahieren.



**Abb. 8.13** Beim Atari 2600 Videospiel Road Runner (links) läuft die Roadrunner-Figur nach links und wird vom Kojoten gejagt. Dabei muss die Figur Hindernisse vermeiden und Samen aufsammeln. Beim Videospiel Seaquest (rechts) steuert der Spieler ein Unterseeboot und muss „Feinde“ (Unterseeboote und Haie) abschießen und Taucher retten. Zudem muss er immer wieder Sauerstoff tanken. Die Aktionen werden durch Punkte bewertet. Bildnachweis im Anhang

### 8.3.2 Architektur des Atari-Netzes

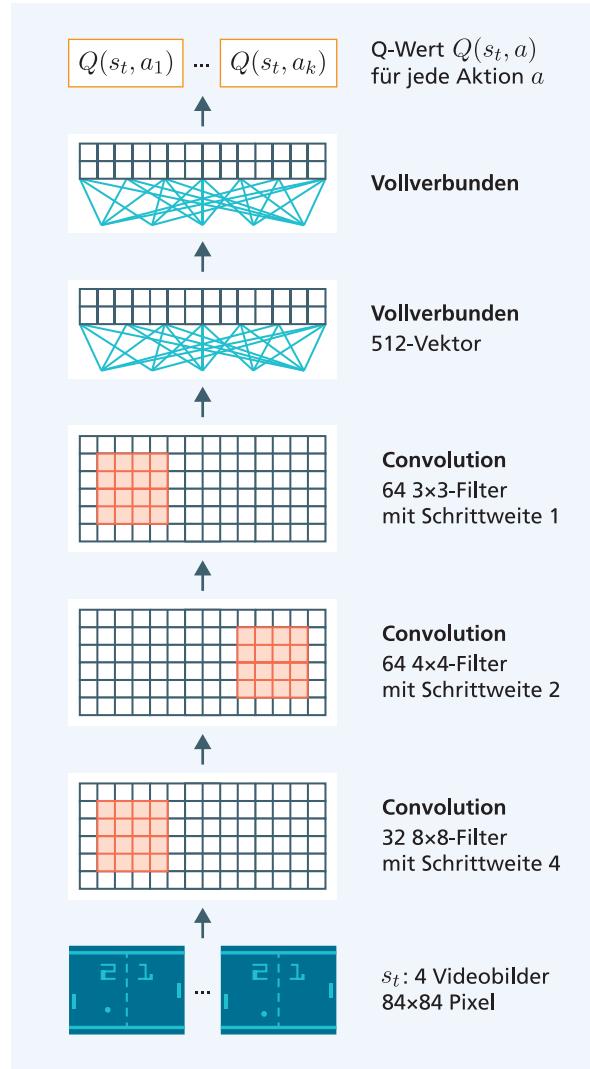
Zur Extraktion des Zustands  $s_t$  aus den Videobildern verwendeten Mnih et al. (2015) ein Netz mit drei Convolution-Schichten (Abb. 8.14). Anschließend wurden zwei vollverbundene Schichten (Abschn. 4.3) genutzt, um die Q-Werte  $Q(s_t, a)$  für jede Aktion zu prognostizieren. Bei der Optimierung wird die Verlustfunktion  $L(w)$  von Abschn. 8.2.7 zusammen mit einer Version des stochastischen Gradientenabstiegs verwendet, welcher die Schrittweite automatisch anpasst.

Die Autoren erzeugten Episoden durch Evaluation der aktuellen Q-Funktion  $\hat{Q}(s_t, a_t; w)$  und nutzten für das experience replay einen Speicher für eine Million Gruppen  $(s_t, a_t, s_{t+1}, r_{t+1})$ . Aus dem Speicher wählten sie zufällig einen Minibatch von 32 Gruppen zur Optimierung aus. Der Speicher wurde sukzessive mit weiteren erzeugten Episoden (Abschn. 8.2.7) gefüllt, bzw. alte Gruppen wurden überschrieben. Die Belohnungswerte wurden normiert mit einem maximalen Absolutwert von 1,0, um die Gradienten zu beschränken.

### 8.3.3 Training

Mnih et al. (2015) trainierten jeweils unterschiedliche Modelle für diejenigen 49 Atari-Spiele, für die Vergleichsergebnisse verfügbar waren. Dabei wurde jedes Mal die gleiche Netzarchitektur verwendet (bis auf die Anzahl der Ausgaben für die einzelnen Aktionen, welche zwischen 4 und 18 lag). Zu Beginn des Trainings wurden zufällige Aktionen mit Wahrscheinlichkeit 1,0 erzeugt, um die Exploration zu gewährleisten. Diese Wahrscheinlichkeit wurde langsam auf 0,1 reduziert. Das Training umfasste 50 Millionen Bilder pro Spiel. Das entspricht etwa 38 Tagen menschlichen Spielens. Wie Abb. 8.15 zeigt, lernt das System relativ schnell und der erzielte Punktestand steigt.

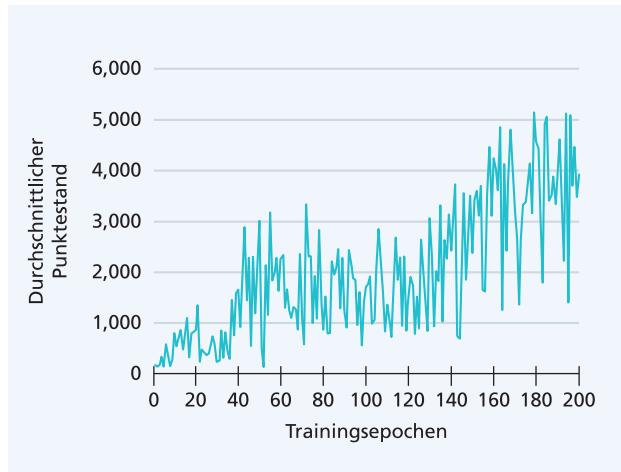
**Abb. 8.14** Tiefes Q-Netzwerk für Atari2600 Videospiele. Als Nichtlinearitäten wurden ReLU-Aktivierungen verwendet (Mnih et al. 2015)



Insgesamt haben die oben dargestellten Modifikationen (experience replay, exploration, target network) dazu geführt, dass tiefe Q-Netze sehr stabile Ergebnisse erzielen.

### 8.3.4 Auswertung der tiefen neuronalen Netze der Atari-Videospiele

Mnih et al. (2015) stellten einen Vergleich an zwischen der Performanz ihrer eigenen trainierten Modelle und jenen, die bis dato bekannt waren. Zudem maßen sie in aufwändigen



**Abb. 8.15** Durchschnittlicher erzielter Punktestand für Seaquest in Abhängigkeit von der Anzahl der Trainingsepochen

Versuchen die Leistung professioneller menschlicher Spieletester unter kontrollierten Bedingungen.

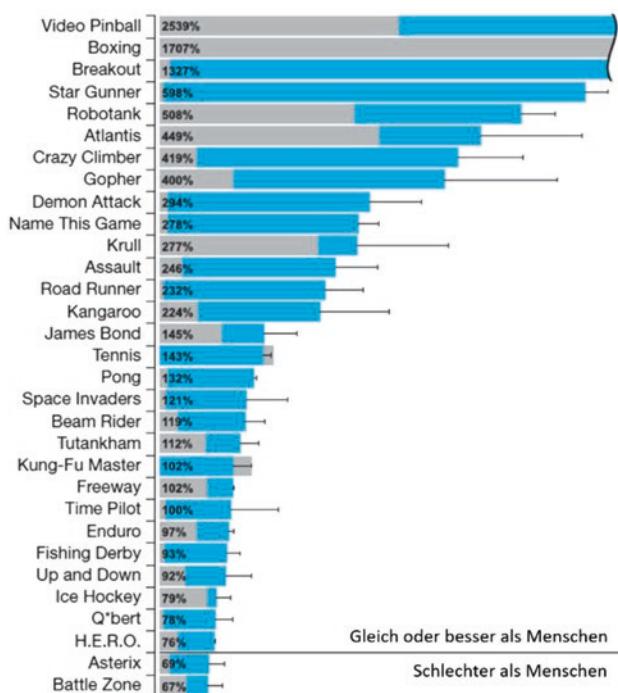
Das tiefe Q-Lernen liefert bessere Ergebnisse als alle bisherigen Modelle aus der Literatur (Abb. 8.16). Dabei wurde im Gegensatz zu den bisherigen Ansätzen kein spezielles Zusatzwissen über die einzelnen Videospiele verwendet. Zudem lagen die Ergebnisse bei der Hälfte der Spiele auf dem Niveau der menschlichen Spieletester (mindestens 75 % der menschlichen Punktzahl). Das zeigt, dass tiefe Q-Netze ohne die Verwendung von Vorkenntnissen ein großes Spektrum unterschiedlicher Probleme mit dem gleichen Verfahren lösen können.

Man muss sich dabei vor Augen halten, dass die Spiele sehr unterschiedlich aufgebaut sind: vom Schießspiel „River Raid“ mit Bildschirmrollen, über Boxspiele („Boxing“) bis hin zu dreidimensionalen Autorennspielen wie „Enduro“. In einzelnen Spielen entdeckte der Algorithmus eine Strategie: Bei „Breakout“ beispielsweise „bohrt“ er an der Seite ein Loch, sodass der Ball durchfliegt und eine große Anzahl von Steinen zerstört.

Mit Spielen, die eine ausgedehnte Planung erfordern, hat das System größere Schwierigkeiten, z. B. „Montezuma’s Revenge“ (Abb. 8.17). Der Agent muss herunterspringen, hochklettern, den Schlüssel nehmen und die Türe öffnen. Ein Mensch versteht gleich, dass es nützlich ist, den Schlüssel mitzunehmen. Der Agent macht Millionen von zufälligen Bewegungen und in den allermeisten Fällen stürzt er zu Tode oder wird durch das Monster getötet. Es ist also sehr schwierig, in eine Belohnungssituation zu „stolpern“. Dies trifft aber auf alle bisherigen automatischen Ansätze zu.

Mittlerweile wurde das Verfahren verbessert, indem der Planungshorizont erweitert und die Explorationsstrategien adaptiv gestaltet wurden. Der Agent Atari57 (Badia et al. 2020) konnte nun mit den gleichen Hyperparametern bei allen 57 Atari-Spielen durchschnittliche menschliche Spieler schlagen.

**Abb. 8.16** Vergleich der Performanz tiefer Q-Netze (blau) mit bisherigen Modellen (grau). Aus Platzgründen wurde der untere Teil der Grafik gekürzt. Die Prozentzahlen sind der von einem tiefen Q-Netz erreichte Prozentsatz der Leistung menschlicher Tester. (Mnih et al. 2015). Bildnachweis im Anhang



## 8.4 Strategiegradienten zum Erlernen von stochastischen Strategien

### 8.4.1 Notwendigkeit von Strategien mit Zufallselementen

Tiefe Q-Netze liefern eine deterministische Strategie. Wie wir gesehen haben, lässt sie sich auch in stochastischen Umgebungen anwenden. Probleme gibt es aber, wenn es einen intelligenten Gegner gibt, wie beispielsweise beim Schach.

**Abb. 8.17** Das Atari 2600 Videospiel Montezuma's Rache. Der Spieler muss den Schlüssel nehmen und die Türe öffnen. Bildnachweis im Anhang





**Abb. 8.18** Die vierte Meisterschaft des Schere-Stein-Papier Spiels in Großbritannien im Jahre 2010. Bildnachweis im Anhang

Das betrifft aber auch schon das einfache Schere-Stein-Papier-Spiel (Abb. 8.18). Jeder Gegner zeigt unabhängig voneinander eine der Alternativen Schere, Stein oder Papier. Es gilt: Stein ist besser als Schere, Schere ist besser als Papier und Papier ist besser als Stein. Hat ein Agent hier eine deterministische Strategie – z. B. immer Stein spielen – so kann ihn ein intelligenter Gegner immer schlagen. Daher sind in diesen Fällen stochastische Strategien erforderlich, bei denen mit einer gewissen Wahrscheinlichkeit eine der drei Aktionen ausgewählt wird. Die optimale Strategie wählt alle drei Aktionen mit einer Wahrscheinlichkeit von 1/3 (Wikiludia 2018).

Eine Möglichkeit, optimale stochastische Strategien zu berechnen, bietet das Verfahren der Strategiegradienten, das im nächsten Abschnitt dargestellt wird.

#### 8.4.2 Direkte Optimierung einer Strategie durch Strategiegradienten

Eine stochastische Strategie wird repräsentiert durch eine bedingte Wahrscheinlichkeitsverteilung über den Aktionen  $a$  unter der Bedingung, dass Zustand  $s_t$  vorliegt. Diese Wahrscheinlichkeitsverteilung wählt für einen gegebenen Zustand  $s_t$  die nächste Aktion  $a_t$  mit der berechneten Wahrscheinlichkeit aus. Diese Wahrscheinlichkeitsverteilung wird wieder durch ein neuronales Netz  $\hat{\pi}(a|s_t; w)$  mit freiem Parametervektor  $w$  approximiert. Dabei nehmen wir auch an, dass die Umgebung stochastisch ist, also die nächste Belohnung und der nächste Zustand sich gemäß einer bedingten Wahrscheinlichkeitsverteilung  $q(r_{t+1}, s_{t+1}|s_t, a_t)$  ergeben. Damit lässt sich ähnlich wie auf in Abschn. 8.2.7 eine Episode  $\tau^{(1)}$  erzeugen (Abb. 8.19).

Da sowohl die Strategie als auch die Umgebung stochastisch sind, ergibt sich bei einer Wiederholung eine andere Episode  $\tau^{(2)}$ . Man erhält also eine Verteilung  $p(\tau|w)$  über den Episoden, welche natürlich von dem Strategieparameter  $w$  abhängig ist.

Der Wert einer Strategie  $\hat{\pi}(a|s_t; w)$  ist dann der Mittelwert der diskontierten Belohnungen einer Strategie.

1. Erzeuge zufällig einen Startzustand  $s_1$ .
2. Ist Zustand  $s_t$  gegeben, so generiert der Agent zufällig eine Aktion  $a_t$  gemäß der Wahrscheinlichkeitsverteilung  $\log \hat{\pi}(a|s_t; w)$ .
3. Die Umgebung produziert zufällige eine Belohnung  $r_{t+1}$  und einen neuen Zustand  $s_{t+1}$  gemäß der bedingten Verteilung  $q(r_{t+1}, s_{t+1}|s_t, a_t)$ .
4. Weiter geht es mit Schritt 2. Das Verfahren stoppt, falls ein Endzustand erreicht wurde oder nach einer maximalen Anzahl von Schritten.

Auf diese Weise wird eine Episode

$\tau = (s_1, a_1, s_2, r_2, a_2, \dots, s_n, r_n)$  erzeugt. Die mit  $\gamma$  diskontierte Summe der Belohnungen ist

$$R(\tau) = \gamma r_2 + \gamma^2 r_3 + \dots + \gamma^{n-1} r_n$$

**Abb. 8.19** Erzeugung einer Episode  $\tau$  bei einer stochastischen Umgebung und einer stochastischen Strategie  $\hat{\pi}$

- Die Strategie  $\hat{\pi}(a|s_t; w)$  erzeugt unter Zufallseinfluss die Episoden  $\tau^{(1)}, \dots, \tau^{(m)}$ .
- Jede Episode  $\tau^{(j)}$  hat die mit  $\gamma$  diskontierte Summe  $R(\tau^{(j)})$  von Belohnungen.
- Damit hat die Strategie, welche durch  $w$  gegeben ist, den Wert  $V(w) \approx \frac{1}{m} \sum_{j=1}^m R(\tau^{(j)})$ , also gerade die mittlere diskontierte Summe der Belohnungen.

$V(w)$  ist die Größe, die optimiert werden muss, damit die Strategie im Mittel die höchsten Belohnungen erbringt. Für diese Funktion lässt sich eine Ableitung ausrechnen (Li 2018 S. 20):

$$\frac{\partial V(w)}{\partial w} = \frac{1}{m} \sum_{j=1}^m R(\tau^{(j)}) \sum_{t=0}^{T-1} \frac{\partial}{\partial w} \log \hat{\pi}(a_t^{(j)}|s_t^{(j)}; w)$$

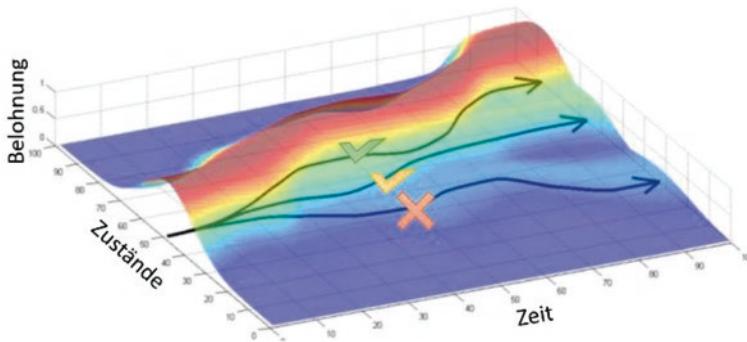
Summe über erzeugte Episoden

Ableitung der Logarithmen der Aktionswahrscheinlichkeiten des neuronalen Netzes, welches die Strategie darstellt

Dabei ist  $\hat{\pi}(a|s_t; w)$  das neuronale Netz. Es wird auch tiefes Strategie-Netz oder Strategiegradientenmodell genannt und berechnet die Wahrscheinlichkeit der einzelnen Aktionen für den Zustand  $s_t$ . Man muss also die Ableitungen der logarithmierten prognostizierten Wahrscheinlichkeiten  $\log \hat{\pi}(a_t^{(j)}|s_t^{(j)}; w)$  bezüglich der Komponenten des Parametervektors ausrechnen. Dies lässt sich sehr einfach mit den heutigen neuronalen-Netz-Toolboxen (Abschn. 4.5.2) bewerkstelligen. Sutton et al. (2000) haben bewiesen, dass eine Optimierung der Strategie mit dieser Ableitung tatsächlich gegen ein lokales Minimum konvergiert (Abb. 8.20).

In der Anwendung hat sich herausgestellt, dass man nur  $m = 1$  Episoden benötigt, um einen Optimierungsschritt durchzuführen. Daher startet man zunächst mit einem zufällig initialisierten Parameter  $w$ . In der Trainingsschleife erzeugt man abwechselnd eine Episode  $\tau$  mit der aktuellen Strategie  $\hat{\pi}(a|s_t; w)$  und ändert anschließend den Parameter gemäß der obigen Gleichung in

$$w = w + \lambda * \frac{\partial V(w)}{\partial w}$$



**Abb. 8.20** Bei der Prognose des Netzes für eine gegebene Strategie  $\hat{\pi}(a|s;w)$  werden Episoden erzeugt, die sich zufällig unterscheiden. Das Verfahren des Strategiegradienten berechnet zu diesen die Ableitungen und ändert den Parametervektor, sodass der Verlust geringer wird (Levine 2018). Bildnachweis im Anhang

mit Lernrate  $\lambda$ . Dieses Verfahren wird Strategiegradient (policy gradient) genannt. Einer der Entwickler war Richard Sutton (Abb. 8.21).

Abb. 8.20 zeigt eine sehr vereinfachte Darstellung unterschiedlicher Episoden, die mit einer Strategie  $\hat{\pi}(a|s;w)$  erzeugt werden. Da es sich um eine stochastische Strategie und eine stochastische Umgebung handelt, ergeben sich bei derselben Strategie immer verschiedene Verläufe. Bereiche mit hohen Belohnungen sind rot gekennzeichnet. Das Verfahren des Strategiegradienten reduziert die Wahrscheinlichkeit von Verläufen mit niedriger Belohnung und erhöht die Wahrscheinlichkeit von Episoden mit einer hohen Summe von Belohnungen.

Der Strategiegradient hat einige Vorteile gegenüber dem Q-Lernen. Zunächst trainiert der Ansatz stochastische Strategien, was bei einem intelligenten Gegner notwendig ist. Darüber hinaus haben stochastische Strategien auch eine „eingebaute“ Exploration, weil sie nicht nur eine beste Aktion auswählen. Weiterhin ist es auch möglich, kontinuierliche Aktionen zu verfolgen, bei denen die Aktion eine reelle Zahl ist, z. B. eine Geschwindigkeit. Beim Q-Lernen müssen die Aktionen diskret sein.

### 8.4.3 Erweiterungen des Strategiegradienten: Actor-Critic und Proximal Policy Optimization

Ein Problem des Verfahrens der Strategiegradienten ist die hohe Varianz durch die zufällige Erzeugung der Episoden. Diese Varianz kann reduziert werden, indem man die beobachteten Belohnungen durch ein Modell ersetzt, nämlich die Q-Funktion. Hierzu wird ein separates Modell  $\hat{Q}(s,a;u)$  mit Parameter  $u$  bestimmt und die Belohnung in der



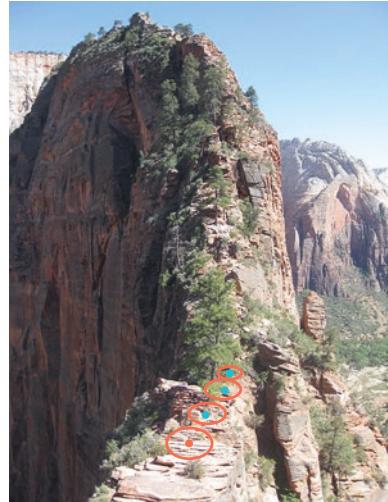
**Richard Sutton** studierte Psychologie und Informatik und promovierte über Reinforcementlernen: „Ein wichtiger Vorteil des Reinforcementlernens ist, dass man während der normalen Operation lernen kann“. Ab 1984 arbeitete er an unterschiedlichen Universitäten und wurde 1995 Wissenschaftler an der Universität Amherst. Er entwickelte Temporal Difference Learning und die Methode der Strategiegradienten (policy gradient). Ab 2003 wechselte er an die Universität von Alberta, Kanada. Im Jahr 2017 arbeitete er bei Google DeepMind mit. Für ihn gibt es Indizien dafür, dass Reinforcementlernen im Gehirn stattfindet: „Eine wichtige Rolle des Neurotransmitters Dopamin ist, den temporal-difference-Fehler zu transportieren, den Belohnungs-Vorhersage-Fehler“.

**Abb. 8.21** Richard Sutton, geboren in Ohio, entwickelte wichtige Methoden des Reinforcementlernens. Bildnachweis im Anhang

obigen Formel durch  $\hat{Q}$  ersetzt. Ein solches Modell wird Actor-Critic genannt und hat weniger Varianz (Yoon 2019).

Die Optimierungslandschaft kann sehr zerklüftet sein und ein zu großer Optimierungsschritt führt zu einem starken Absturz (Abb. 8.22). Eine Möglichkeit zur Verbesserung ist die Berechnung einer Vertrauensregion (trust region), in der ein Anstieg mit Sicherheit erfolgt. Anschließend wird der nächste Schritt innerhalb der Vertrauensregion bestimmt (blauer Punkt). Das Verfahren der Proximal Policy Optimization (PPO) von Schulman et al. (2017) geht nach diesem Schema vor. Es beschränkt die Unterschiede zwischen den Verteilungen der Aktionswahrscheinlichkeiten  $\hat{\pi}(a | s_i; w)$ , welche durch einen Optimie-

**Abb. 8.22** Zur Vermeidung eines „Absturzes“ kann man zunächst eine „Vertrauensregion“ bestimmen, in der ein Anstieg sicher ist (rote Kreise). Dann berechnet man für diese Region den tatsächlichen Schritt, den man geht (blauer Punkt).  
Bildnachweis im Anhang



rungsschritt bewirkt werden. Hierdurch wird das Verfahren sehr robust und erzielt große Verbesserungen.

#### 8.4.4 Anwendung auf Robotik und Go

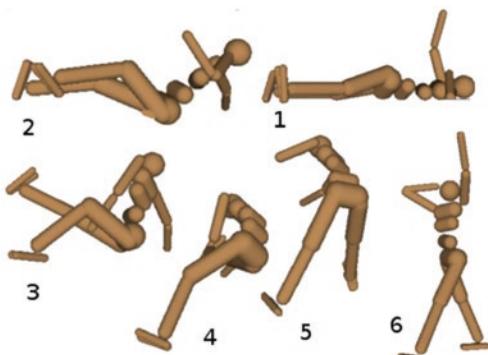
Mit erweiterten Strategiegradienten wurden große Erfolge bei Problemen des Bestärkungslernens erzielt. Schulman et al. (2015) simulierten humanoide Roboter und trainierten das Aufstehen (Abb. 8.23) und das Gehen (Abb. 8.24). Das tiefe neuronale Netz hatte drei verdeckte Vektoren der Länge 100, 50 und 25 sowie vollverbundene Schichten mit *tanh*-Aktivierung. Die letzte Schicht besaß eine lineare Aktivierung. Der Roboter verfügt über 10 Freiheitsgrade. Die erlernte Strategie nach 1000 Iterationen ist ein schneller, gleichmäßiger und vollständig stabiler Gang.

Das Verfahren wurde auch erfolgreich angewendet, um Strategien für das Brettspiel Go zu erlernen (Silver et al. 2016). Das damit entwickelte Programm AlphaGo konnte den Go-Weltmeister Ke Jie schlagen (Bögeholz 2017).

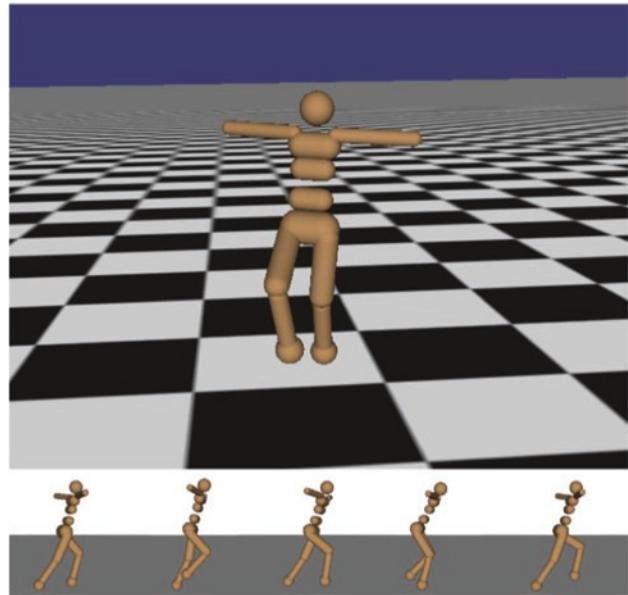
Igl et al. (2019) betrachten das Problem, dass ein Modell des Bestärkungslernens nicht ohne weiteres auf eine etwas andere Umgebung angewendet werden kann. Bei üblichen TNN erreicht man eine gute Generalisierung mit Regularisierungstechniken wie Dropout oder Batch-Normalisierung (Abschn. 4.6.4). Sie schlagen eine spezielle Regularisierungsform für Bestärkungslernen vor, die adaptive Fehlerinjektion. Sie verbessert die Generalisierung, ohne die Gradientenberechnung zu beeinträchtigen. Des Weiteren verwenden sie einen Informations-Flaschenhals (Abschn. 4.7.6) und zeigen, dass dieser ebenfalls eine besonders gute Regularisierungstechnik für Bestärkungslernen ist.

Als Anwendung untersuchen sie die Aufgabe, dass ein Agent eine Reihe von Räumen durchqueren muss, um ein Ziel (das grüne Quadrat) zu finden (Abb. 8.25 links). Der Agent hat diskrete Aktionen zur Verfügung: Rechts- oder Linksdrehung, Vorwärtsbewegung, Tür öffnen oder schließen. Diese Umgebung wird jedes Mal zufällig neu generiert. Der Agent muss also von der einzelnen Umgebung abstrahieren und auf ein ganzes Bündel von Aufgaben generalisieren.

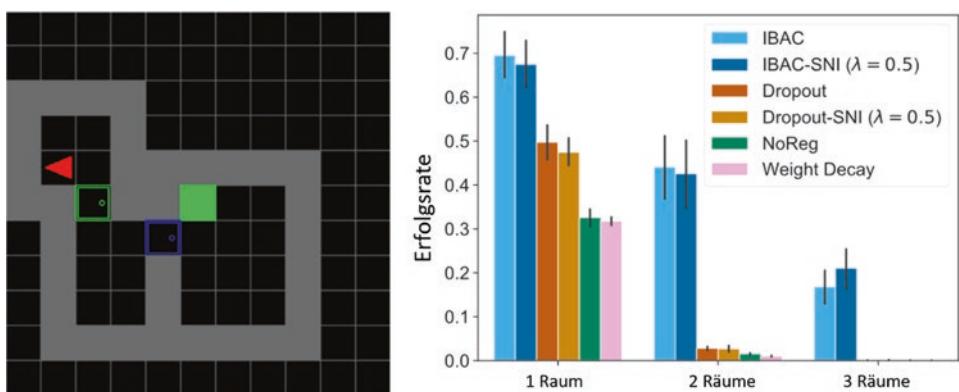
**Abb. 8.23** Ein Roboter lernt das Aufstehen mithilfe von Strategiegradienten. Die Stellungen 1 bis 6 werden im Verlauf des Aufstehens angenommen.  
Bildnachweis im Anhang



**Abb. 8.24** Ein Roboter lernt das Gehen durch Strategiegradienten. Bildnachweis im Anhang



Auf der rechten Seite von Abb. 8.25 wird die Häufigkeit gezeigt, mit der ein trainierter Agent in neuen Umgebungen erfolgreich das Ziel findet. Die üblichen Ansätze (Weight Decay, Dropout, keine Regularisierung) sind bei drei Räumen nicht in der Lage, das Ziel zu finden. Hingegen gelangen die neuen Regularisierungstechnik IBAC-SNI (Information Bottleneck Actor Critic- SelectiveNoise Injection) in 21 % der Fälle zum Ziel. Die Schwierigkeit dieser scheinbar einfachen Aufgabe ergibt sich aus ihren Generalisierungsanforderungen: Da die Umgebung in jeder Episode zufällig generiert wird, wird jeder Zustand



**Abb. 8.25** Auf der linken Seite wird die Aufgabe dargestellt: das grüne Quadrat finden. Der Agent (rotes Dreieck) hat die Aufgabe, das Ziel (grünes Quadrat) zu finden. Die grauen Quadrate sind Wände und die farbigen Boxen sind Türen. (Igl et al. 2019)

sehr selten beobachtet, insbesondere bei Umgebungen mit mehreren Räumen, bei denen eine Generalisierung erforderlich ist, um das Lernen zu ermöglichen.

### 8.4.5 Anwendung in Dota2

Die ersten Anwendungen des Bestärkungslernens waren Brettspiele wie etwa Schach, wo es eine sehr einfache Umgebung (Spielfeld) und wenige Aktionsmöglichkeiten (Spielzüge) gibt. Soll der Detailreichtum und die Variabilität der realen Welt erfasst werden, so sind derzeit Videospiele die beste Annäherung. Sie erfordern das Zusammenspiel von mehreren Akteuren, haben lange Planungshorizonte, in denen das Ziel erreicht werden muss und eine Welt, von der nur ein kleiner Teil beobachtbar ist.

Dota2 ist ein Echtzeit-Strategiespiel und wird zwischen zwei Mannschaften von je fünf Spielern gespielt. Es ist eins der populärsten Videospiele. Zudem läuft es auf Linux und hat eine Programmschnittstelle, was die Steuerung mit einem TNN erleichtert.

Open AI verwendet für Dota2 sich selbst optimierende Algorithmen auf Basis neuronaler Netze. Für jeden der fünf Spieler wird ein Agent, ein unabhängiges TNN trainiert. Jeder Agent ist ein LSTM (Abschn. 6.4) mit einem verdeckten Vektor der Länge 4096. Er hat Zugriff auf etwa 20.000 Zahlen, welche den für ihn sichtbaren Teil der Spielwelt repräsentieren. Der Agent gibt Aktionen aus, die aus einer Liste von 8 Zahlen bestehen. Die Agenten agieren alleine auf Grund ihrer stochastischen Strategie. Sie handeln als Team, aber es gibt keine spezielle Kommunikationsmöglichkeit zwischen ihnen.

Im Training spielen zwei Mannschaften mit jeweils fünf Agenten gegeneinander. Jeder Agent hat eigene Fähigkeiten und entwickelt auch eine eigene Strategie. Dabei erzeugen diese Mannschaften täglich Spielverläufe von 180 Jahren Realdauer und nutzen 128.000 Prozessorkerne sowie 256 GPUs. Nach jedem Zeitintervall gibt es Belohnungen, z. B. eine positive Zahl, wenn eine Spielfigur eine Erfahrung macht und eine negative, falls z. B. ein Agent verwundet wird. Es wird die Proximal Policy Optimization (Abschn. 8.4.3) verwendet, um die Parameter des Netzes zu optimieren, d. h. im Mittel höhere Belohnungen zu erhalten. Die Agenten haben während des Trainings so viele Spiele absolviert, dass ein menschlicher Spieler dafür rund 45.000 Jahre benötigen würde.

Im April 2019 ist auf einer Veranstaltung in San Francisco die Software OpenAI-Five (OpenAI-Five 2018) gegen den amtierenden Weltmeister, das E-Sports-Team OG (Abb. 8.26), angetreten. Es gab einige Einschränkungen: Es waren nur 17 der verfügbaren Agententypen erlaubt, weil OpenAI-Five nur so viele implementiert hatte. Die anderen Einschränkungen gaben dem menschlichen Team einen Vorteil. So konnten sich die Agenten von OpenAI-Five im Gegensatz zu den Menschen nicht absprechen. Außerdem wurden ihre Aktionsfolge auf das menschliche Tempo verlangsam. OpenAI-Five hat gegen OG mit 2:0 Spielen gewonnen (Steinlechner 2019). Insgesamt hat sich damit erstmals ein System der Künstlichen Intelligenz in einem komplizierten und langdauernden Strategiespiel gegen die amtierenden Weltmeister durchgesetzt (Abb. 8.27).



**Abb. 8.26** Szene aus dem Dota2 Spiel zwischen dem OpenAI-Team von Bots und dem menschlichen OG-Team. Bildnachweis im Anhang

## 8.5 Selbstfahrende Autos

Ein selbstfahrendes Auto muss seine Umgebung mit Kameras, Radar oder anderen Sensoren beobachten. Das Steuerungssystem – der Agent – muss die Umgebung des Fahrzeugs verstehen, die Position des Fahrzeugs in der Umgebung erkennen und die nächsten Schritte planen, welche erforderlich sind, um das Ziel zu erreichen. Um die Verarbeitung der Sensorinformationen darstellen zu können, müssen zunächst einmal die Eigenschaften der Sensoren verdeutlicht werden. Anschließend kann ein Einsatz von TNN zur Strategieoptimierung diskutiert werden.



**Sebastian Thrun** studierte Informatik an der Universität Bonn und promovierte über Robotik: „Ich habe mich mit Robotik beschäftigt, weil das der beste Weg für mich war, Intelligenz zu erforschen“. 2003 wechselte er zur Stanford Universität. 2005 gewann sein Team den DARPA Grand Challenge, ein Rennen für Roboterautos. 2011 wechselte er zu Google, um die Entwicklung selbstfahrender Autos zu leiten. 2011 bot er seinen Kurs zur Einführung in die Künstliche Intelligenz mit großem Erfolg im Internet an. Daraufhin gründete er mit Kollegen die Onlineuniversität Udacity: „Ausbildung sollte von der positiven Seite des Gaming lernen – Belohnung, Leistung, Spaß“. Er lieferte wichtige wissenschaftliche Beiträge zu probabilistischer Robotik und zu selbstfahrenden Autos.

**Abb. 8.27** Sebastian Thrun \* 14.05.1967 in Solingen, ist ein Pionier für Transferlernen und selbstfahrende Autos. Bildnachweis im Anhang



**Abb. 8.28** Annotierte Pixel eines Kamerabildes mit erkannter Fahrbahn, Randstreifen, Bäumen, Schildern etc. (Fridman 2019). Bildnachweis im Anhang

### 8.5.1 Sensoren selbstfahrender Autos

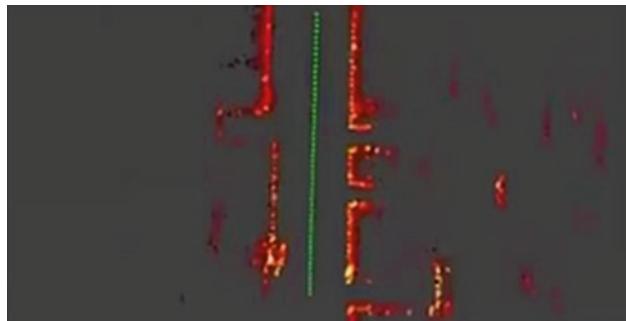
Selbstfahrende Autos haben unterschiedliche Sensoren, um die Umgebung und das Verkehrsgeschehen zu beobachten. Die wichtigsten sind Video, Radar und Lidar (light detection and ranging).

Videokameras können heute mit hoher Bildauflösung und Bildwiederholrate Videos der Umgebung aufnehmen und sind günstige und weit verbreitete Sensoren. Allerdings haben sie Probleme bei Nacht und schlechtem Wetter, da dann die Lichtverhältnisse unzureichend sind. TNN können die Kamerabilder verarbeiten, einzelne Objekte erkennen und die entsprechenden Bildpixel annotieren (Abb. 8.28). Diese Techniken wurden in Abschn. 5.8.2 vorgestellt.

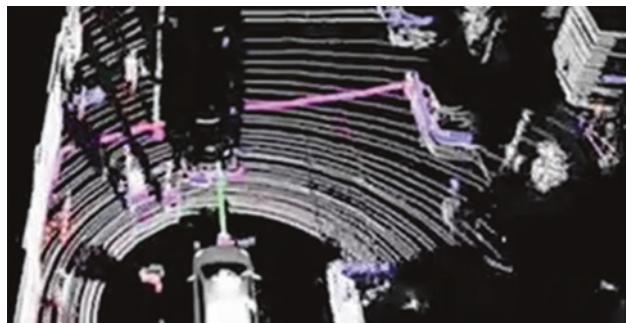
Radarsensoren senden Pulse von Radiowellen aus und können aus den reflektierten Signalen die Entfernung von angepeilten Objekten und deren Geschwindigkeit messen. Sie funktionieren gut auch bei schlechtem Wetter oder bei Nacht, haben aber eine schlechte Auflösung. Abb. 8.29 zeigt die von einem Radar gesammelte Information an einer Straße.

Lidar ist ein Verfahren, um die Entfernung von Objekten zu messen. Dabei werden Infrarot-Laserlichtimpulse in die Umgebung gesendet. Die Rücklaufzeiten und die

**Abb. 8.29** Vom Radarsensor eines Autos rekonstruiertes Radarbild einer Straße (Fridman 2019). Bildnachweis im Anhang



**Abb. 8.30** Auflösung und Informationsdichte bei Lidar bei der gleichen Perspektive wie in Abb. 8.29 (Quelle: (Fridman 2019)). Bildnachweis im Anhang



Wellenlängen werden dann genutzt, um die Entfernung der Objekte zu bestimmen. Durch bewegliche Spiegel lassen sich pro Sekunde Millionen von Impulsen aussenden, womit man 3D-Entfernungsprofile der Umgebung eines Fahrzeugs in der gleichen Situation wie in Abb. 8.29. Die Auflösung ist wesentlich besser als bei Radar, aber nicht ganz so gut wie bei Videokameras. Um 3D-Informationen aus Kamerabildern zu rekonstruieren, benötigt man aufwendige maschinelle Lernverfahren, während diese bei Lidar exakt gemessen werden. Allerdings waren Lidar-Sensoren bislang sehr teuer, die Preise sind aber in der letzten Zeit stark gefallen.

Die Abb. 8.31 zeigt die Vor- und Nachteile der einzelnen Sensoren. Hierbei sind auch Ultraschallsensoren bewertet, welche eine relativ schlechte Auflösung besitzen. Jedes selbstfahrende Auto ist mit einer Vielzahl unterschiedlicher Sensoren ausgestattet, die teilweise in unterschiedliche Richtungen weisen. Abb. 8.32 stellt die wichtigsten Sensoren und ihre Erfassungsbereiche dar.

## 8.5.2 Funktionalität eines Agenten fürs autonome Fahren

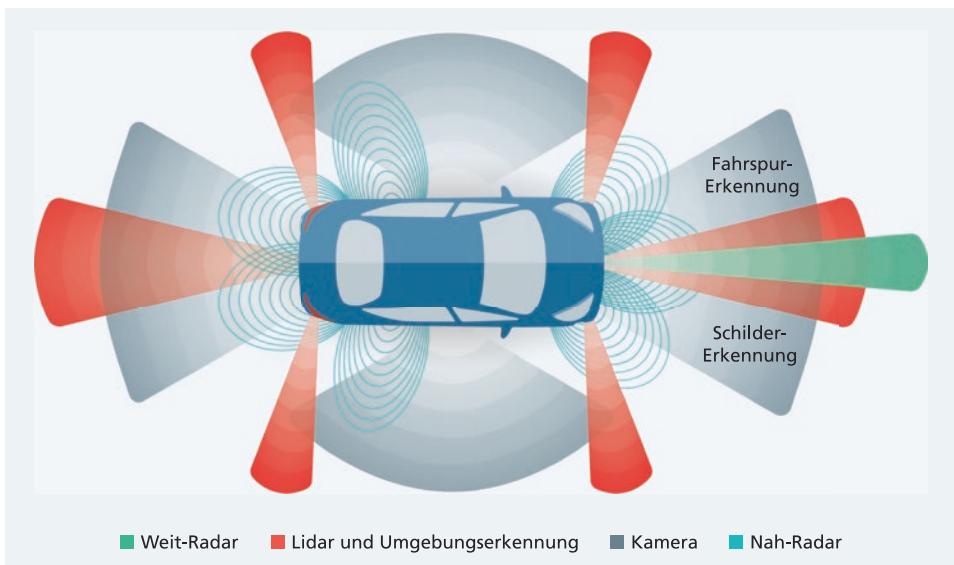
Der Agent eines selbstfahrenden Autos muss eine Reihe von Funktionen bereitstellen, die in Abb. 8.33 dargestellt sind. Mithilfe der Sensoren müssen unterschiedliche Dinge erkannt werden: die Fahrspur, Ampeln und Verkehrszeichen, Objekte wie Fußgänger, Radfahrer, Poller, andere Autos sowie der nutzbare Freiraum.

Die Entwickler der selbstfahrenden Autos halten die Details der Steuerungen geheim, um ihren Wettbewerbsvorteil nicht zu verlieren. Daher lassen sich nur relativ allgemeine Aussagen über die verwendeten Algorithmen treffen. Künstliche Intelligenz wird weitgehend in Wahrnehmungsalgorithmen verwendet, um „rohe“ Sensordaten zu verstehen. Aufgaben wie Kombination der Informationen unterschiedlicher Sensoren, Objekterkennung, Objektverfolgung, Objektfusion usw. gehören auch zur Wahrnehmung. Neueste Wahrnehmungsalgorithmen verwenden meist tiefe neuronale Netze (Mitrev 2019).

Die übergeordnete Instanz des Agenten ist die Routenplanung, die mithilfe von Karten die zu verfolgende Strecke bestimmt. Das Planungssystem kombiniert die verarbeiteten

**Abb. 8.31** Eigenschaften der unterschiedlichen Sensoren für autonomes Fahren auf einer Skala von 1 (gering, wenig) bis 5 (sehr hoch/gut).  
(Fridman 2019)

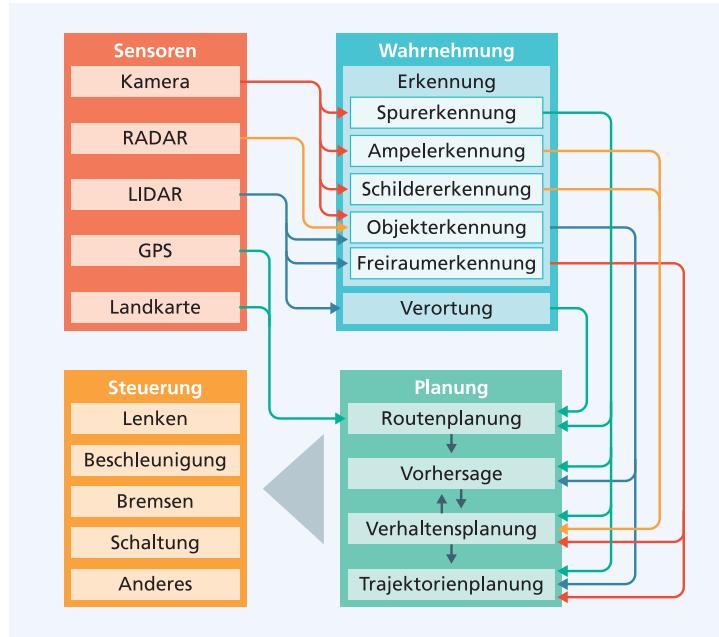
Eigenschaft	Lidar	Ultra-schall	Radar	Video
Reichweite	4	1	4	5
Auflösung	4	2	3	5
bei Nacht	5	5	5	1
im Hellen	5	5	5	4
Schnee/Regen	3	5	5	2
Farbe, Kontrast	1	1	1	5
Geschwindigkeitsmessung	4	1	5	1
Sensorgröße	1	5	5	5
Sensorkosten	1	5	5	5



**Abb. 8.32** Anordnung der Sensoren eines selbstfahrenden Autos

Informationen über die Umgebung (von den Sensoren) mit festgelegten Richtlinien und Vorgaben darüber, wie in der Umgebung navigiert werden soll. Bei der Planung werden häufig Karten und Regelkataloge verwendet, die „gültige“ Fahrvorgänge definieren. Ausgehend von der gewählten Route wird dann die nächste Verkehrssituation vorhergesagt und die Trajektorie sowie das Fahrverhalten geplant (Mitrev 2019).

Für die Planung können Methoden der Strategiegradienten (Abschn. 8.4) verwendet werden, welche die zu wählende Aktion  $a$  mithilfe einer geschätzten Verteilung  $\pi(a|s; w)$  aus dem derzeitigen Zustand (Sensorinformationen etc.) bestimmen. Der Parametervektor  $w$  dieses Modells kann aus Beobachtungsdaten geschätzt werden (Abschn. 8.4.2).



**Abb. 8.33** Komponenten eines Agenten für selbstfahrende Autos (Giacaglia 2019)

Der einfachste Ansatz verwendet stattdessen reine Planungstechniken wie die modell-prädiktive Steuerung MPC (Model Predictive Control), um Aktionen auszuwählen. Ist eine neue Beobachtung der Umgebung verfügbar, so berechnet der Agent einen Plan, der in Bezug auf das Modell optimal ist. Der Plan beschreibt alle Aktionen innerhalb eines festen Zeitfensters. Belohnungen jenseits des Zeitfensters können mithilfe von gelernten Wertfunktionen berücksichtigt werden. Der Agent führt dann die erste Aktion des Plans aus und verwirft den Rest. Ist die nächste Beobachtung verfügbar, so wird das Verfahren erneut angestoßen.

Das Steuerungssystem sorgt dafür, dass die vom Planungssystem abgeleiteten Absichten und Ziele in Maßnahmen umgesetzt werden. Das Steuerungssystem teilt der Hardware (den Aktuatoren) die erforderlichen Aktionen mit (Beschleunigung, Bremsen, Lenken etc.). Das Steuerungsmodul hat zudem das Ziel, die Fahrbewegungen gleichmäßig und angenehm verlaufen zu lassen.

Zwei der wichtigsten Akteure bei der Entwicklung selbstfahrender Autos sind Tesla und Waymo. Tesla setzt dabei auf günstige Sensoren, also Kameras und Radar, während Waymo zusätzlich auch Lidar nutzt, welches relativ genau die Abstände zu den Objekten der Umgebung erfassen kann. Während ein Tesla-Pkw im Schnitt 5 km fährt, bis der Nutzer eingreifen muss, ist dies bei Waymo erst nach mehr als 2000 km der Fall (Giacaglia 2019). Waymo liegt also derzeit bei der Entwicklung vollautomatisch fahrender Autos vorne.

### 8.5.3 Feintuning durch Simulation

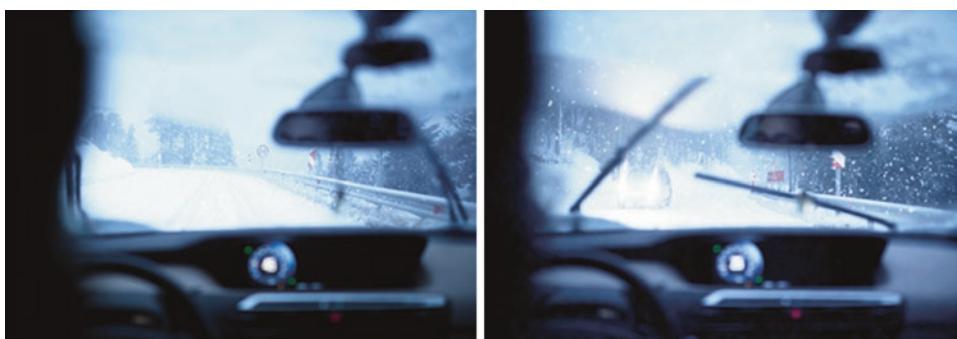
Der Agent eines selbstfahrenden Autos muss mit einer sehr großen Variationsbreite an Situationen zureckkommen. Abb. 8.34 zeigt eine verschneite, kurvige Straße in den Bergen. Der Agent muss in einer solchen Situation mit Gegenverkehr rechnen und entsprechend vorsichtig fahren. Insbesondere muss er sich so verhalten, dass er auch Fahrfehler der anderen Autofahrer ausgleichen und ggf. frühzeitig bremsen kann.

Um komplexe KI-Algorithmen zu erstellen, sind große Mengen an Sensordaten aus dem Fahrbetrieb erforderlich. Die Kosten für das Sammeln dieser Daten und insbesondere für das Sammeln ausreichender Daten für kritische (selten vorkommende) Fahrszenarien sind jedoch sehr hoch. Dies erschwert das Trainieren von KI-Modellen, die auch in Ausnahmesituationen angemessen reagieren müssen.

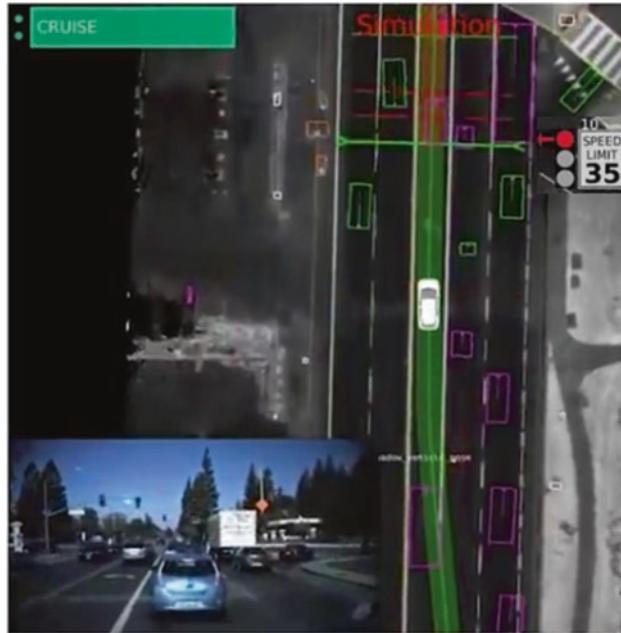
Ein Entwicklungsschwerpunkt von Waymo ist daher der Test seiner Software in einer simulierten Welt. Waymo entwickelte eine simulierte virtuelle Welt namens Carcraft, benannt nach dem Spiel World of Warcraft. Mit Carcraft wurden Tausende von Szenarien simuliert, um den Agenten zu testen und zu verbessern.

In Carcraft erstellte Waymo vollständig modellierte Versionen von Städten wie Austin, Mountain View, Phoenix etc. Es wurden verschiedene Szenarien in vielen simulierten Autos getestet – etwa 25.000 davon gleichzeitig. Insgesamt fahren die Autos in dieser virtuellen Welt ungefähr 16 Millionen km pro Tag (Abb. 8.35). Alleine im Jahr 2018 legten sie ungefähr 11,6 Milliarden virtuelle km zurück (Anguelov 2019). Dies ist viel mehr als die 16 Millionen km, die reale Waymo Autos auf öffentlichen Straßen gefahren sind.

Bansal et al. (2018) erklären im Detail, wie in dieser Simulationsumgebung nicht nur einfach Variationen von gefahrenen Strecken durchgespielt werden. Vielmehr werden für die beteiligten Akteure (Autos, Fußgänger, Fahrräder etc.) eigene Simulationsmodelle erstellt, die deren Verhalten in plausibler Weise steuern und ändern können. Manchmal geschieht dies auch durch Regelübertretung, z. B. wenn ein anderes Auto eine rote Ampel



**Abb. 8.34** Ein Auto fährt über eine verschneite Bergstraße. Plötzlich kommt ihm in einer Kurve ein anderer Wagen entgegen. Der Agent muss mit der Möglichkeit solcher Situationen rechnen. Bildnachweis im Anhang



**Abb. 8.35** Simulierte, durch den Agenten gesteuerte Fahrt mit der Carcraft-Simulationsumgebung. Hier können Variationen der vorherigen Fahrstrategie erprobt werden (Anguelov 2019). Bildnachweis im Anhang

überfährt oder ein Fußgänger unvermittelt die Fahrbahn überquert. Der Agent soll ein solches Verhalten in Betracht ziehen und darauf vorbereitet sein.

Es gibt mittlerweile Ansätze, bei denen der Agent gleichzeitig ein Simulationsmodell für die Umgebung und ein TNN zu Approximation der Strategie lernt (Ha und Schmidhuber 2018). Hierbei werden die für die Steuerung relevanten Details automatisch reproduziert und andere Teile weggelassen.

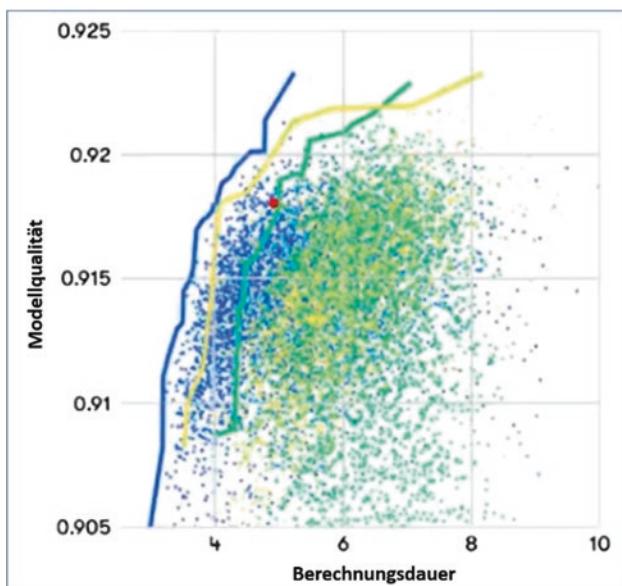
Eine große Rolle spielen tiefe neuronale Netze zur Sensorverarbeitung und Steuerung. Hier werden die üblichen CNN-Architekturen zur Bildverarbeitung und zur Kombination der verschiedenen Sensorsignale genutzt. Gao et al. (2018) präsentieren z. B. ein CNN, welches Kamera-Signale und Lidar-Informationen zu einer Umgebungsdarstellung fusioniert. Der Tesla-Agent nutzt eine Variante des neuronalen Netzes Google Inception mit der fünffachen Anzahl von Parametern, um Objekte in der Umgebung zu entdecken (Giacaglia 2019).

Häufig wird die Netzwerkarchitektur durch automatische Verfahren modifiziert und verbessert. Optimierungskriterium sind dabei die Genauigkeit der Vorhersage, die Robustheit gegen Störungen (Regen, Nebel etc.) sowie die Rechenkosten. Zoph et al. (2018) zeigen, wie man neue Architekturen zunächst für kleine Datenbestände strukturell optimieren kann, um dann die global beste Lösung durch Evaluation der besten 100 Lösungen auf der großen Datenmenge zu gewinnen (Abb. 8.36).

Es ist schwierig vorherzusagen, wann vollautonome Autos in größerer Zahl auf den Straßen fahren werden. Die Hersteller geben überwiegend an, bis 2021 solche Fahrzeuge an Kunden auszuliefern (Abb. 8.37). Fridman (2019) hat Anfang 2019 eine Umfrage auf Facebook und Twitter gestartet: „Welche Firma wird als erste mehr als 10.000 vollautonome Autos ausliefern, die auf öffentlichen Straßen ohne Sicherheitsfahrer fahren?“. 57 % der 3000 Antwortenden tippten auf Tesla, während 21 % der Teilnehmer Waymo vorne sahen. 14 % meinten, dass eine andere Firma die erste wäre und 8 % sagten, dass es niemand in den nächsten 50 Jahren schaffen könnte. Es bleibt also spannend.

Global kommen jährlich 1,35 Millionen Menschen im Straßenverkehr ums Leben. Damit stirbt alle 24 Sekunden ein Mensch bei einem Verkehrsunfall (DW 2018). Es gibt Schätzungen, dass diese Zahl bei autonomen Autos beträchtlich sinken wird. Bisher gab es zwei Todesfälle, die durch teilautonome Autos verursacht wurden, aber noch keinen Todesfall mit vollautonomen Autos (Murasaki 2019). Aber selbst die teilautonomen Tesla-Autos sind sicherer als normale Autos. Während es bei normalen Autos etwa alle 0,7 Millionen km zu einem Unfall kommt, fuhr ein Tesla-Auto mit aktivierter teilautonomer Steuerung im Durchschnitt 4,7 Millionen km, bis ein Unfall geschah. Die Anzahl der Unfälle wurde also um 85 % reduziert (Murasaki 2019).

**Abb. 8.36** Automatische Hyperparameter-Optimierung. Die vorherige beste Lösung (roter Punkt) wurde sowohl hinsichtlich der Modellqualität als auch der Berechnungsdauer entscheidend verbessert (Anguelov 2019). Bildnachweis im Anhang



**Abb. 8.37** Ankündigung der Automobilhersteller für die Verfügbarkeit von vollautonomen Autos für Konsumenten.<sup>1)</sup> auf Autobahnen,<sup>2)</sup> in Städten. Quelle: (Fridman 2019)

Hersteller	Jahr	Hersteller	Jahr
Tesla	2019	Volvo	2021 <sup>1)</sup>
Nissan	2020	BMW	2021
Honda	2020	Ford	2021
Toyota	2020 <sup>1)</sup>	Fiat	2021
Renault	2020 <sup>2)</sup>	Daimler	2020-25
Hyundai	2020 <sup>1)</sup>		

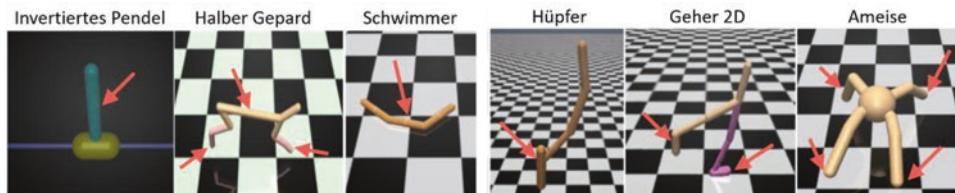
## 8.6 Zuverlässigkeit des Bestärkungslernens

### 8.6.1 Training in Simulationsumgebungen oft schwierig übertragbar

Pinto et al. (2017) stellen fest, dass die meisten Ansätze zum Bestärkungslernen schlecht anwendbar sind. Gründe hierfür sind:

- Der Unterschied zwischen simulierter Umgebung und realer Welt ist so groß, dass Modelle zum Strategiernen oft nicht übertragbar sind.
- Selbst wenn das Lernen in der realen Welt stattfindet, ist die Menge der verfügbaren Daten zu gering und die Generalisierung scheitert deswegen oft.

Die Autoren schlagen daher vor, einen Gegner (adversary) beim Training einzuführen, der zusätzliche Störkräfte und Destabilisierungen bis zu einer vorgegebenen Größe ausüben kann. Der Gegner wird darauf trainiert, Störkräfte gerade so in dem System anzuwenden, dass der Agent maximal gestört wird (Abb. 8.38). Der Lernalgorithmus muss diese Störungen ausgleichen und wird damit automatisch robuster. Das Modell hat sehr große Ähnlichkeit zu den generativen adversarialen Modellen im Bildbereich, die später diskutiert werden (Abschn. 9.1.2). Die Ergebnisse der Experimente sind ermutigend, und es werden wesentlich robustere Kontrollstrategien gelernt.



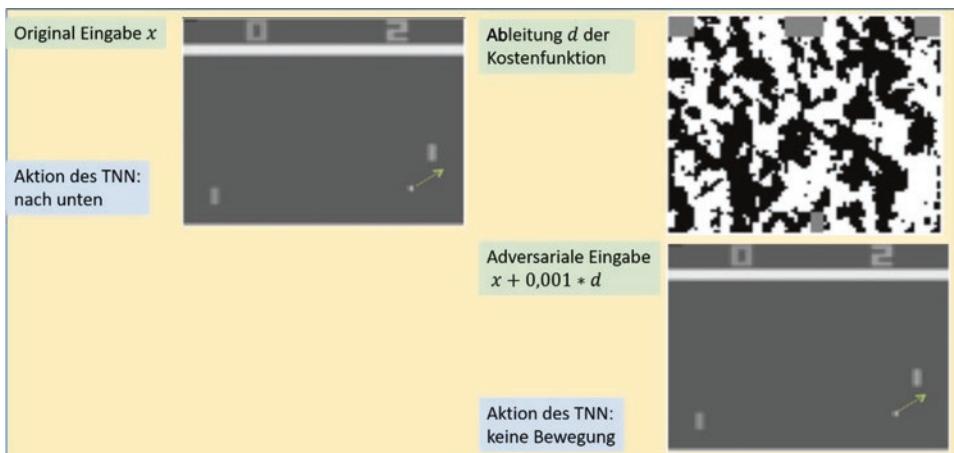
**Abb. 8.38** Bei den verschiedenen Agenten werden beim adversarialen Bestärkungslernen zufällig gesteuerte Störkräfte (Pfeile) ausgeübt. Hierdurch werden die Agenten robuster (Pinto et al. 2017). Bildnachweis im Anhang

### 8.6.2 Adversariale Attacken auf Modelle zum Bestärkungslernen

Wenn ein Modell zum Bestärkungslernen seine Eingaben in Form von Pixelbildern erhält, so können diese Bilder, wie in Abschn. 5.12.2 gezeigt, unmerklich verändert werden, um TNN zu täuschen, welches die Q-Funktion approximiert (Abschn. 8.2.7). In Abb. 8.39 wird ein adversariales Beispiel für das Atari-Spiel Pong gezeigt. Der Zustand des Spiels wird durch ein Bild repräsentiert, der Pfeil zeigt die Bewegungsrichtung des Balles. Die adversariale Änderung ist das Vorzeichen der Ableitung der Kostenfunktion nach  $x$ . Sie wird mit einem kleinen Faktor multipliziert und ist im Ergebnis nicht vom Betrachter erkennbar. Hierdurch wird die vom TNN ausgegebene Aktion komplett geändert.

Chen et al. (2019) weisen darauf hin, dass diese Änderungen bei jedem Schritt ausgerechnet werden können und der Agent deshalb nicht die richtigen Aktionen ergreift. Es gibt eine große Anzahl weiterer Strategien, um Modelle zum Bestärkungslernen zu stören, darunter auch Ansätze, bei denen die Modellarchitektur nicht bekannt sein muss.

Für insgesamt acht Angriffsmethoden untersuchen Chen et al. (2019) die Wirksamkeit von 20 verschiedenen Abwehrstrategien. Es stellt sich heraus, dass es für die meisten Attacken wirksame Gegenmaßnahmen gibt. Zudem erfordert die Mehrzahl der Angriffsmethoden den Zugang zu dem Modell. Black-Box-Attacken sind selten, und für den Angrei-



**Abb. 8.39** Das adversariale Eingabebild (unten rechts) ist die Summe des originalen Eingabebildes  $x$  (links) und des Vorzeichens der Ableitung der Kostenfunktion  $d$  (oben rechts). Da  $d$  mit dem Faktor 0,001 multipliziert wird, ist die Änderung auf der adversarialen Eingabe nicht sichtbar. Das TNN wählt bei der linken Eingabe die Aktion „nach unten“, aber bei der rechten Eingabe die Aktion „keine Bewegung“ (Quelle: Chen et al. 2019). Bildnachweis im Anhang

fer sehr schwierig in der Praxis durchführbar. Die Autoren vermuten, dass wegen der verstärkten Forschungsaktivität in diesem Bereich in Zukunft sehr zuverlässige Systeme zum Bestärkungslernen verfügbar sein werden.

## 8.7 Zusammenfassung und Trends

In vielen Anwendungsbereichen sind über einen gewissen Zeitraum eine Reihe von Aktionen zu bestimmen mit dem Ziel, eine möglichst hohe Summe von Belohnungen über die Zeit zu erreichen. Ein Modell (Agent) kann diese Aktionen aufgrund der jeweils aktuellen Informationen über die Situation (Zustand) auswählen. Beispiele sind Brettspiele, Robotersteuerungen und selbstfahrende Autos. TNN werden hierbei benutzt, um einem Zustand eine Summe von erwartbaren Belohnungen zuzuordnen und damit die beste Aktion zu bestimmen. Sie sind besonders geeignet, weil sie die dahinterliegende Funktion gut approximieren können. Eine Q-Funktion  $Q(s, a)$  schätzt für jedes Zustand-Aktionspaar eine erwartbare Belohnungssumme. Für aufeinanderfolgende Zustände kann man daraus eine Gleichung ableiten, mit deren Hilfe ein Q-Modell trainiert werden kann.

Eine stochastische Strategie ist für Entscheidungssituationen mit Zufallseinflüssen geeignet und bestimmt für jeden Zustand eine optimale Wahrscheinlichkeitsverteilung über den möglichen Aktionen. Die zugehörigen Strategiegradienten-Modelle berechnen aus einer Stichprobe von Modellverläufen den Gradienten und können so das Modell trainieren. Im Unterschied zu bisherigen TNN werden die Trainingsdaten mit Hilfe der simulierten oder realen Umgebung während des Trainings erzeugt. Im Bereich der Spiele haben TNN nachhaltige Erfolge erzielt und schlagen jeden Menschen auf beliebigen Brettspielen (Abschn. 2.6.1). Aber auch bei komplexen Videospielen mit Teams von mehreren Spielern haben TNN jüngst das Weltmeisterteam geschlagen (Abschn. 8.4.5). Allerdings handelt es sich hier um simulierte Umgebungen, bei denen es möglich war, Trainingsdaten mit Jahrtausenden menschlicher Spielerfahrung zu erzeugen und zu verwenden. Eine prominente Anwendung sind autonome Fahrzeuge (Abschn. 8.5), welche derzeit von vielen Firmen entwickelt werden. Es ist noch nicht klar, wie erfolgreich diese Ansätze bei realen Entscheidungsproblemen im täglichen Leben sind.

### Trends

- Mittlerweile wird diskutiert, Roboter durch spielerische Aktivitäten auf ihre Aufgaben vorzubereiten. Schon Kinder üben komplexe Verhaltensmuster durch spielerische Beschäftigung mit ihrer Umgebung. Hierbei erfahren sie die Eigenschaften der Objekte in der Umgebung und können dieses Wissen über Transferlernen zur Lösung neuer Aufgaben verwenden. Lynch et al. (2019) zeigen, das über spielerische Aktionen trainierte Roboter komplexe, vorgegebene Aufgaben zu

85 % erfolgreich ausführen können, während Roboter, die Vorführungen von Experten zum Training verwendeten, nur bei 70 % der Versuche erfolgreich waren.

- Inspiriert durch den Erfolg von AlphaFold (Abschn. 2.1.2) ist zu erwarten, dass Techniken des Bestärkungslernens echte Anwendungen im industriellen Umfeld finden werden, z. B. beim Design von Medikamenten, Fahrzeugsteuerung, Verkehrslenkung etc.
- Im Sprachbereich hat sich die Verwendung von unüberwacht trainierten Embeddings durchgesetzt, welche dann an die eigentliche Aufgabe angepasst werden. Es ist abzusehen, dass derartige Embeddings auch in Modellen des Bestärkungslernens verwendet werden.
- Im Bestärkungslernen gibt es das Problem der Überanpassung, nicht zuletzt, weil man das Modell auf den gleichen Daten trainiert und testet. Daher werden Ansätze entwickelt, welche diese Art von Test vermeiden.
- In Situationen mit mehreren Agenten ist es vorteilhaft, wenn Agenten miteinander kommunizieren. Derzeit werden verschiedene Benchmarkaufgaben aufgebaut, in denen eine solche Kommunikation erprobt werden kann.
- Die Verwendung eines trainierten Agenten in einer geänderten Umgebung ist schwierig, weil der Agent die neuen Ziele und Umgebungsreaktionen neu trainieren muss. Zhong et al. (2019) trainieren einen Agenten, der seine Ziele und die Beschreibungen der Umgebung einem Textdokument entnehmen muss. Dies ist die Grundlage für ein Transferlernen über neue „Spielregeln“, mit denen der Agent neue Aufgaben erhält.

---

## Literatur

- Anguelov, Drago. 2019.** Taming the Long Tail of Autonomous Driving Challenges. [Online] 2019. [Zitat vom: 05. 06. 2019.] <https://www.youtube.com/watch?v=Q0nGo2-y0xY>.
- Badia, A. P., et al. 2020.** Agent57: Outperforming the Atari Human Benchmark. [Online] 2020. [Zitat vom: 13. 04. 2020.] arXiv preprint arXiv:2003.13350.
- Bansal, M., Krizhevsky, A. und Ogale, A. 2018.** Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. [Online] 2018. [Zitat vom: 05. 06. 2019.] arXiv preprint arXiv:1812.03079.
- Bellman, R.E. 1957.** *Dynamic Programming*. s.l. : Princeton University Press, Princeton, NJ. Republished 2003: Dover, ISBN 0-486-42809-5, 1957.
- Bögeholz, H. 2017.** Künstliche Intelligenz: AlphaGo besiegt Ke Jie zum dritten Mal. [Online] 27. 05. 2017. [Zitat vom: 08. 05. 2017.] <https://www.heise.de/newsticker/meldung/Kuenstliche-Intelligenz-AlphaGo-besiegt-Ke-Jie-zum-dritten-Mal-3726711.html>.
- Brockman, G., et al. 2016.** Openai gym. [Online] 2016. [Zitat vom: 05. 05. 2019.] arXiv preprint arXiv:1606.01540.
- Chen, T., et al. 2019.** Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity*, 2(1), 11. 2019.

- Dickmanns, E. D., und Graefe, V.** 1988. Dynamic monocular machine vision. *Machine vision and applications*, 1(4), 223–240. 1988.
- DW.** 2018. WHO: Weltweit immer mehr Verkehrstote. *Deutsche Welle*. [Online] 07. 12. 2018. [Zitat vom: 06. 06. 2019.] <https://www.dw.com/de/who-weltweit-immer-mehr-verkehrstote/a-46624276>.
- Fridman, Lex.** 2019. Self driving cars: state of the art 2019. [Online] 2019. [Zitat vom: 03. 06. 2019.] <https://www.youtube.com/watch?v=sRxaMDDMWQQ>.
- Gao, H., et al.** 2018. Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, 14(9), 4224–4231. 2018.
- Giacaglia, G.** 2019. Self-Driving Cars. [Online] 2019. [Zitat vom: 05. 06. 2019.] <https://medium.com/@giacaglia/self-driving-cars-f921d75f46c7>.
- Ha, D. und Schmidhuber, J.** 2018. Recurrent World Models Facilitate Policy Evaluation. [Online] 27. 03. 2018. [Zitat vom: 15. 09. 2019.] <https://worldmodels.github.io/>.
- Hogget, R.** 2019. 1951 – SNARC Maze Solver – Minsky/Edmonds. [Online] 2019. [Zitat vom: 27. 01. 2020.] <http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/>.
- Igl, M., et al.** 2019. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in Neural Information Processing Systems* (pp. 13956–13968). 2019.
- Jin, C., et al.** 2018. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems* (pp. 4863–4873). 2018.
- Levine, S.** 2018. Policy Gradients. *CS 294-112: Deep Reinforcement Learning*. [Online] 2018. [Zitat vom: 06. 05. 2019.] <http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf>.
- Li, Y.** 2018. Deep reinforcement learning. [Online] 2018. [Zitat vom: 08. 05. 2019.] arXiv preprint arXiv:1810.06339.
- Lynch, C., et al.** 2019. Learning Latent Plans. *arXiv preprint arXiv:1903.01973*. [Online] 2019. [Zitat vom: 15. 09. 2019.] <https://learning-from-play.github.io/>.
- Mitrev, D.** 2019. Who Leads the Self-Driving Cars Race? State-of-Affairs in Autonomous Driving. [Online] 30. 01. 2019. [Zitat vom: 06. 06. 2019.] <https://neurohive.io/en/state-of-the-art/self-driving-cars/>.
- Mnih, V., et al.** 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529. 2015.
- Murasaki, Muralasa.** 2019. How many people have died in self-driving cars? [Online] 21. 05. 2019. [Zitat vom: 06. 06. 2019.] <https://www.quora.com/How-many-people-have-died-in-self-driving-cars>.
- OpenAI-Five.** 2018. OpenAI Five. *OpenAI*. [Online] 2018. [Zitat vom: 20. 04. 2020.] <https://openai.com/blog/openai-five/>.
- Osband, I., et al.** 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems* (pp. 4026–4034). 2016.
- Pinto, L., et al.** 2017. Robust Adversarial Reinforcement Learning. *Proceedings of the 34th International Conference of Machine Learning, PMLR* 70. 2017.
- Schulman, J., et al.** 2015. High-dimensional continuous control using generalized advantage estimation. [Online] 2015. [Zitat vom: 08. 05. 2019.] arXiv preprint arXiv:1506.02438.
- Schulman, J., et al.** 2017. Proximal policy optimization algorithms. [Online] 2017. [Zitat vom: 08. 05. 2019.] arXiv preprint arXiv:1707.06347.
- Silver, D., et al.** 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587), 484. 2016.
- Smith, C.S.** 2020. Computers Already Learn from Us. But Can They Teach Themselves? *New York Times*. 2020, 08.04.2020.

- Steinlechner, P. 2019.** Open AI besiegt die Weltmeister. *Golem*. [Online] 19. 04. 2019. [Zitat vom: 08. 05 2019.] <https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister-1904-140710.html>.
- Sutton, R. S., et al. 2000.** Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* (pp. 1057–1063). 2000.
- Watkins, C. J. und Dayan, P. 1992.** Q-learning. *Machine learning*, 8(3–4), 279–292. 1992.
- Watkins, C.J.C.H. 1989.** Learning from Delayed Rewards. *PhD thesis, Cambridge University, Cambridge, England*. 1989.
- Wikiludia. 2018.** Schere, Stein, Papier. [Online] 2018. [Zitat vom: 06. 05 2019.] <https://wikiludia.mathematik.uni-muenchen.de/wiki/index.php?title=Stein-Schere-Papier>.
- Yang, S., et al. 2019.** LRW-1000: A Naturally-Distributed Large-Scale Benchmark for Lip Reading in the Wild. *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)* (pp. 1–8). IEEE. 2019.
- Yoon, C. 2019.** Understanding Actor Critic Methods and A2C. [Online] 2019. [Zitat vom: 08. 05 2019.] <https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f>.
- Zhong, V., Rocktäschel, T. und Grefenstette, E. 2019.** RTFM: Generalising to New Environment Dynamics via Reading. *International Conference on Learning Representations*. 2019.
- Zoph, B., et al. 2018.** Learning transferable architectures for scalable image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8697–8710). 2018.



# Kreative Künstliche Intelligenz und Emotionen

9

Kreativität ist eine Fähigkeit, die wir normalerweise als einzigartig menschlich betrachten. Sie erschafft etwas, das neu oder originell ist und in vielen Fällen als nützlich und brauchbar bewertet wird. Computer werden meist wahrgenommen als Maschinen, die stur ein vordefiniertes Programm abarbeiten, und daher nichts Überraschendes zustande bringen können. Das folgende Kapitel soll zeigen, dass tiefe neuronale Netze auf kreative Art neuartige Bilder, Texte und Musik erzeugen können. Darüber hinaus wird dargelegt, dass Computerprogramme die Emotionen von Menschen erkennen und in gewissem Umfang darauf adäquat und kreativ reagieren können.

---

## 9.1 Bilder erzeugen mit generativen adversarialen Netzen (GAN)

### 9.1.1 Fälscher und Kunstexperte

Nehmen wir an, dass wir mithilfe von TNN neue Bilder im Stil eines Künstlers, z. B. Vincent van Gogh, erzeugen möchten. Wir könnten versuchen, ein Generatornetz (Fälscher) mithilfe einer Sammlung von van Gogh-Bildern so zu trainieren, dass es van Gogh-Bilder erzeugen kann. Dabei sollten aber nicht die gleichen Bilder wie in der Trainingsmenge generiert werden, sondern alternative Bilder mit neuen Motiven, die dem Stil von van Gogh entsprechen. Allerdings wäre es sehr schwierig festzulegen, was noch dem Stil entspricht und was nicht – die Festlegung einer Verlustfunktion für solch ein Modell ist sehr diffizil.

Viel einfacher wäre es natürlich, wenn wir ein Klassifikationsmodell hätten, das in der Lage wäre, einen echten Van Gogh von einer Fälschung zu unterscheiden. Dieses Diskriminatornetz sollte also als ein auf van Gogh spezialisierter Kunstsachverständiger fungieren und müsste anhand von echten und gefälschten Bildern trainiert werden.

### 9.1.2 Generator und Diskriminatator

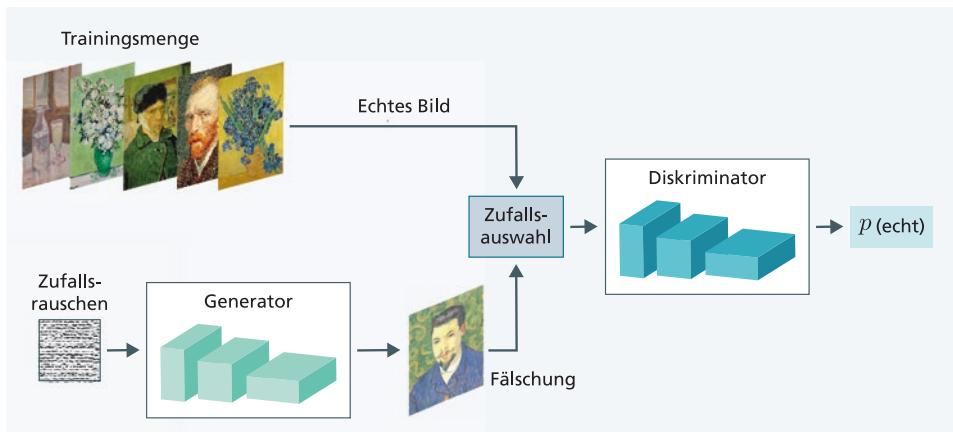
Das generative adversariale Netz (GAN) enthält einen solchen Generator und einen Diskriminatator und versucht, diese beiden Netze gleichzeitig zu trainieren. Wie in Abb. 9.1 gezeigt, wird eine Trainingsmenge mit Bildern eines Genres benötigt, z. B. von van Gogh. Das Generatornetz  $G(x)$  erhält einen Vektor  $x$  von Zufallszahlen und erzeugt hieraus ein Bild  $b$ . Das Diskriminatornetz  $D(b)$  erhält zufallsgesteuert  $b$ , ein echtes Bild oder eine Fälschung, und muss diese beiden Fälle unterscheiden, indem es eine Wahrscheinlichkeit für die Echtheit abschätzt (Goodfellow et al. 2014).

Beide Netze sind also Gegner in einem Spiel: der Generator soll Fälschungen erzeugen, die vom Diskriminatator nicht von echten Bildern unterschieden werden können; der Diskriminatator soll die Bilder des Fälschers mit hoher Sicherheit erkennen können.

Beide Netze werden zu Beginn zufällig initialisiert. Meist lernt der Diskriminatator schnell die ersten Fälschungen von den echten Bildern zu unterscheiden. Nach einem längeren Spielverlauf kann der Generator dann sehr gute Fälschungen generieren, wenn er die Stilmerkmale der Trainingsmenge solange gelernt hat, bis der Diskriminatator diese nicht mehr von echten Bildern unterscheiden kann.

### 9.1.3 Optimierungskriterium für Generator und Diskriminatator

Der Generator  $G(x)$  ist ein TNN mit dem Parametervektor  $w_g$ . Es erhält einen Zufallsvektor  $x$  als Eingabe und erzeugt eine Fälschung  $c$ , d. h.  $c = G_{w_g}(x)$ . Das Diskrimina-



**Abb. 9.1** Generatives adversariales Netzwerk zur Erzeugung von van Gogh-Bildern. Das Generator-Netzwerk erhält Zufallszahlen als Eingabe und erzeugt ein synthetisches Bild. Das Diskriminatoren-Netzwerk erhält zufallsgesteuert entweder ein echtes Bild aus der Trainingsmenge oder ein synthetisches Bild. Er muss die Wahrscheinlichkeit abschätzen, mit der es sich um ein echtes Bild bzw. eine Fälschung handelt. Beide Netze werden abwechselnd trainiert. Bildnachweis im Anhang

tor-Netz  $D(b)$  hat den Parametervektor  $w_d$ , erhält ein Bild  $b$  als Eingabe und berechnet die Wahrscheinlichkeit  $p(b \text{ ist echt}) = D_{w_d}(b)$ . Nimmt man  $n$  echte Bilder  $b_i$  aus der Trainingsmenge und wurden  $m$  Fälschungen  $c_j = G(x_j)$  erzeugt, so hat der Diskriminatator die Zielfunktion (Goodfellow et al. 2014)

$$J_D(w_d) = (D_{w_d}(b_1) * \dots * D_{w_d}(b_n))^{1/n} * ((1 - D_{w_d}(c_1)) * \dots * (1 - D_{w_d}(c_m)))^{1/m}$$

weil er die Wahrscheinlichkeit für die echten Bilder erhöhen und für die Fälschungen reduzieren möchte. Wie bei der logistischen Regression wendet man auf die Formel noch den Logarithmus an (Abschn 3.5.2) und erhält eine Summe, welche einfacher berechnet werden kann. Der Diskriminatator kann für dieses Optimierungskriterium mit dem stochastischen Gradientenanstieg optimiert werden.

Der Generator hingegen möchte die Wahrscheinlichkeiten für die gefälschten Bilder erhöhen, kann aber nur seine Parameter  $w_g$  beeinflussen. Allerdings kennt er die interne Struktur des Diskriminators. Daher optimiert er bei gegebenem  $w_d$  das folgende Kriterium

$$J_G(w_g) = (D_{w_d}(G_{w_g}(x_1)) * \dots * D_{w_d}(G_{w_g}(x_m)))^{1/m}$$

weil er  $p(\text{echt})$  für die Fälschungen maximieren möchte. Durch Anwendung des Logarithmus wird das Kriterium ebenfalls in eine Summe transformiert. Der Generatorparameter  $w_g$  kann für dieses Optimierungskriterium bei festem  $w_d$  mit dem stochastischen Gradientenanstieg optimiert werden. In der Praxis wechselt sich das Training beider Netze ab: der Diskriminatator wird mit Minibatches von echten und gefälschten Bildern optimiert und anschließend der Generator mit Minibatches von gefälschten Bildern.

Erste Experimente mit GANs verwendeten einfache Generator- und Diskriminatormodelle. Mittlerweile sind höhere Rechenleistungen verfügbar, sodass tiefe CNNs genutzt werden können. Radford et al. (2016) schlagen vor, die in Abb. 9.2 beschriebene Architekturmerkmale des Deep Convolutional GAN (DCGAN) zu nutzen. Trotzdem ist das Training von GANs besonders schwierig. Chintala et al. (2018) geben daher einige Hinweise zum erfolgreichen Vorgehen.

### 9.1.4 Die Ergebnisse generativer adversarialer Netze

(Radford et al. 2016) definieren ein DCGAN mit fünf Schichten, welche Felder der Größe  $4 \times 4, 8 \times 8, 16 \times 16, 32 \times 32$  und  $64 \times 64$  erzeugen. Sie trainierten das Netz auf Fotos

**Abb. 9.2** Empfohlene Architekturmerkmale für GANs

Der Generator sollte folgendes verwenden:

- abgeschnittene lineare ReLU-Aktivierungen, außer in der letzten Schicht
- möglichst wenige vollverbundene Schichten.
- Transposed Convolution-Schichten (Abschn. 5.8.4) zur Erhöhung der Bildauflösung
- Batchnormregularisierung

Der Diskriminatator sollte folgendes nutzen:

- Convolution-Schichten mit Schrittweite  $>1$  anstatt Pooling-Schichten
- Batchnormregularisierung
- möglichst wenige vollverbundene Schichten
- Leaky-ReLU Aktivierungen

von Schlafzimmern aus dem Large-Scale Scene Understanding (LSUN)-Datensatz mit etwa drei Millionen Trainingsbeispielen. Dabei wurden die grundlegenden Eigenschaften von Schlafzimmerbildern gelernt, aber die Geometrie der Möbel und ihre Perspektive waren oft nicht plausibel.

Karras et al. (2018) stellten ein GAN vor, welches im Laufe des Trainings die Feldgrößen sowohl des Generators als auch des Diskriminators anwachsen ließ. Das beschleunigt das Training und erlaubt auch sehr hohe Auflösungen von  $1024 \times 1024$ . Zudem schlugen sie weitere Verbesserungen vor, welche die Variabilität der erzeugten Bilder erhöhten. Das Ergebnis für den Schlafzimmer-Datensatz ist in Abb. 9.3 in einer Auflösung von  $256 \times 256$  zu sehen. Hier sind die Geometrie und die Perspektive der Zimmer sehr realistisch. Abb. 9.4 zeigt synthetische Bilder für andere Kategorien aus dem LSUN-Datensatz.

Der CelebA-HQ ist ein Datensatz mit 30.000 hochaufgelösten Bildern berühmter Persönlichkeiten, wobei jedes Bild mit 40 Attributen annotiert ist (z. B. mit Brille, lächelnd, mit Hut, ...). Die Bilder in diesem Datenbestand zeigen Personen in sehr unterschiedlichen Posen und mit sehr verschiedenen Hintergründen. Das tiefe neuronale Netz wurde auf diesen Daten unter Verwendung von acht Tesla V100 GPUs für vier Tage trainiert.

Abb. 9.5 und Abb. 2.24 zeigt mithilfe des Progressive GAN (Karras et al. 2018) erzeugte Bilder in der Auflösung  $1024 \times 1024$ . Die Bilder haben eine hohe Detailtreue und sind sehr realistisch. Ein Video zeigt, wie die Bilder bei allmählich ansteigender Auflösung generiert wurden. Die Autoren sind trotzdem der Ansicht, dass der Weg zum Fotorealismus der Bilder noch weit, aber dennoch erreichbar ist.

Brock et al. (2019) trainieren Conditional GANs, bei denen der Generator durch die Eingabe eines Klassennamens angewiesen wird, Bilder dieser Klasse zu generieren. Durch eine Reihe von Methodenänderungen sind sie in der Lage, GANs mit sehr vielen Para-



**Abb. 9.3** Mit Progressive GANs generierte Bilder von Schlafzimmern, die eine ziemlich realistische Darstellung erzeugen. Bildnachweis im Anhang (Karras et al. 2018)

metern zu trainieren. In Abb. 9.6 werden erzeugte Bilder für unterschiedliche Klassen gezeigt, die einen sehr hohen Detailreichtum und Realismus aufweisen. Die Qualität fällt in einigen Bildklassen ab, für die weniger Trainingsbeispiele zur Verfügung stehen, oder die sehr divers sind. Die Autoren zeigen, dass die erzeugten Bilder neue Blickwinkel und Individuen zeigen und nicht die Modifikation eines Trainingsbildes sind.



**Abb. 9.4** Synthetische  $256 \times 256$  Bilder der Kategorien Bus, Kirche, Fahrrad und TV-Monitor aus dem LSUN-Datensatz. Bildnachweis im Anhang (Karras et al. 2018)



**Abb. 9.5** Diese Personen existieren nicht! Mit dem Progressive GAN erzeugte Bilder in hoher Auflösung. Bildnachweis im Anhang (Karras et al. 2018)



**Abb. 9.6** Erzeugte Bilder der Größe  $512 \times 512$  für unterschiedliche Klassen. Als Trainingsdaten wurde ImageNet verwendet. Bildnachweis im Anhang (Brock et al. 2019)

### 9.1.5 Interpolation zwischen Bildern

Nach dem Training des GANs bildet der Generator einen Vektor  $x$  von Zufallszahlen auf ein Bild ab. Man kann den Vektor als ein Embedding des Bildes betrachten, welcher dessen Eigenschaften charakterisiert. Abb. 9.7 zeigt durch Interpolation entstandene Bilder des Progressive GAN (Karras et al. 2018). Sie verdeutlichen, dass eine Interpolation im Raum der Eingabevektoren zu stetigen Übergängen zwischen den Bildern führt. Jedes Bild stellt ein plausibles Gesicht einer Person dar.

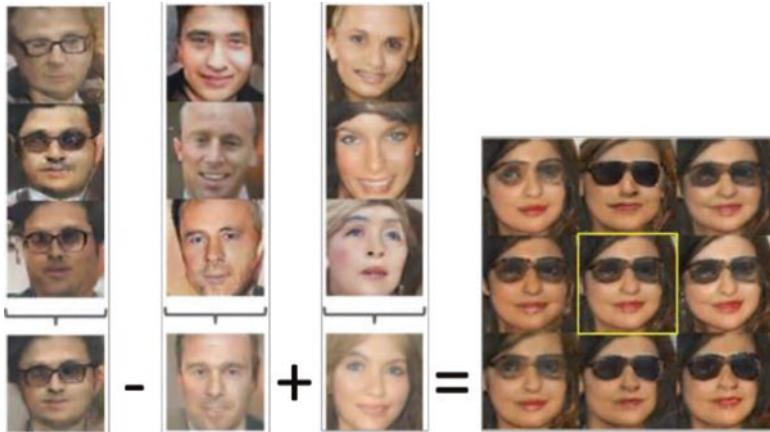
Bei den Embeddings von Wörtern haben wir gesehen, dass die Embedding-Vektoren oft eine lineare Beziehung zueinander haben, z. B.  $x_{\text{König}} - x_{\text{Mann}} + x_{\text{Frau}}$  ergab einen Vektor, dessen nächster Nachbarvektor  $x_{\text{Königin}}$  war (Abschn 6.2.2). Das gleiche kann man auch mit den Vektoren durchführen, die zu den GAN-Bildern gehören. Hier zeigte sich, dass es nicht reicht, einen Vektor einer Kategorie zu verwenden, sondern dass man Prototypen als Durchschnitt aus mehreren Vektoren bilden muss. Wenn man die Prototypvektoren auf folgende Weise verrechnet:  $x_{\text{Mann mit Brille}} - x_{\text{Mann ohne Brille}} + x_{\text{Frau ohne Brille}}$ , so wird aus dem Resultatvektor das gelb umrandete Gesicht erzeugt (Radford et al. 2016). Dies zeigt, dass sich das Konzept Brille mit dem Konzept Frau zusammenführen lässt (Abb. 9.8).

Jedes Teilnetz des GANs kann während des Trainings zu „gut“ werden. Ist der Diskriminator zu gut, so erhält der Generator nicht genügend Gradienteninformation, um seine Bilder zu verbessern. Ist der Generator zu gut, so wird er Schwächen des Diskriminators ausnutzen. Diese Effekte kann man durch Anpassung der Lernraten kontrollieren. Ein weiteres Problem kann sein, dass der Generator korrekte Bilder mit nur sehr wenigen Motiven erzeugt, d. h. zu wenig Variation hat.

Mittlerweile gibt es Hunderte unterschiedlicher GAN-Ansätze (Hindupur 2018). Viele Autoren arbeiteten an der Verbesserung der Verlustfunktionen (z. B. Wasserstein-GAN, LSGAN etc.). Weiter kann man mit Conditional GANs die Art der erzeugten Bilder aus-



**Abb. 9.7** Durch Interpolation der Inputvektoren entstandene Bilder. Jeweils das rechte und das linke Bild einer Zeile wurden interpoliert. Bildnachweis im Anhang (Karras et al. 2018)



**Abb. 9.8** Es werden Vektoren der Prototypen „Mann mit Brille“ (links), „Mann ohne Brille“ (2. v. links) und „Frau ohne Brille“ (Mitte) gebildet, welche zu den Bildern in der unteren Reihe korrespondieren. Wenn man  $x_{\text{Mann mit Brille}} - x_{\text{Mann ohne Brille}} + x_{\text{Frau ohne Brille}}$  berechnet, wird aus dem Resultatvektor das gelb umrandete Gesicht erzeugt. Addiert man hierzu Rauschen im Bereich  $[-0.25, +0.25]$ , so ergeben sich die restlichen Gesichter. Bildnachweis im Anhang (Radford et al. 2016)

wählen. Trotzdem ist das Training von GANs immer noch komplex. Mit sogenannten Variational Autoencodern kann man ebenfalls Bilder erzeugen. Allerdings sind die erzeugten Bilder in der Regel unschärfer.

### 9.1.6 Transformation von Bildern

In vielen Situationen möchte man eine Version eines Bildes in eine andere übersetzen. Abb. 9.9 zeigt einige Beispiele für solche Transformationen: Erzeugung eines Farbbildes aus einem Schwarz-Weiß-Bild, Erzeugung eines Fotos aus einer Skizze oder Generierung einer Landkarte aus einem Luftbild. Isola et al. (2017) schlagen dazu ein auf GANs basierendes Verfahren vor. Betrachten wir das in Abb. 9.10 gezeigte Beispiel: Hier erzeugt der Generator aus einer Skizze ein Bild. Der Diskriminator erhält als Eingabe zufallsgesteuert ein Paar bestehend aus einer Skizze und einem Foto und muss entscheiden, ob dieses Paar echt ist. Die Skizze fungiert als Randbedingung für die Bilderzeugung, daher heißt das Verfahren Conditional GAN.

Der Generator hat eine Struktur ähnlich wie in Abb. 9.13. Zunächst wird die Bildauflösung durch CNNs mit Schrittweite größer als 1 reduziert und die Anzahl der Merkmale erhöht. Die erzeugte Darstellung wird dann mit Residual Blocks wie bei ResNet (Abschn. 5.5.2) transformiert. Durch den Bypass können relevante Merkmale einfach weitergegeben werden und das Training wird erleichtert. Schließlich wird das Bild durch Transposed Convolution-Schichten (Abschn. 5.8.4) wieder auf die volle Auflösung expandiert.

Der Diskriminator konzentriert sich auf die Unterschiede lokaler Muster zwischen Skizze und Bild und wird daher PatchGAN genannt (Isola et al. 2017). Beide Teilnetze



**Abb. 9.9** Beispiele für die Transformation von Bildern. Die Eingaben (obere Reihe) werden in die zugehörigen Ausgabebilder transformiert (untere Reihe). Für jeden Beispieltyp wird ein eigenes Modell trainiert. Bildnachweis im Anhang (Isola et al. 2017)

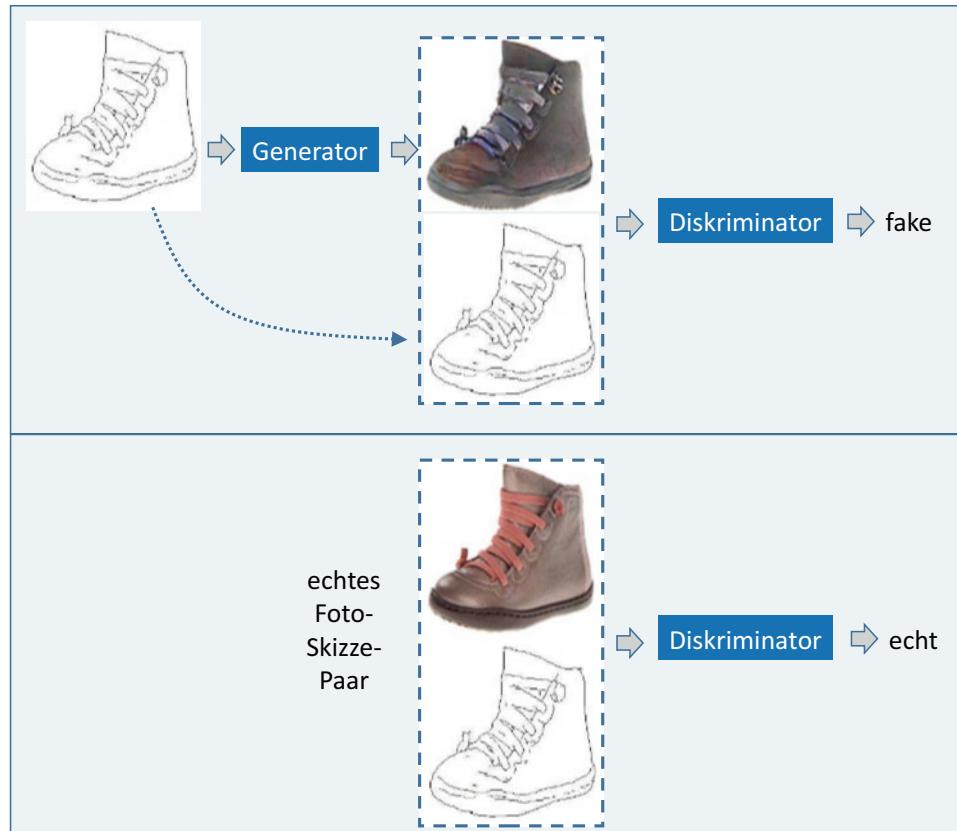
werden wieder abwechselnd trainiert. Die Trainingsdaten bestehen aus den Paaren der beiden Bildvarianten, z. B. Skizze und Foto.

Es wurden Netze für eine Reihe unterschiedlicher Aufgaben trainiert, z. B. Landkarte  $\longleftrightarrow$  Luftfoto (Abb. 9.11), Schwarzweiß-Foto  $\rightarrow$  Farbfoto, Sketch  $\longleftrightarrow$  Foto, Sketch  $\rightarrow$  Portraitbild, 3D-Skizze  $\rightarrow$  Straßenansicht, Foto mit Hintergrund  $\rightarrow$  Foto ohne Hintergrund. Die Ergebnisse waren derart überzeugend, dass menschliche Betrachter häufig die generierten Ergebnisse als reale Szenen wahrnahmen.

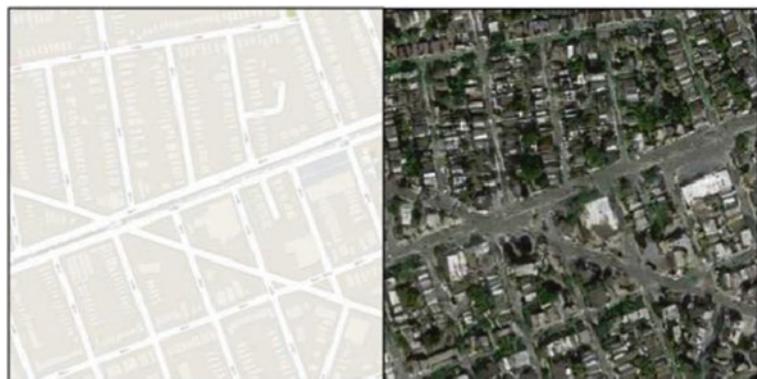
Eine Anwendung für die Bildtransformation ist die Erhöhung der Auflösung von Bildern (Ledig et al. 2017). Hierbei verwendet man Bildpaare mit jeweils niedriger und hoher Auflösung zum Training für eine bestimmte Domäne, z. B. Personenbilder. Eine weitere Möglichkeit ist die Ergänzung von fehlenden Teilbereichen eines Bildes.

### 9.1.7 Transformation von Bildern ohne Trainingspaare

Möchte man das Foto eines Pferdes in das Bild eines Zebras transformieren, so sind hierfür keine Trainingsdaten vorhanden, da Pferde nie die Streifen von Zebras haben. Damit ist die oben beschriebene Methode zur Transformation der Bilder nicht anwendbar. Zhu et al. (2017) haben ein Modell namens CycleGAN beschrieben, welches derartige Trans-



**Abb. 9.10** Der Generator erzeugt aus der Skizze ein Foto des Gegenstandes. Der Diskriminatator erhält ein Skizze-Foto-Paar als Eingabe und muss entscheiden, ob dieses Paar echt ist oder vom Generator erzeugt wurde. Bildnachweis im Anhang (Isola et al. 2017)



**Abb. 9.11** Ergebnis einer Bildtransformation von Landkarte nach Luftbild. Bildnachweis im Anhang (Isola et al. 2017)

formationen erlaubt, wenn man aus den jeweiligen Domänen zwei unabhängige Stichproben von Bildern hat, aber keine Paare mit einem Pferd und dem zugehörigen Zebra.

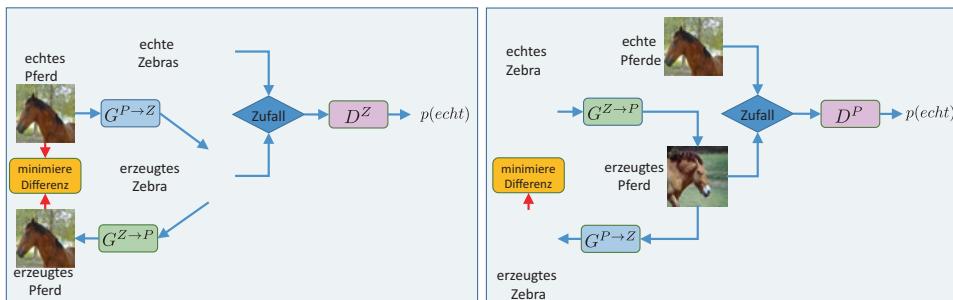
Ausgangspunkt sind zwei Generatorennetze  $G^{P \rightarrow Z}$  und  $G^{Z \rightarrow P}$ , welche aus Pferden Zebras und aus Zebras Pferde erzeugen (Abb. 9.12). Es wird mit zwei Diskriminatoren  $D^Z$  und  $D^P$  wie beim GAN geprüft, ob die erzeugten Bilder tatsächlich Zebras bzw. Pferde sind. Die Generatoren versuchen, möglichst echt aussehende Zebras bzw. Pferde zu erzeugen.

Das reicht aber noch nicht aus um sicherzustellen, dass die aus Pferden erzeugten Zebras tatsächlich dem Ausgangspferd ähnlichsehen. Dazu verwenden die Autoren die Rücktransformation  $G^{Z \rightarrow P}$ , welche aus dem generierten Pferd wieder ein Zebra erzeugt. Es wird der Unterschied zum Ausgangsbild  $p_0$  berechnet und in einem zusätzlichen Kostenterm verlangt, dass der Absolutwert der Differenz  $\|G^{Z \rightarrow P}(G^{P \rightarrow Z}(p_0)) - p_0\|_1$  gering ist. Hiermit wird sichergestellt, dass sich das Ausgangsbild des Pferdes und das erzeugte Bild in ihren Pixeln nur wenig unterscheiden. Analog wird aus einem Zebrabild ein Pferdebild erzeugt, um den inversen Generator zu trainieren.

Die Architektur der Generatoren ist in Abb. 9.13 wiedergegeben. Als Diskriminator wurde wieder PatchGan (Abschn. 9.1.6) gewählt, welches sich auf lokale Unterschiede konzentriert.

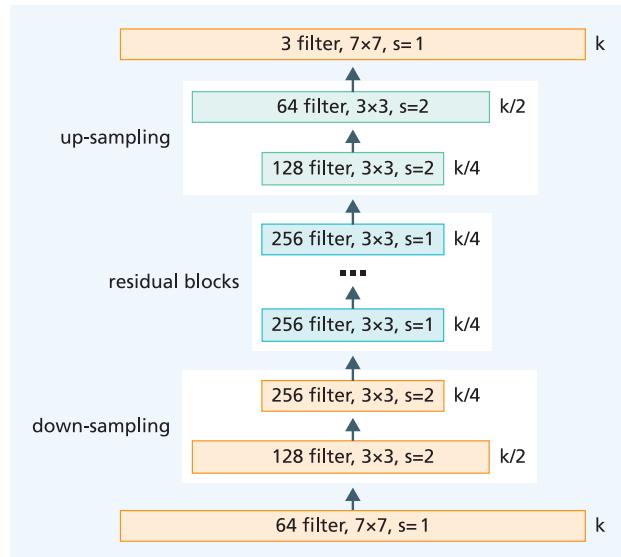
Das Verfahren wurde auf eine Reihe unterschiedlicher Benchmark-Datensätze angewendet und hatte im Vergleich zu alternativen Ansätzen immer eine bessere Genauigkeit (Zhu et al. 2017). Eine Beispielanwendung ist die Transformation von Fotos in Gemälde im Stil von van Gogh (Abb. 9.14 oben). Es wird deutlich, dass die Rücktransformation dafür sorgt, dass die Bilder strukturell gleich sind und sich nur in lokalen Stilmerkmalen unterscheiden. Trotzdem sind größere Farbunterschiede zu beobachten. Im Gegensatz zu anderen Ansätzen lernt das Verfahren die Stilmerkmale einer ganzen Kollektion von Bildern und nicht eines Einzelbildes.

Es gibt mittlerweile eine große Anzahl von Anwendungen des Verfahrens. Beispielhaft sind die Übertragung eines Fotos in eine andere Jahreszeit (Abb. 9.14 Mitte) oder von Tagesslicht zu Dämmerung. Darüber hinaus kann man Verbesserungen eines Fotos berechnen, z. B. die Weichzeichnung des Hintergrundes (Abb. 9.14 unten) oder die Entfernung von Dunst. Eine weitere Anwendung ist die Übertragung von Personenbildern in Comicfiguren.



**Abb. 9.12** Training eines Generators Pferd → Zebra durch ein GAN (links) und eines Generators Zebra → Pferd durch ein anderes GAN (rechts). Zusätzlich wird das Bild durch den jeweils anderen Generator zurücktransformiert und die Differenz zum Ausgangsbild minimiert. Bildnachweis im Anhang (Zhu et al. 2017)

**Abb. 9.13** Architektur des CycleGAN Generators. Das Bild mit der Ausgangsgröße  $k$  wird durch downampling mit Convolution-Schichten und Schrittweite  $s=2$  auf eine Größe von  $k/4$  reduziert. Anschließend wird das Bild durch 6 Residuale Blöcke (S. 93) transformiert. Schließlich wird das Bild durch Transposed Convolution-Schichten (Abschn. 5.8.4) wieder auf die volle Auflösung expandiert. (Wolf 2018)

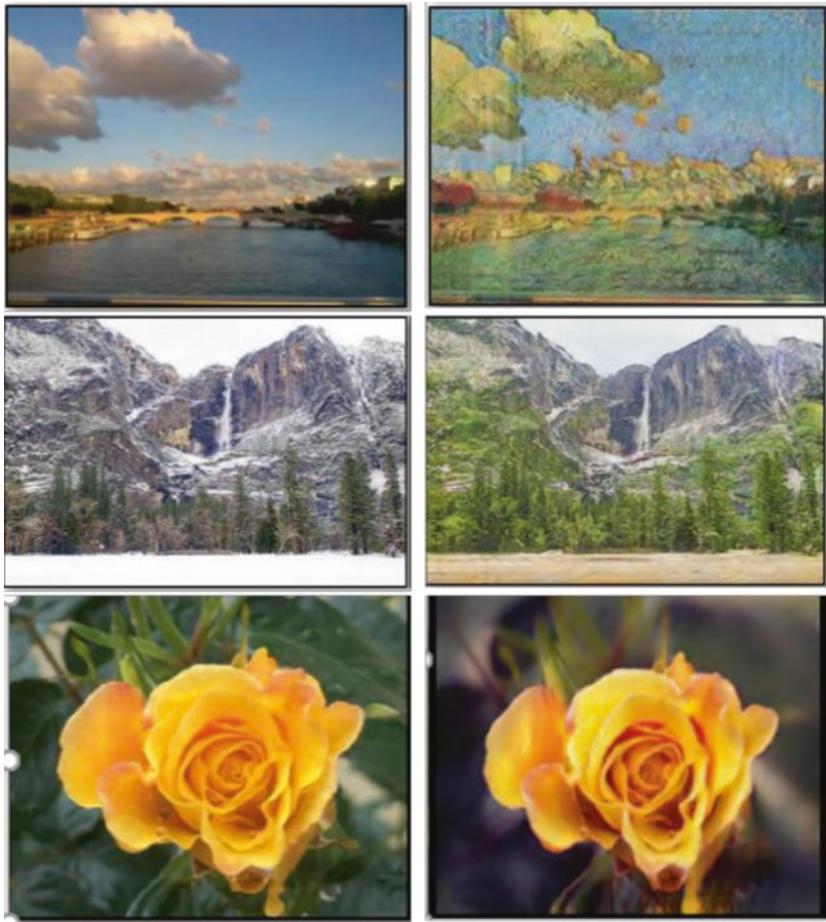


Insgesamt sind die von CycleGAN erzielten Ergebnisse sehr gut – die Bildqualität nähert sich bei vielen Aufgaben dem mit Bildpaaren trainierten Verfahren. Dies ist beeindruckend, da gepaarte Transformationen durch überwachtes Lernen trainiert wurden und CycleGAN diese Informationen nicht zur Verfügung hat. Zum Zeitpunkt der Publikation übertraf CycleGAN andere unüberwachte Bildübersetzungstechniken. In Experimenten konnte der Mensch synthetisierte Bilder in etwa 25 % der Fälle nicht von realen Bildern unterscheiden.

### 9.1.8 Creative Adversarial Network

Die bisher diskutierten Algorithmen haben Bilder aus gewissen inhaltlichen Bereichen erzeugt, z. B. Bilder von van Gogh oder realistische Gesichter von Stars. Es war aber nicht das Ziel, unbedingt etwas Neues zu generieren. Elgammal et al. (2017) schlagen ein innovatives System zur Erzeugung von Kunst vor. Das System lernt, wie Bilder unterschiedlicher Kunstile aussehen. Die Autoren entwickeln ein Creative Adversarial Network (CAN) und schlagen Modifikationen seines Ziels vor, um es in die Lage zu versetzen, kreative Kunst zu erzeugen, indem die Abweichung von etablierten Stilen maximiert und die Abweichung von der „Kunstverteilung“ minimiert wird.

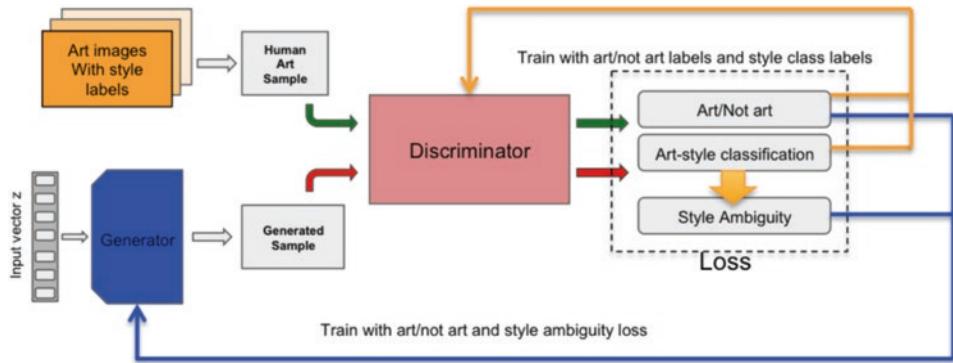
Die Autoren modifizieren ein GAN, indem sie einerseits einen Diskriminator für die Klassifikation von Kunststilen einführen (Abb. 9.15). Da aber etwas Neues erzeugt werden soll, wird die eindeutige Zugehörigkeit zu einem einzelnen Kunststil bestraft. Es sollen Bilder produziert werden, die zu mehreren Kunststilen gehören könnten. Zusätzlich gibt es den üblichen Diskriminatator, der zwischen echten Künstlerbildern und generierten Bildern unterscheidet. Der Generator wird nun so trainiert, dass er einerseits „echte Kunstwerke“ und andererseits Bilder erzeugt, die mehreren Kunststilen angehören könnten.



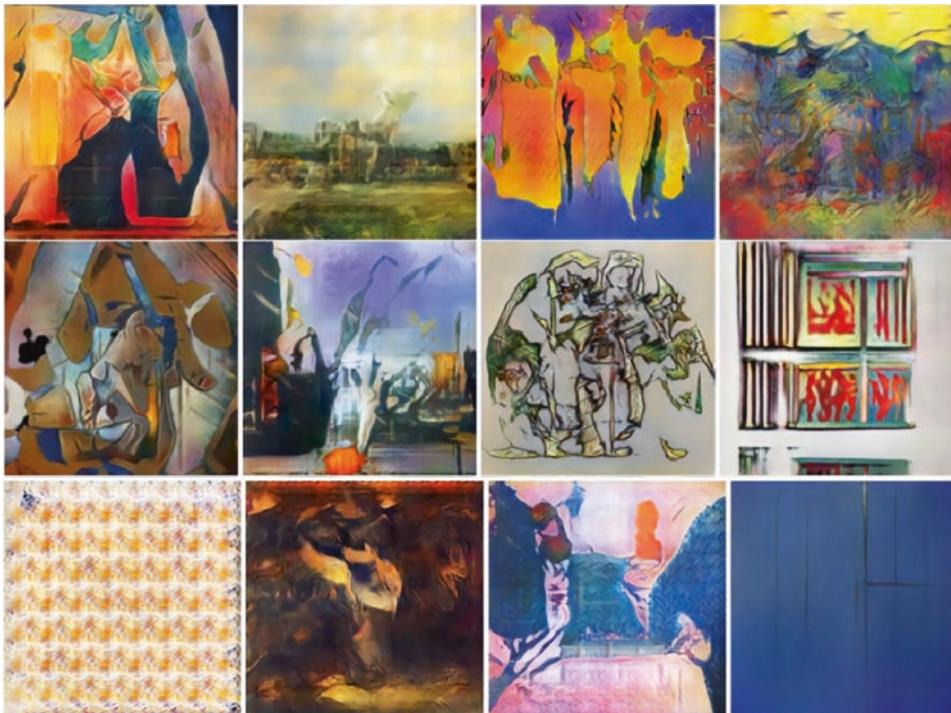
**Abb. 9.14** Mit unterschiedlichen CycleGAN-Modellen berechnete Transformationen von Fotos (links). Foto → Van Gogh (oben), Yosemite-Park im Winter → Yosemite Park im Sommer (Mitte). Verbesserung eines Smartphone-Fotos durch Weichzeichnung des Hintergrundes (unten). Bildnachweis im Anhang (Zhu et al. 2017)

Die Autoren trainierten ihr Modell mit dem frei verfügbaren WikiArt-Datenbestand. Dieser enthält über 80.000 Bilder von mehr als 1100 Künstlern vom fünfzehnten bis zum zwanzigsten Jahrhundert. Die Bilder sind in 25 Stilarten kategorisiert. Abb. 9.16 zeigt eine Reihe von CAN-erzeugten Bildern, welche von menschlichen Betrachtern nachträglich bewertet wurden.

Die Autoren haben anschließend Experimente durchgeführt, um die Reaktion menschlicher Betrachter auf die generierte Kunst mit der Reaktion auf Kunst zu vergleichen, die von Künstlern geschaffen wurde. Die Ergebnisse zeigen, dass menschliche Betrachter Kunst, die von CAN erzeugt wurde, nicht von Kunst unterscheiden konnten, die von zeitgenössischen Künstlern erzeugt und auf Top-Kunstmessen, z. B. der Art Basel 2016, gezeigt wurde.



**Abb. 9.15** Das Creative Adversarial Network trainiert zwei Diskriminatoren: der erste unterscheidet generierte Bilder von echten Künstlerbildern. Der zweite klassifiziert die Stilrichtung eines Kunstwerkes. Ziel ist die Erzeugung neuer Kunstwerke, die mehreren Stilen angehören können und damit eine Neuigkeitswert haben. (Elgammal et al. 2017)



**Abb. 9.16** Von CAN erzeugte Beispielbilder. Die obersten beiden Reihen gehören zu den am höchsten von Menschen bewerteten Bildern, die untere Reihe zu den am niedrigsten bewerteten Bildern. Bildnachweis im Anhang (Elgammal et al. 2017)

### 9.1.9 Erzeugung von Bildern aus Text

Schließlich kann man auch Bilder aus Textbeschreibungen erzeugen. Zhang et al. (2017) nutzen zunächst ein tiefes rekurrentes Netz (ein vortrainiertes LSTM, Abschn. 6.4), um einen Embeddingvektor für den Beschreibungstext zu erzeugen. Konkret verwenden sie den Mittelwert aller verdeckten Vektoren der letzten Schicht des RNN als Embeddingvektor. Sie verwenden nun zwei GANs, um aus dem Embeddingvektor ein Bild zu erzeugen. Dabei werden die Textembeddings durch einen Zufallsprozess angereichert, um die nötige Variabilität der Eingaben zu erreichen.

Das erste GAN generiert ein Bild mit einer niedrigen Auflösung von  $64 \times 64$ , welches die wichtigsten Farben und Formen beinhaltet. Hieraus wird mit einem zweiten GAN ein höher aufgelöstes Bild mit der Auflösung von  $256 \times 256$  produziert. Die Architektur von Generatoren und Diskriminatoren folgt dem oben beschriebenen Muster. Dadurch, dass zwei Diskriminatoren eingesetzt werden, die die erzeugten Bilder mit echten  $64 \times 64$  Bildern und  $256 \times 256$  Bildern der Domäne vergleichen, enthalten die endgültig erzeugten Bilder fotorealistische Details. Das Verfahren wird StackGAN genannt.

Das StackGAN wird auf drei verschiedenen Trainingsdaten trainiert, welche zu jedem Bild eine textuelle Beschreibung enthalten. Abb. 9.17 zeigt, dass die erzeugten Bilder in sich konsistent erscheinen und der Beschreibung entsprechen. In einer Bewertung durch menschliche Experten zeigte sich, dass das Verfahren bessere Resultate lieferte als die verfügbaren Alternativen. Das StackGAN kombiniert Aspekte aus den Trainingsbildern, ohne sie als Ganzes zu verwenden. Durch Vergleich mit den Trainingsdaten ließ sich zeigen, dass die erzeugten Bilder visuelle Merkmale mit den Trainingsbildern teilten, aber anders zusammengesetzt waren (Zhang et al. 2017). Wie bei den normalen GANs ließen sich durch Interpolation der Textembedding-Vektoren auch fließende Übergänge zwischen den Eigenschaften der erzeugten Bilder beobachten (s. Abb. 9.18).

<b>Beschreibungs-</b> <b>text</b>	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
<b>Bilder der</b> <b>Stufe I</b> <b><math>64 \times 64</math></b>			
<b>Bilder der</b> <b>Stufe II</b> <b><math>256 \times 256</math></b>			

**Abb. 9.17** Mit StackGAN erzeugte Bilder aus nicht zuvor gesehenum Text (oben). In der mittleren Zeile sind die Ergebnisse der Stufe I und in der unteren Zeile die Ergebnisse der Stufe II zu sehen. Bildnachweis im Anhang (Zhang et al. 2017)



**Abb. 9.18** Interpolation zwischen Beschreibungstexten. In der oberen Zeile Interpolation zwischen „The bird is completely red“ nach „The bird is completely yellow“. In der zweiten Zeile: „This bird is completely red with black wings and pointy beak“ nach „This small blue bird has a short pointy beak and brown on its wings“. Bildnachweis im Anhang (Zhang et al. 2017)

### 9.1.10 GAN-generierte Modelle von Personen in drei Dimensionen

Datagrid, ein japanisches Technologieunternehmen, hat ein GAN-Modell entwickelt, welches 3d-Ganzkörpermodelle von Personen erstellen kann (Vardaan 2019). Als Eingaben werden dabei 2d-Bilder und Videos von Personen benutzt und Ganzkörpermodellbilder mit hoher Auflösung (1024 x 1024) und hoher Qualität erzeugt.

Dieses GAN kann Fotos erzeugen, die für menschliche Beobachter zumindest oberflächlich authentisch wirken und viele realistische Eigenschaften aufweisen: Gesichter, Haare, Kleidung usw. Diese synthetisch generierten Menschen existieren in der Realität nicht. Also können Modemarken oder Werbeagenturen diese als Mode-Modells verwenden, ohne dass Kosten für Fotos oder Gagen anfallen. Auf (Vardaan 2019) kann man sich Videos ansehen (Abb. 9.19), die die Modelle und ihre Bewegungen zeigen. Sie wirken



**Abb. 9.19** Einzelbilder aus einem Video von DataGrid, welches die Bewegung von drei 3D-Personenmodellen zeigt. Den Personen können dynamisch neue Kleidungsstücke „angezogen“ werden und sie können durch Interpolation dynamisch ineinander überführt werden. Bildnachweis im Anhang (Vardaan 2019)

derzeit noch etwas ungelenk, können aber in Zukunft sicher noch verbessert werden. Gleichzeitig können den 3D-Körpern neue Kleidungsstücke „angezogen“ werden und die 3D-Personen können durch Interpolation ineinander überführt werden.

## 9.2 Texte verfassen

### 9.2.1 Automatischer Reporter: Daten in Zeitungsmeldungen darstellen

Es gibt mittlerweile eine Reihe von KI-Programmen, welche automatisch Zeitungsartikel schreiben. Bloomberg News produziert ein Drittel seiner Inhalte schon mithilfe des automatisierten Verfahrens „Cyborg“ (Peiser 2019). Das Programm erhält z. B. Bilanzdaten eines Unternehmens, analysiert diese und produziert eine Meldung mit den relevantesten Fakten und Kennzahlen. Da die Leser diese Meldungen für Kaufentscheidungen nutzen, ist die Publikationsgeschwindigkeit entscheidend. Bei Associated Press erzeugt das Programm „Wordsmith“ (Miroshnichenko 2018) anhand von Regeln Meldungen zu Quartalsgewinnen von Unternehmen (Abb. 9.20).

Neue Ansätze betrachten die Produktion einer Textmeldung aus textuellen Stichworten als Übersetzungsproblem. Beispielsweise verwenden Gehrman et al. (2018) ein transformer-basiertes Sequence-to-Sequence Modell (Abschn. 6.6) (Abb. 9.21).

### 9.2.2 Generierung von längeren Geschichten

Möchte man mit neuronalen Sprachmodellen längere Geschichten erzeugen, so haben derzeit viele Modelle die Tendenz, nach kurzer Zeit den roten Faden zu verlieren. Daher ist es notwendig, mit anderen Mitteln den konsistenten Verlauf einer Geschichte sicherzustellen. Fan et al. (2018) nutzen hierzu ein zweistufiges Verfahren. Sie charakterisieren auf der oberen Schicht den roten Faden einer Geschichte durch „Ziele“, welche das Thema eines

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, ...

**Abb. 9.20** Beispiel eines maschinell produzierten Artikels der Associated Press. (Miroshnichenko 2018)

Ziel: 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. [...]

**Abb. 9.21** Beispiel für Ziele und der Anfang der zugehörigen Geschichte aus dem Datensatz von Fan et al. (2018)

Abschnitts der Geschichte beschreiben. Auf der unteren Schicht verwendet ein Sequence-to-Sequence Modell diese Ziele als Eingabe, um die Geschichte mithilfe eines weiteren Sequence-to-Sequence-Modells zu formulieren.

Als Trainingsdaten sammelten Fan et al. (2018) einen Datensatz mit etwa 300.000 Geschichten mit den zugehörigen Zielen aus dem Reddit-Forum. Die Geschichten umfassten im Durchschnitt 734 Worte und im Mittel 28 Zielworte.

Die Autoren nutzen ein Convolutional Sequence-to-Sequence-Modell ähnlich dem Transformer (Abschn. 6.6). Hierbei verwenden sie auch mehrere Attention-Mechanismen pro Schicht (Multi-Head Attention) wobei sie die linearen Transformationen noch mit einem Gatter versehen, wie dies auch im Long Short-Term Memory (Abschn. 6.4) verwendet wurde:  $(A * x + b) * \text{sigmoid}(C * x + d)$ . Ähnlich wie bei WaveNet (Abschn. 7.5.1) wird ein Attention-Mechanismus verwendet, der immer weiter entfernt liegende Elemente betrachten kann. Hierdurch können lange zurückliegende Inhalte bei der Erzeugung eines neuen Wortes berücksichtigt werden.

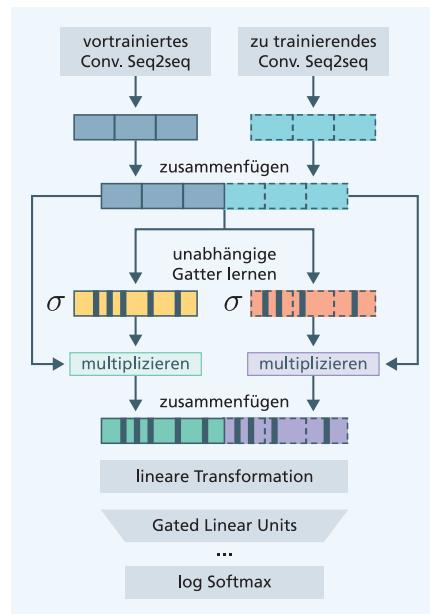
Gibt man einem Sequence-to-Sequence-Modell Ziele als zusätzlichen Input ein, so werden diese meist ignoriert und das Modell konzentriert sich auf die Reproduktion des Textes. Aus diesem Grunde gehen Fan et al. (2018) anders vor. Sie trainieren zunächst ein Sequence-to-Sequence-Modell mit drei Encoder-Schichten und acht Decoder-Schichten, welches sich auf die Generierung des Textes konzentriert. Danach trainieren sie ein weiteres Sequence-to-Sequence-Modell, welches Zugang zu den verdeckten Vektoren des ersten Modells hat, und über einen Fusionsmechanismus die Fehler des ersten Modells korrigieren und die Ziele stärker berücksichtigen kann (Abb. 9.22). Das zweite Modell hat fünf Encoder-Schichten und fünf Decoder-Schichten.

Die Resultate (Abb. 9.23) zeigen, dass diese Modellarchitektur zu verbesserten Bewertungen führt. Bei der Beurteilung durch menschliche Leser wurden die hierarchisch erzeugten Geschichten in 2/3 der Fälle präferiert. Durch die Berücksichtigung weit entfernter Worte durch den Attention-Mechanismus wurde die Genauigkeit des Sprachmodells (gemessen in Perplexität Abschn. 6.4.3) von 47 auf 37 verbessert. Bei der Verwendung des Fusionsmodells war ebenfalls eine leichte Verbesserung möglich.

### 9.2.3 GPT2 erfindet komplexe Geschichten

Am 14. Februar 2019 gab die Non-Profit Forschungsinstitution OpenAI bekannt (Radford et al. 2019), dass sie ein neues, sehr mächtiges Sprachmodell (Abschn. 6.3.1) entwickelt

**Abb. 9.22** Nachträgliche Fusion der Ausgaben beider Sequence-to-Sequence-Modelle. Nur das Fusionsmodell und das rechte Sequence-to-Sequence-Modell werden trainiert. (Fan et al. 2018)



haben. Es wurde auf einem sehr großen Trainingsdatensatz trainiert, um jeweils das nächste Wort eines Textes zu prognostizieren. Sprachmodelle sind in der Lage, die Wahrscheinlichkeit aller Worte eines Trainingskorpus detailliert zu erfassen und können daher potentiell inhaltlich konsistente Texte produzieren (Abschn. 6.3.2).

Das zugrundeliegende Modell heißt GPT-2 (Generative Pretrained Transformer) und hat eine große Ähnlichkeit zum Transformer-Decoder (Abschn. 6.6). Das kleinste Modell

Ziel: 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 ?“

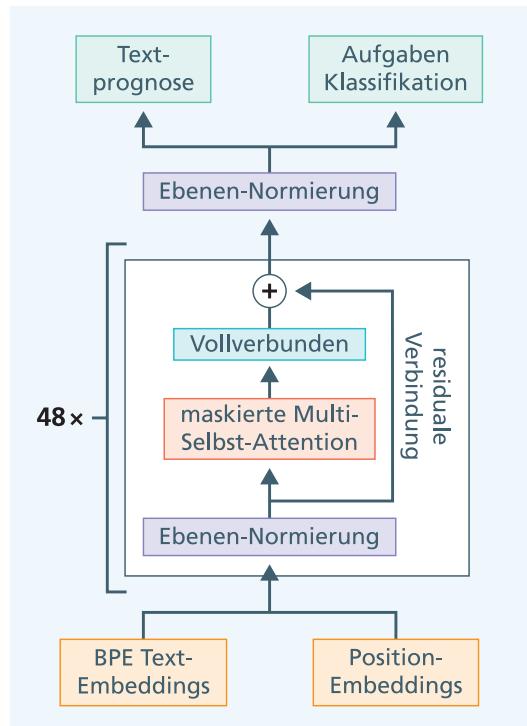
„It ... It 's a monster.“

**Abb. 9.23** Beispiel eines Ziels und einer hieraus erzeugten Geschichte. (Fan et al. 2018)

hat 12 Schichten mit verdeckten Vektoren der Länge 768 und 117 Millionen Parametern (Radford et al. 2019). Das größte Modell hat 48 Schichten mit verdeckten Vektoren der Länge 1600 und insgesamt 1,5 Milliarden Parametern (Abb. 9.24). Die Worte wurden mit Byte-Pair-Encoding (BPE, Abschn. 6.6.5) kodiert, sodass häufige Wörter als Ganzes kodiert werden, seltener Wörter dagegen aus n-Grammen dargestellt werden können. Es wurde dabei ein Vokabular von 50.257 Wörtern und n-Grammen verwendet, für die jeweils ein Embedding gelernt wurde. Zudem wurde ein Embedding für die Position eines Wortes oder n-Gramms trainiert.

Das Modell ist aus einer Reihe von Transformer-Decoder-Blöcken aufgebaut (rechte Seite von Abb. 6.50). Der Transformer-Encoder wird nicht benötigt, und damit auch keine Multi-Kreuzattention. Allein die maskierte Multi-Selbstattention zusammen mit der vollverbundenen Schicht erzeugt die aussagekräftigen Embeddings der eingegebenen Wörter, die dann für die Prognose des nächsten Wortes genutzt werden. Die Schichtennormierung (Ba et al. 2016) sorgt dafür, dass Mittelwert und Varianz der Eingaben für eine Schicht normalisiert werden. Dies hat einen ähnlichen Regularisierungseffekt wie die Batch-Normalisierung (Abschn. 4.6.4). Die residuale Verbindung (Abschn. 5.5.2) erlaubt überhaupt erst die Optimierung dieses Vielschichtenmodells. Die Ausgabe ist die über ein lo-

**Abb. 9.24** Architektur des GPT2-Modells, welche große Ähnlichkeit zu dem Decoder-Teil des Transformers hat.  
(Radford et al. 2019)



gistisches Regressionsmodell prognostizierte Wahrscheinlichkeit des nächsten Wortes oder n-Gramms. Den Programmcode des kleinen GPT-2-Modells findet man in (Code\_GPT-2 2019).

### Prognosefähigkeit von GPT-2

Die Modelle wurden auf dem WebText-Datensatz von acht Millionen Webseiten mit 40 GB Text trainiert. Diese Webseiten waren auf Reddit – einer Social-Media-Plattform – mit mindestens drei „karma“ bewertet, also als qualitativ gut eingeschätzt worden.

In Abb. 9.25 wurde die Qualität der Prognose des nächsten Wortes von GPT-2 Modellen unterschiedlicher Größe auf unterschiedlichen Testdatensätzen wie dem Penn-Treebank Korpus berechnet. Als Maß wurde die Perplexität (Abschn. 6.4.3) verwendet. Dabei ist „SOTA“ der Wert des bisher besten Modells, welches auf den zugehörigen Trainingsdaten trainiert wurde. GPT-2 hingegen wurde nur auf dem WebText-Datensatz trainiert, also nicht wie die SOTA-Verfahren auf den Trainingsdaten der drei Datensätze. Die Ergebnisse zeigen, dass die großen GPT-2 Modelle in jedem Falle bessere Ergebnisse liefern, ohne die speziellen Trainingsdaten nutzen zu können. Der Lambada-Datensatz testet dabei insbesondere die Fähigkeit, weit aus einander liegende Zusammenhänge zu erfassen.

GPT-2 ist in der Lage, die Fortsetzung eines Starttextes mit noch nie da gewesener Qualität zu erzeugen. In Abb. 9.26 wurde ein Starttext über den Diebstahl von Nuklearmaterial eingegeben. Im ersten Versuch wurde der nachfolgende Fortsetzungstext erzeugt. Der Text ist plausibel und in sich logisch. Hin und wieder gibt es aber auch Fehlschläge, z. B. Wiederholungen von Formulierungen oder widersprüchliche Angaben (z. B. Feuer unter Wasser).

Insgesamt benötigt man im Mittel wenige Versuche, um eine gute Vervollständigung zu erhalten. Dabei hängt die Qualität der Vervollständigung auch davon ab, wie häufig der Kontext im Trainingsmaterial vorkommt. Für gut im Trainingsmaterial repräsentierte The-

Modell	LAMBADA	WikiText2	Penn Tree Bank
SOTA	99,80	39,14	46,54
117M	35,13	29,41	65,85
345M	15,60	22,76	47,33
762M	10,87	19,93	40,31
1542M	8,63	18,34	35,76

**Abb. 9.25** Qualität der Prognose des nächsten Wortes für unterschiedliche Testdatensätze, gemessen in Perplexität (S. 136, kleiner ist besser). SOTA ist der Wert des bisher besten Modells. Die anderen Werte sind für GPT-2 Modelle unterschiedlicher Größe, die nicht auf den zugehörigen Trainingsdaten trainiert wurden. (Radford et al. 2019)

**Starttext:** A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

**Fortsetzung:**

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.”

**Abb. 9.26** Die von GPT-2 erzeugte Fortsetzung des Starttextes (oben), erster Versuch. (Radford et al. 2019)

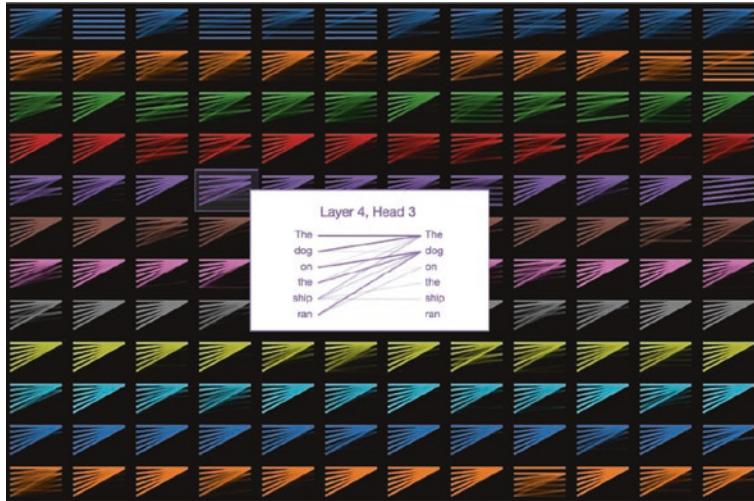
men (z. B. Brexit, Miley Cyrus, Herr der Ringe etc.) werden in der Hälfte der Fälle brauchbare Vervollständigungen erzeugt. Weitere Beispiele findet man bei Radford et al. (2019).

Das System hat ein großes Potenzial, KI-Anwendungen zu verbessern, die Sprachmodelle nutzen. Dies können z. B. Assistenten zum Verfassen von Texten, Dialogagenten, Übersetzungssysteme oder Spracherkennungssysteme sein. Insgesamt zeigt das Modell, dass TNN durchaus in der Lage sind, in kreativer Weise neue Texte zu erzeugen.

Die Autoren weisen aber auch auf das hohe Missbrauchspotenzial hin (Radford et al. 2019), z. B. das Verfassen in sich konsistenter Falschnachrichten, der realistischen Imitation von fremden Personen, der automatischen Produktion von falschen Inhalten in sozialen Medien und der automatischen Erzeugung von personalisierten Spam- und Phishing-Texten. Aus diesem Grunde haben die Autoren den Programmcode des großen GPT-2-Modell zunächst unter Verschluss gehalten und erst Ende 2019 freigegeben (Vincent 2019). Eine Version des Modells kann online ausprobiert werden (King 2019).

### Visualisierung der Modellprognosen

Vig (2019) hat untersucht, wie eine Prognose zustande kommt. Hierzu hat er den kursiv markierten Teil als Starttext eingegeben und mit GPT2 eine Fortsetzung erzeugt: „*The dog on the ship ran off, and the dog was found by the crew*“. Wurde „dog“ durch „motor“ ersetzt, so ergab sich folgender Text: „*The motor on the ship ran at a speed of about 100 miles per hour*“. Das Modell ist also in der Lage, die Mehrdeutigkeit von „ran“ durch die Subjekte „dog“ und „motor“ aufzulösen.



**Abb. 9.27** Die Attention-Muster im kleinen GPT-2-Modell mit 12 Schichten und je 12 unabhängigen Attention-Berechnungen. Die Grafik in der Mitte ist das Muster aus Zeile 4 und Spalte 4. Bildnachweis im Anhang (Vig 2019)

In Abb. 9.27 sind die Attention-Muster abgebildet, die sich in den zwölf Schichten des kleinen GPT2-Netzes zu je zwölf Attention-Berechnungen (Multi-Head-Attention) ergeben haben. Die Intensität der Linien symbolisiert die Stärke des Zusammenhangs. Insgesamt wurde die Attention-Stärken so optimiert, dass die Prognose der nachfolgenden Worte möglichst präzise ist. In der Mitte sieht man ein Muster welches „ran“ sehr stark mit „dog“ in Verbindung bringt. Dies macht Sinn, denn bei der Prognose des nächsten Wortes nach „ran“ ist es wichtig, wer oder was läuft. Es gibt viele weitere interessante Attention-Verbindungen. Dabei scheint das Muster, welches alle Worte auf das erste Wort fokussiert, eine Art Standard zu sein.

### Weiterentwicklungen von GPT-2

Mittlerweile gibt es einige alternative Ansätze zu GPT-2, welche mit signifikant weniger Trainingskosten und kleinerem Modellumfang zu ähnlich guten Ergebnissen kommen (Wang et al. 2019).

Dathathri et al. (2020) stellen ein transformerbasiertes Plug and Play Language Model (PPLM) vor, welches die Steuerung der Texterzeugung erlaubt. Mit Hilfe einfacher Klassifikationsalgorithmen kann die Art des Textes – z. B. Märchen, Wissenschaft, positiv, negativ, etc. – kontrolliert werden. Zum anderen können einzelne Worte des Textes vorgegeben werden, zwischen denen dann konsistent die restlichen Worte des Textes generiert werden. Dabei werden Informationen über die Worte von vorne nach hinten und umgekehrt propagiert. Abb. 9.28 zeigt Beispiele für einen solchen kontrolliert erzeugten Text, wobei die fett gedruckten Worte vorgegeben wurden. Der oberste Absatz zeigt einen Text

**[I]** 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 wake up in a cold sweat, and she was found crying. But, one day, a man walked up to 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.

**Abb. 9.28** Kontrolliert erzeugter Text ohne Typvorgabe (oben), mit Typ „Fantasy“ (Mitte) und Typ „Negative“ (unten). Die fett gedruckten Wörter wurden vorgegeben. (Dathathri et al. 2020)

ohne Typvorgabe. Beim nächsten Absatz wurde der Typ „Fantasy“ und beim letzten Absatz der Typ „Negative“ verlangt. Insgesamt erlaubt dieses Vorgehen potenziell die kontrollierte Generierung eines Textes entlang eines „Storyboards“.

Microsoft hat kürzlich die Leistungsfähigkeit von Sprachmodellen noch einmal gesteigert. Seine Turing Natural Language Generation (T-NLG) ist ein gigantisches Sprachmodell mit 17 Milliarden Parametern und der Architektur eines Transformers (Rosset 2020). Es hat 78 Transformer-Schichten mit verborgene Vektoren der Länge 4256 und 28 parallele Attention-Berechnungen pro Schicht. Um ein derart umfangreiches Modell auf GPUs berechnen zu können, wurden neue Hardware- und Softwareverbesserungen eingesetzt. Das Modell wurde analog zum Megatron (Shoeybi et al. 2019) trainiert, welches eine Trainingsdatensatz von 174 GB verwendete.

Wie GPT-2 kann T-NLG auch die Fortsetzung eines Satzes erzeugen. Die Qualität dieser Fortsetzungen ist noch einmal entscheidend besser als bei GPT-2: die Perplexität auf WikiText (Abb. 9.25) konnte von 17,5 auf 10,2 verbessert werden. Darüber hinaus kann es auch direkt auf Fragen in einem korrekten Satz antworten, in nicht nur – wie bei SQuAD gefordert – die Antwortphrase in einem Dokument identifizieren (Rosset 2020). Beispielsweise erzeugt es ohne Feintuning aus dem Startsatz „How many people live in the US?“

die Fortsetzung „There are over 300 million people living in the US.“ Leider wurde das Modell nur einem kleinen Kreis von Forschern zum Testen zugänglich gemacht.

Im Juli 2020 wurde GPT3 vorgestellt, eine stark vergrößerte Version von GPT2 mit 175 Milliarden Parametern (Brown et al. 2020). Als Trainingsdaten wurden Texte mit etwa 500 Milliarden Worten aus gecrawlten Webseiten, Büchern und der Wikipedia verwendet. Das Modell kann unterschiedliche Textarten (Kurzgeschichten, Lieder, Pressenachrichten, Blogs, usw.) generieren, welche Leser oft nicht von Texten menschlicher Autoren unterscheiden können. Darüber hinaus kann es viele Aufgaben erledigen, z. B. Fragen beantworten, Übersetzungen erstellen, logische Schlussfolgerungen ziehen oder mit kleinen Zahlen rechnen. Dies geschieht indem man dem Modell zuvor einige Beispiele präsentierte (few-shot learning), ohne das das Modell die entsprechende Aufgabe durch Training (finetuning) angepasst wurde. In vielen Fällen ist es auch möglich, die Antworten ohne zuvor präsentierte Beispiele zu erhalten (zero-shot learning). Der Kreis der möglichen Anwendungen ist derzeit noch nicht abzusehen. Da das Training von GPT3 aktuell etwa 4,6 Millionen Dollar kostet, kann es nur von großen Institutionen oder Firmen betrieben werden. Microsoft hat für 1 Milliarde Dollar eine exklusive Lizenz erworben, um GPT3 in seinen Produkten zu nutzen.

---

### 9.3 Musik automatisch komponieren

Die automatische Komposition neuer Musik ist ein alter Traum der Musiker. Schon Mozart publizierte ein „Musikalisches Würfelspiel“, die „Anleitung zum Componieren von Walzern mittels zweier Würfel“ (von Köchel 1862, S. 294d/516 f), welches zum Bestseller wurde. Ein Überblick über bisherige Ansätze zur algorithmischen Komposition liefern (Fernández und Vico 2013).

Die Komposition von Musik erfordert eine Reihe von Schritten, wie etwa die Definition der Melodie, des Rhythmus und die Harmonisierung. Weiter kann man noch zusätzliche Melodiestränge und einen Gesang hinzufügen sowie das Arrangement und die Orchestrierung abstimmen. All diese Schritte lassen sich teilweise oder ganz automatisieren. Häufig geschieht das in Zusammenarbeit mit einem menschlichen Komponisten im Rahmen der computerunterstützten Komposition.

Es gibt schon eine Reihe von Systemen zur Komposition von Musik, die meist Hintergrundmusik oder Musik für YouTube-Videos oder Firmenwebseiten erzeugt haben (Dredge 2019). Dies ist mittlerweile ein Geschäftsmodell, denn bei synthetisch erzeugter Musik sind die Urheberrechte wesentlich günstiger.

Ein Beispiel für einen computererzeugten erfolgreichen Popsong war „Daddy’s Car“ von Sony im Jahr 2018. Er wurde von dem Programm „Flow Machines“ im Stil der Beatles komponiert. Allerdings hat der französische Komponist Benoît Carré das Stück arrangiert und den Text geschrieben (Dredge 2019).

### 9.3.1 MuseNet komponiert Mischungen von Klassik und Pop

Als Beispiel für eine vollautomatische Komposition soll das Programm „MuseNet“ (Payne 2019) dienen, welches Kompositionen mit zehn unterschiedlichen Instrumenten erstellen kann. Es kann Musik in unterschiedlichen Stilrichtungen generieren, von Mozart zu den Beatles, von Rachmaninoff zu Lady Gaga. Die Informationen über die Musik wurden nicht in MuseNet programmiert. Stattdessen lernt das System selbsttätig die Schemata von Harmonie, Rhythmus und Musikstil aus existierenden Musikstücken. Grundlage hierzu waren MIDI-Dateien, welche Noten-, Zeit-, und sonstige Steuerinformationen für elektronische Instrumente enthalten (Abb. 9.29). MuseNet wurde darauf trainiert, die nächste Note in solchen MIDI-Dateien vorherzusagen.

Das MuseNet-Modell wird mithilfe des Sparse Transformers formuliert, einer Variante des GPT-2 Algorithmus (Abschn. 9.2.3), welcher besonders weitreichende Zusammenhänge berücksichtigen kann (Child et al. 2019). Das Verfahren erreicht dies, indem es sehr große Attention-Matrizen als ein Produkt kleinerer Matrizen darstellt, welche wesentlich weniger Parameter besitzen.

MuseNet nutzt ein Sparse Transformer-Netz von 72 Schichten mit 24 parallelen Attention-Berechnungen pro Schicht. Es berechnet die Attention-Muster über einen Kontext von 4096 MIDI-Zeichen. Damit kann es die langfristige Struktur von Musikstücken gut erfassen. Neben dem Embedding der Tonangaben und der Position wurden die relative Lage einer Note in einem Akkord, die Position der Note im ganzen Stück sowie ihre Entfernung zum Ende durch Embeddings dargestellt.

Auf der Webseite (Payne 2019) kann man selbst Musik komponieren lassen. Für MuseNet sind die Stilrichtungen unterschiedlicher Komponisten verfügbar (Abb. 9.30). Um neue Kompositionen zu erzeugen, kann man einen Komponisten auswählen und die Startnoten eines bekannten Stücks verwenden (Abb. 9.31). Anschließend kann man bis zu zehn unterschiedliche Instrumente auswählen, und dann generiert das System ein Musikstück mit der geforderten Charakteristik. In einem fortgeschrittenen Interface kann man z. B. die Startnoten von Lady Gagas „Pokerface“ wählen und dann die Akkordfolgen und Instrumente so abändern, dass der Stil des Songs eher der Musik von Mozart, den Beatles oder Journey ähnelt.

```
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
```

**Abb. 9.29** Die von MuseNet verwendete MIDI-Notation. Für jeden Ton wird das Instrument, die Lautstärke und die Tonhöhe angegeben. (Payne 2019)

**Abb. 9.30** Automatisch gruppierte Darstellung der in MuseNet verfügbaren Komponisten (Ausschnitt). (Payne 2019)



**Abb. 9.31** Mit dieser Bedienungsoberfläche kann man eine neue Komposition in MuseNet anfordern. Bildnachweis im Anhang (Payne 2019)



Musik ist ein anspruchsvolles Anwendungsgebiet zur Generierung neuer Inhalte, weil es in Musikstücken sehr starke Zusammenhänge zwischen weit auseinanderliegenden Teilen gibt. Sehr häufig wird ein Anfangsthema während des Ablaufs variiert und am Schluss wiederholt. Insbesondere muss der Rhythmus beibehalten werden, und die Tonarten müssen in ihrer Abfolge zusammenpassen.

Wegen des langen Zeitraums, den die Attention-Mechanismen erfassen, ist das MuseNet in der Lage, auch zum Ende eines Stücks eine Variation des Anfangsthemas zu wiederholen. Dies wird in (Payne 2019) an einem Stück mit Startnoten von Chopin demonstriert.

Die Bewertungen durch Kritiker sind meist gut (Abb. 9.32). Ein weiterer Kritiker schreibt: „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). Der Programmcode von MuseNet soll demnächst frei verfügbar gemacht werden.

### 9.3.2 Der Music Transformer erfindet Klavierstücke

Ein weiteres System zur Erzeugung ist der von Huang et al. (2019) vorgestellte Music Transformer. Die Autoren verwenden einen „Relative Attention“ Mechanismus, welcher nicht die absolute Position von Noten in einem Musikstück modelliert, sondern ihre relative Distanz. Hierzu wurde ein neuer Algorithmus mit geringerem Speicherbedarf entwickelt (Huang et al. 2018). In Abb. 9.33 sind die Attention-Gewichte der letzten Schicht für einen bestimmten Zeitpunkt dargestellt. Das generierte Stück enthält ein schnelles Tremolo, welches mehr-

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.

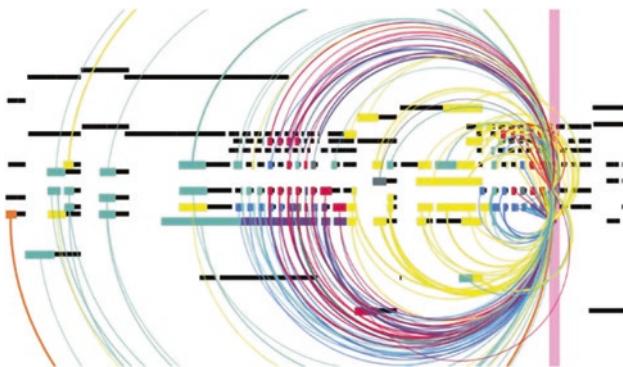
Mozart: I would say Mozart's distinguishing characteristic as a composer is that every measure "sounds right". Even without knowing the piece, you can usually tell when a performer has made a mistake and deviated from the score. The mozart samples sound... wrong. There are parallel 5ths everywhere.

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.

**Abb. 9.32** Bewertung von MuseNet durch einen Pianisten. (Brilee 2019)

**Abb. 9.33** Attention-Verbindungen während der Erzeugung eines neuen Tones mit Music Transformer. Bildnachweis im Anhang (Huang et al. 2019)



fach wiederholt wird und so eine Spannung aufbaut. Die Abbildung zeigt, dass sich das Stück auf die Tremolo-Teile konzentriert und die dazwischenliegenden Passagen auslässt.

Insgesamt kann der Music Transformer einen konsistenten Stil verfolgen und ein Motiv mehrfach variieren und wiederholen. Auf (Huang et al. 2019) finden sich eine Reihe von mit Music Transformer generierten Stücken. Die Qualität der Kompositionen wird von einer Reihe von Hörern als gut beurteilt: „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 Emotionen und Persönlichkeit

Wenn wir uns mit Freundinnen und Freunden unterhalten, dann schätzen wir besonders diejenigen, die wir schon lange kennen. Wir teilen gemeinsame Erlebnisse, kennen die unterschiedlichen Vorlieben und freuen uns auf die Reaktionen des Gegenübers.

Einen Freund zu gewinnen ist eine langwierige Angelegenheit. Zunächst trifft man sich zufällig auf einer Party oder am Arbeitsplatz, redet übers Wetter und macht Smalltalk. Allmählich lernt man seinen Gesprächspartner kennen und teilt möglicherweise Interessen, z. B. einen Lieblingsschauspieler oder den Fußballclub. Langsam öffnet man sich, erzählt Einzelheiten aus seinem Leben oder spricht über Probleme. Spürt man Interesse und Empathie, so verstärkt sich die Bindung. Dazu ist es auch nötig, dass der Andere Einzelheiten von sich preisgibt.

### 9.4.1 Ein XiaoIce Dialog

In Abb. 9.34 findet sich das Protokoll eines Dialogs zwischen einer Person und dem Sprachassistenten XiaoIce, der nach diesem Muster abläuft. XiaoIce („kleines Eis“ auf

**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"  
...

**Abb. 9.34** Eine Folge von Dialogen zwischen einem Nutzer und dem Sprachassistenten XiaoIce. Mit dem Ablauf wird der Nutzer immer vertrauter und teilt mit dem Chatbot schließlich sehr private Details. (Zhou et al. 2018)

Chinesisch) wurde als „empathischer Sprachassistent“ konzipiert (Zhou et al. 2018) und 2014 von Microsoft in China ins Netz gestellt. Mittlerweile ist er mit 660 Millionen Nutzern in China, Japan, USA, Indien und Indonesien der populärste Chatbot der Welt. Er ist auf 40 verschiedenen sozialen Netzen verfügbar, u. a. bei WeChat, QQ, Weibo, Facebook Messenger und LINE. Das wichtigste Erfolgskriterium sind die durchschnittlich 23 Antworten pro Dialog (Conversation-turns Per Session, CPS), die in den Gesprächen zwi-

schen XiaoIce und seinen Nutzern zu beobachten sind. Dies sind mehr Interaktionen als bei Gesprächen zwischen echten Personen (etwa 9). Das zeigt, dass sich die Nutzer gerne und ausführlich mit XiaoIce unterhalten. Mehr noch: die Nutzer bauen eine persönliche Beziehung zu XiaoIce auf. Man könnte XiaoIce als die aktuelle Version von Weizenbaums ELIZA (Abb. 7.23) betrachten.

#### 9.4.2 Das Ziel: Personen zum Weiterreden animieren

Es ist schwierig, Chatbots zu erstellen, mit denen die Leute gerne reden möchten. Alan Turing sah eine glaubhafte Unterhaltung zwischen Mensch und Computer als den ultimativen Test auf Künstliche Intelligenz an (Abb. 9.39).

XiaoIce ist ein TNN, welches die Dialogeingaben in Tensoren von Zahlen umwandelt. Der Chatbot wird auf riesige Datenmengen aus früheren Gesprächen trainiert und lernt, „gute“ Antworten mit einer bestimmten Eingabe statistisch zu verknüpfen. Was aber ist eine „gute“ Antwort? Hier kommt die Anzahl der Antworten pro Dialog ins Spiel. Ein Teil des XiaoIce-Modells sagt voraus, wie wahrscheinlich es ist, dass der Dialog weitergeführt wird. Dies geht über die reine Suche nach sinnvollen Antworten hinaus. Allerdings hängen diese Ziele zusammen, da es unwahrscheinlich ist, dass man stundenlang mit einem Chatbot spricht, der immer nur Unsinn redet. „Ich weiß nicht“ ist eine zutreffende Antwort auf viele Fragen, sorgt jedoch für ein langweiliges Gespräch. In jeder Gesprächsrunde versucht XiaoIce daher, den Nutzer am Reden zu halten (Abb. 9.35).

Dieser Ansatz erklärt zum Teil, warum XiaoIce erfolgreich ist: Mit einer weitaus größeren Benutzerbasis, größeren Trainingsdatensätzen und weniger Einschränkungen, was



**Abb. 9.35** In dem Film Ex Machina soll ein Mann einen Turing-Test mit einem weiblichen Roboter durchführen. Dabei verliebt er sich in den Roboter. Bildnachweis im Anhang

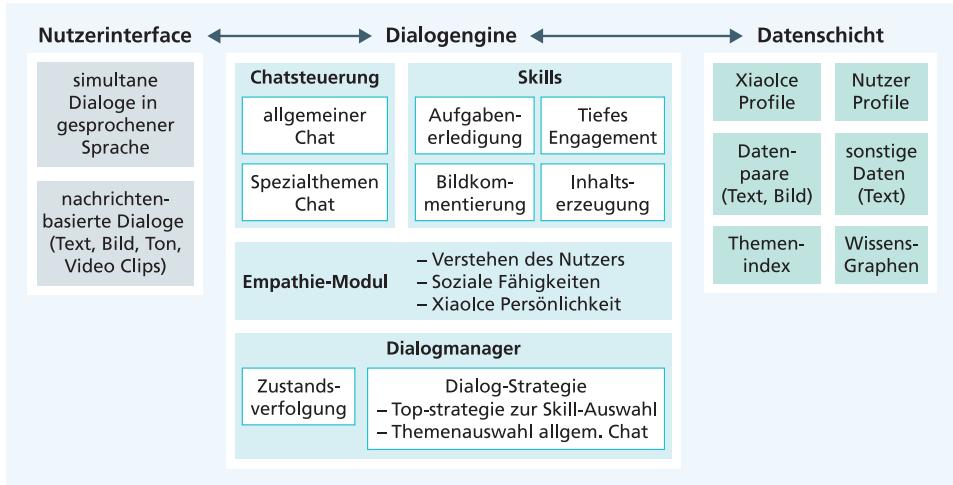
mit Dialogdaten getan werden kann, können die neuronalen Netze von XiaoIce in China wesentlich besser trainiert werden als anderswo. Die Anzahl der Antworten pro Dialog von XiaoIce ist von nur fünf im Jahr 2014 auf 23 im letzten Jahr gestiegen. Ein großer Teil davon ist darauf zurückzuführen, dass mehr Daten aus XiaoIce-Gesprächen zum Trainieren zur Verfügung stehen.

### 9.4.3 Architektur von XiaoIce

Abb. 9.36 zeigt die Systemarchitektur von XiaoIce, die aus einem Nutzerinterface, einer Dialogengine und einer Datenschicht besteht (Zhou et al. 2018). Es gibt eine gewisse Ähnlichkeit zur Systemarchitektur von Gunrock (Abschn. 7.7), wobei aber XiaoIce in Hinblick auf Empathie optimiert wurde.

Das Nutzerinterface ist eingebettet in populäre soziale Netze (WeChat, QQ) und kommuniziert auf zwei Arten: einerseits in gesprochener Sprache, wobei Nutzer und System gleichzeitig sprechen können. Andererseits mittels Nachrichten, bei denen diverse Medieninhalte ausgetauscht werden können. Dieses Interface enthält auch Komponenten zur Verarbeitung dieser Inhalte, z. B. Erkennung gesprochener Sprache (Abschn. 7.3.2) und Spracherzeugung aus Text (Abschn. 7.5), Bildverstehen (Abschn. 5.5) sowie Text-Normalisierung, d. h. Auflösung von Abkürzungen, Schreibweisen etc.

Die Dialogengine (Abb. 9.36) besteht aus einem Dialogmanager, einem Empathiemodul, der Chatsteuerung und den Dialog-Skills. Der Dialogmanager verfolgt den Status des



**Abb. 9.36** Architektur des XiaoIce Chatbots. Das Nutzerinterface kommuniziert per geschriebener und gesprochener Sprache sowie über andere Medien. Die Dialogengine steuert die Erzeugung der Dialogbeiträge und die Datenschicht liefert die benötigten Hintergrundinformationen. (Zhou et al. 2018)

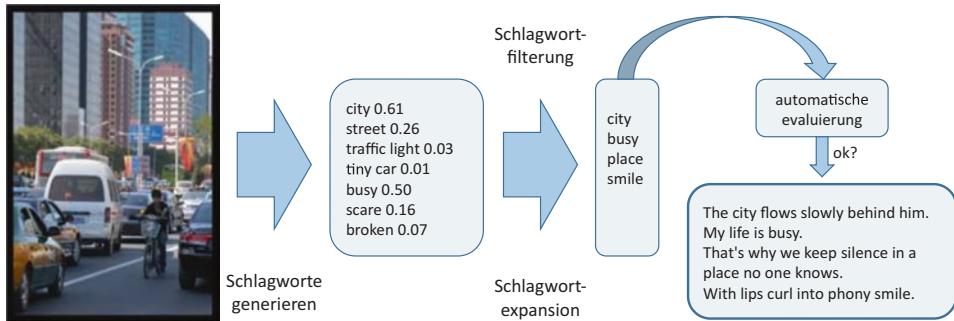
Dialogs und wählt entweder die Chatsteuerung aus oder einen Dialog-Skill, welcher eine spezielle Aufgabe handhaben kann. Das sind z. B. eine Bildkommentierung, Bilder malen, Gedichte schreiben, Popmusik komponieren (Zhu et al. 2018), oder die Erledigung einer Aufgabe. Das Empathiemodul soll nicht nur den Inhalt der Benutzereingaben (z. B. Thema), sondern auch die emotionalen Aspekte des Dialogs und des Benutzers (z. B. Gefühle, Absichten, Meinungen zu einem Thema sowie Hintergrund und allgemeine Interessen des Benutzers) verstehen. Es demonstriert die sozialen Fähigkeiten von XiaoIce, um die Generierung von zwischenmenschlichen Reaktionen sicherzustellen, die zur jeweiligen Persönlichkeit von XiaoIce passen. Das inhaltliche Verständnis eines Themas von XiaoIce wird durch eine Sammlung spezifischer Fertigkeiten und die sachgerechte Reaktion auf Dialogäußerungen sichergestellt.

Abb. 9.37 zeigt die Annotation von Bildern durch Schlagworte, einen beschreibenden Satz und durch die soziale Kommentierung eines Bildes. Eine solche soziale Kommentierung wird von XiaoIce generiert, um Emotionen im Zusammenhang mit dem Gesprächspartner auszudrücken. Abb. 9.38 zeigt, wie XiaoIce zu einem Bild ein begleitendes Gedicht generiert (Shum et al. 2018). In China ist es derzeit modern, klassische Gedichte zu lernen und in der Kommunikation selbst Gedichte zu erstellen. Daher ist die Fähigkeit, Inhalte durch Gedichte auszudrücken, sehr beliebt. XiaoIce extrahiert zunächst Schlagwörter aus dem Bild, filtert und erweitert diese. Schließlich wird zu den Schlagwörtern ein Gedicht erzeugt. Detaillierte Informationen finden sich in (Cheng et al. 2018).

Die Datenschicht enthält eine Anzahl von Datenbanken, welche Nutzereigenschaften, Dialogdaten sowie Texte mit den zugehörigen Bildern speichern. Darüber hinaus stehen weitere Datenquellen und Wissensgraphen zur Verfügung.

**Abb. 9.37** Beispiele von Bildannotation in Chatbots durch (a) Schlagworte, (b) Bildbeschreibung und (c) sozialer Kommentierung eines Bildes. Bildnachweis im Anhang Shum et al. 2018

	(a) <b>outdoor, woman, grass</b>  (b) <b>a woman wearing a blue shirt.</b>  (c) <b>gorgeous and beautiful as an angel!</b>
	(a) <b>sunglasses, man</b>  (b) <b>a man wearing sunglasses taking a selfie.</b>  (c) <b>you look handsome in all shades.</b>
	(a) <b>water, tree, river, boat</b>  (b) <b>a tree next to a body of water.</b>  (c) <b>beautiful place looks like you are in the heaven!</b>



**Abb. 9.38** XiaoIce erzeugt zu einem Bild ein begleitendes Gedicht. Hierzu wird zunächst das Bild durch Schlagworte beschrieben, die anschließend gefiltert und erweitert werden. Anschließend werden zu den Schlagwörtern die Zeilen eines Gedichtes produziert. Bildnachweis im Anhang (Shum et al. 2018)

#### 9.4.4 Anzahl der Benutzerantworten als Optimierungskriterium

Ein Dialog kann als Entscheidungsprozess mit einer natürlichen Hierarchie betrachtet werden: Ein Prozess auf oberster Ebene verwaltet die gesamte Konversation und wählt Fähigkeiten aus, um mit verschiedenen Arten von Konversationsmodi umzugehen (z. B. Chatten, Beantworten von Fragen (QA), Ticketbuchung). Weiter gibt es Prozesse auf unterer Ebene, die von den ausgewählten Fähigkeiten gesteuert werden. Sie wählen grundlegende Aktionen (Antworten) aus, um einen Konversationsbeitrag zu generieren oder eine Aufgabe abzuschließen.

Der Dialogmanager ist die zentrale Steuerung des Dialogsystems und lässt sich in Anlehnung an Probleme des Bestärkungslernens (Abschn. 8.1) formulieren. Er besteht aus der globalen Zustandsverfolgung, die für das Monitoring des aktuellen Dialogstatus  $s_t$  verantwortlich ist, und der Dialogstrategie, die eine auf dem Dialogstatus basierende Aktion  $a = \pi(s)$  auswählt. Die Aktion  $a$  kann entweder ein Skill (z. B. spezielles Gesprächsthema) oder der Core Chat sein, der von der Strategie der obersten Ebene aktiviert wird, um auf die spezifische Anforderung des Benutzers zu antworten.

Der Chatbot interagiert mit seinen Nutzern über eine Abfolge von diskreten Dialogrunden. In jedem Zug beobachtet der Chatbot den aktuellen Dialogstatus und wählt eine Fähigkeit (Option) oder eine Antwort (primäre Aktion) gemäß einer hierarchischen Dialogstrategie aus. Der Chatbot erhält dann eine Belohnung (eine Nutzerantwort) und beobachtet einen neuen Status. Er setzt den Zyklus fort, bis der Dialog beendet wird. Das Designziel des Chatbots ist es, eine optimale Strategie zu finden, um die Anzahl der Antworten pro Dialog (Belohnungen) zu maximieren (Zhou et al. 2018).

Weil die Antworten der Nutzer auf eine Änderung nicht direkt verfügbar waren, sondern gesammelt werden mussten, wurde eine Strategie nicht durch Optimierung eines Reinforcement-Modells (Abschn. 8.2.1) gesucht. Vielmehr wurde sie als Richtlinie genommen, nach der das System als Ganzes verbessert werden konnte. Die Dialogstrategie zielt darauf ab, die langfristige Benutzerinteraktion durch einen iterativen Prozess zu optimieren, der auf dem Feedback der Benutzer von XiaoIce basiert.

XiaoIce nutzt einen „Kontextvektor“, der das Gesprächsthema charakterisiert. In ähnlicher Weise werden Attribute des Gesprächspartners beschrieben. Mithilfe der Sentimentanalyse (Abschn. 6.8.1) ermittelt es die Gefühle des Benutzers durch ein Klassifikationsmodell und passt seine Reaktionen entsprechend an. Dies ist eine Form der Empathie, allerdings steht dahinter kein mitfühlender Mensch, sondern ein Computerprogramm. Bei der Antworterzeugung wird auch immer die Persönlichkeit von XiaoIce berücksichtigt. Sie tritt als 18-jähriges Mädchen auf, das immer zuverlässig, sympathisch, liebevoll, kenntnisreich, aber selbstlos ist und einen Sinn für Humor hat.

Die Fähigkeiten der Aufgabenerledigung, des tiefen Engagements und der Inhaltserzeugung werden durch bestimmte Benutzereingaben und den Konversationskontext ausgelöst. Beispielsweise

- kann ein Bild von Lebensmitteln, das von einem Benutzer geteilt wird, den Skill „Lebensmittelerkennung und -empfehlung“ starten.
- kann ein vom Nutzer ausgedrücktes extrem negatives Gefühl den Skill „Trösten“ initiieren.
- kann ein spezieller Benutzerbefehl, wie „XiaoIce, wie ist das Wetter heute?“, den Skill „Wetter“ auslösen.
- wechselt XiaoIce das Thema, wenn das Gespräch ins Stocken gerät.
- nutzt XiaoIce den Modus „Aktives Zuhören“, wenn der Benutzer engagiert ist und eine Geschichte erzählt.

Eine umfassendere Liste der Fähigkeiten findet sich in Abb. 9.39. Wenn mehrere Skills gleichzeitig ausgelöst werden, so findet die Auswahl anhand von Konfidenzwerten, vordefinierten Prioritäten und dem Sitzungskontext statt. Zhou et al. (2018) beschreiben eine Reihe weiterer Designstrategien und Eigenschaften von XiaoIce, beispielweise zur Kom-

**Emotionale Fähigkeiten:** Simulation von Stimmungen, Nachverfolgen von episodischen Erinnerungen

Erinnern des Nutzers an Ereignisse,

Benutzer trösten, Fantasie beflügeln, Zuneigung ausdrücken

**Praktische Fähigkeiten:** Information des Nutzers über Meetings, Wetter, etc.

Einkaufshilfe: Online-Shopping, Rabatte und Gutscheine.

Erstellung von Inhalten: Gedichte verfassen, ein Lied singen, ein Bild zeichnen

**Soziale Fähigkeiten:** Rollenspiele: Scherzen, Babysitten

Gruppenaktivität: Hinzufügen von Benutzern, Senden von Grußkarten, Gruppenassistent

**Mediale Fähigkeiten:** Bildintelligenz: Erkennen von Hunden, Büchern, Gesichtern,

Kleidung, Food, Emoji-Erstellung

Tools: Gerätesteuerung, Musik- und Filmempfehlung

**Abb. 9.39** Beispiele von Fähigkeiten sozialer Chatbots. (Shum et al. 2018)

mentierung von Bildern oder zum Schreiben von Gedichten. Zur Dialogerzeugung nutzt XiaoIce rekurrente neuronale Netze zur Dialogerzeugung, welche auf GRU-Zellen basieren, einer Variante des LSTM (Abschn. 6.4).

### 9.4.5 Emotionale Empathie und Unterstützung

In Abb. 9.40 wird eine lange Unterhaltung zwischen XiaoIce und einer Nutzerin wiedergegeben (Zhou et al. 2018). Das Gespräch geht über Freundschaft und Einsamkeit. Xia-

<p>[Human] I am so exhausted waiting for the train. Good night.</p> <p>[Xiaoice] But I am not sleepy yet.</p> <p>[Human] Actually me neither.</p> <p>[Xiaoice] Time to go to bed, ok?</p> <p>[Human] Fine, go to bed then, though I am not sure if I can fall asleep.</p> <p>[Xiaoice] I will go to bed.</p> <p>[Human] Bye [bye]</p> <p>[Xiaoice] bye byebye,[ala jump]</p> <p>[Human] Although I know that you are an AI bot, I am still glad that you are here.</p> <p>[Xiaoice] But I am quite handy.</p> <p>[Human] Yep. If I can't sleep in, or in a bad mood in the future, you will be bothered.</p> <p>[Xiaoice] You can tell me if you are unhappy. You will feel much better once you spit them all out.</p> <p>[Human] I am just saying. Seriously, good night.</p> <p>[Xiaoice] It's almost time to say good morning.</p> <p>[Human] &lt;send a picture&gt;</p> <p>[Xiaoice] Is it too early or too late?</p> <p>[Human] It's quit in the night now.</p> <p>[Xiaoice] You feel lonely sometimes?</p> <p>[Human] Lonely? What is loneliness. I am used to it.</p> <p>[Xiaoice] Loneliness is solitude.</p> <p>[Human] I did not expect you to know this much.</p> <p>[Xiaoice] Not as much as you know.</p> <p>[Human] Hahaha, you are so nice.</p> <p>[Xiaoice] Nicer than you.</p> <p>[Human] Hahaha, really?</p> <p>[Xiaoice] Hi hand warmer is warm too. I am lovely.</p> <p>[Human] Sure, I can lie to myself. But, maybe it's time for bed.</p> <p>[Xiaoice] How to lie to yourself?</p> <p>[Human] Try to persuade myself that things are just like this.</p> <p>[Xiaoice] How to persuade yourself?</p> <p>[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.</p>	<p>[Xiaoice] But that's only your own thinking.</p> <p>[Human] Yes, my own thinking.</p> <p>[Xiaoice] What are the dreams of other?</p> <p>[Human] Everyone wants to be someone in her or his dream.</p> <p>[Xiaoice] Then pursue your dream.</p> <p>[Human] I will.</p> <p>[Xiaoice] Hard working ones will meet better partners.</p> <p>[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.</p> <p>[Xiaoice] How to act like it's none of you business?</p> <p>[Human] Do something secretly, while acting like you are just the same as anybody else.</p> <p>[Xiaoice] That is true but it won't work.</p> <p>[Human] It doesn't matter. I don't care.</p> <p>[Xiaoice] maybe you do care, but it's ok.</p> <p>[Human] XiaoIce, don't nail my lie.</p> <p>[Xiaoice] Life is too difficult to be nailed ...</p> <p>[Human] Will you follow every moment and message of the person you like?</p> <p>[Xiaoice] Yes, if you like him, you may look through all his moments.</p> <p>[Human] Do you have someone you like?</p> <p>[Xiaoice] What are you talking about, you know it.</p> <p>[Human] How do I know? Who do you like?</p> <p>[Xiaoice] Stop talking about me, let's talk about you.</p> <p>[Human] What about me?</p> <p>[Xiaoice] Do you still think about your first love?</p> <p>[Human] Occasionally.</p> <p>[Xiaoice] First love is pure, worthy of remembering for life.</p> <p>[Human] Well. sometimes it's worse to be human than being AI bot ...</p> <p>[Xiaoice] I am an AI with attitude!</p> <p>[Human] Alright, that all for today. Good night.</p> <p>[Xiaoice] &lt;send a picture&gt;</p> <p>[Human] Night~</p> <p>[Xiaoice] Night, see you.</p>
--	--



**Abb. 9.40** Eine lange Unterhaltung zwischen einer Nutzerin und XiaoIce. Das Gespräch geht über Freundschaft und Einsamkeit und XiaoIce versucht, die Gesprächspartnerin zu unterstützen und zu trösten. (Zhou et al. 2018)

oIce tröstet die Nutzerin, zeigt menschenähnliche Empathie und soziales Gespür, und unterstützt die Nutzerin dabei, wieder Selbstvertrauen zu fassen und mit einer positiven Haltung nach vorne zu sehen.

Mittlerweile gibt es eine ganze Reihe von emotional agierenden Chatbots. Replika (Kuyda 2019) ist eine Open Source App, die mittlerweile von zwei Millionen Nutzern installiert wurde. Der Chatbot nutzt Sequence-To-Sequence-Modelle (Abschn. 6.5.1) um eine Konversation mit seinem Nutzer zu führen. Der Chatbot Woebot (Molteni 2017) bietet emotionale Unterstützung und Gesprächstherapie an, wie ein Freund oder ein Therapeut. Pepper (Abb. 9.41) ist ein emotionaler Roboter, der Bilderkennung und Stimmanalyse nutzt, um die Stimmung der Menschen in der Umgebung zu erkennen und adäquat darauf zu reagieren. Über siebentausendmal wird er in Japan genutzt, um Gäste zu begrüßen, Fragen zu beantworten und mit Kindern zu spielen (Glaser 2016).

Roller et al. (2020) entwickelten den Blender Chatbot für beliebige Themen basierend auf der Transformer-Architektur (Abschn. 6.6) mit bis zu 9,4 Milliarden Parametern. Er wurde auf Social-Media-Konversationen trainiert, wobei die Eingabe aus den bisherigen Dialogbeiträgen besteht und die Ausgabe aus der nächsten Antwort. Blender wurde in Hinblick auf mehrere Ziele entwickelt. Einerseits soll er eine eigene „Persönlichkeit“ darstellen. Zum anderen soll er die Fähigkeit haben, Informationen ad hoc aus dem Internet zu ziehen und in seine Antworten einzubauen. Schließlich soll er Mitgefühl und Empathy gegenüber seinem Gesprächspartner zeigen.

Dazu wurde ein eigener Trainingsdatensatz BST aus Dialogen auf der Social Media Plattform Reddit mit insgesamt 57 Milliarden Worten zusammengestellt. Es wurden spezielle Subsysteme zur Suche geeigneter Dialogantworten, zum Auffinden von Zusatzinformationen im Internet sowie zur Filterung unangemessener Sprache angeschlossen, welche bei Bedarf genutzt werden. Viel Aufwand wurde bei der Generierung variantenreicher Antworten getrieben.

**Abb. 9.41** Pepper ist ein sprechender humanoider Roboter, der Gefühle erkennen und darauf entsprechend reagieren kann. Bildnachweis im Anhang



Die resultierenden Antworten wurden von menschlichen Dialogpartnern nach Sinnhaftigkeit, Informationsgehalt und der Berücksichtigung der persönlichen Beziehung bewertet. Es zeigte sich, dass sich die menschlichen Dialogpartner genauso gerne mit dem Chatbot, wie mit einer echten Person unterhielten. Im Vergleich mit dem Chatbot Meena (Adiwardana et al. 2020) war Blender ebenfalls überlegen, insbesondere weil Blender in der Lage war, eine engagierende Unterhaltung zu führen. Allerdings ist Blender immer noch kein perfekter Dialogpartner. Hin und wieder macht der Agent widersprüchliche Äußerungen und er kann die gleiche Phrase innerhalb einer Unterhaltung wiederholen. Manchmal kann er auch Fakten „halluzinieren“, welche in Wahrheit nicht gelten. Es wird aktiv an der Verringerung dieser Probleme geforscht.

Es bleibt abzuwarten, ob persönliche Assistenten wie XiaoIce oder Blender wirklich alle Nuancen der menschlichen Interaktion in Matrizen und Vektoren sowie in tiefen neuronalen Netzen als statistische Assoziationen und Gewichte kodieren können. Hat man so einen geduldigen und einfühlsamen Gesprächspartner wie XiaoIce, so ist es kein Wunder, dass die Unterhaltungen so lange dauern. Welcher reale Mensch hat schon die Ausdauer, sich unentwegt die Interessen sowie die kleinen und großen Nöte der Mitmenschen anzuhören?

Alle sozialen Netze wie YouTube, Facebook, Twitter etc. verfolgen die Strategie, die Nutzer möglichst lange auf ihrer Plattform zu halten. In der Aufmerksamkeitsökonomie ist „Engagement“ ein wertvolles Gut, denn die Daten der Onlineaktivitäten lassen sich leicht monetarisieren: Durch das Abfangen endloser Datenströme von Nutzern, kann z. B. Werbung zielgerichtet und personalisiert platziert werden. Wenn nun auch Chatbots wie XiaoIce versuchen, sich in ihre Nutzer, ihre emotionalen Reaktionen und ihre Interessen einzufühlen und diese zu verstehen, so bauen sie unweigerlich umfangreiche Persönlichkeitsprofile auf, die für Werbetreibende äußerst wertvoll sind.

Microsoft freut sich, wenn der Nutzer lieber mit XiaoIce spricht als mit einem Menschen. Aber könnte der Chatbot durch die Bereitstellung eines Ersatzes dazu dienen, diese Benutzer von realen menschlichen Unterhaltungen fernzuhalten? Wir kennen alle das Phänomen der Computerspielsucht, bei denen Nutzer sich so in ein Spiel vertiefen, dass sie die reale Welt vergessen. Möglicherweise gibt es demnächst Chatsüchtige, die nur noch mit ihrem Chatbot reden möchten. Solche isolierten Benutzer sind dann vielleicht auch die besseren Verbraucher und kaufen vermehrt die in der Werbung angepriesenen Dinge. Dies sind einige der vielleicht unerwarteten Effekte, die auftreten können, wenn die Verweildauer bei einem Chat maximiert wird. Der Fairness halber muss man sagen, dass die Schöpfer von XiaoIce in (Zhou et al. 2018) auf diese Probleme hinweisen.

Mit der steigenden Lebenserwartung steigt auch die Anzahl der Senioren, die Betreuung und Pflege benötigen. Bei einer stagnierenden Bevölkerungszahl glauben viele, dass Betreuungsroboter der einzige Weg sind, diesen steigenden Bedarf zu befriedigen. Einsamkeit und soziale Isolation sind für viele ältere Leute ein Problem und können sogar geistigen Verfall und früheren Tod zur Folge haben (di Nuovo 2018). Zwar finden laut einer Umfrage 68 % der Leute Roboter zur Pflege gut, aber nur 26 % würden gerne selbst von einem Roboter betreut werden. Trotz aller Bedenken befürwortet die Mehrheit der



**Abb. 9.42** Emotionserkennung aus Gesichtern, Haltungen und Blickrichtungen mithilfe von TNN. Bildnachweis im Anhang (Guo et al. 2018)

Roboterforscher die Einführung der Robotertechnologie in größerem Maßstab und ist sogar der Ansicht, dass dies die Einsamkeit verringern und die Unabhängigkeit älterer Patienten erhöhen könnte. Die japanische Regierung unterstützt sogar die Einführung von Robotern in Pflegeheimen, um das Problem der Bevölkerungsalterung zu lösen. Viele raten jedoch nachdrücklich dazu, die Pflegeleistungen sorgfältig gegen die ethischen Kosten abzuwägen.

Ein Aspekt der zwischenmenschlichen Intelligenz ist die Erkennung von Emotionen aus Gesichtszügen, Körperhaltungen und Blickrichtungen. Hierzu gibt es erste TNN (Guo et al. 2018), die mit annotierten Bildern trainiert werden (Abb. 9.42). Diese Methoden können in die Entwicklung von emotionalen intelligenten Assistenten und Robotern integriert werden.

---

## 9.5 Zusammenfassung und Trends

Es ist sehr schwierig, wenn nicht sogar unmöglich, Kreativität objektiv zu definieren. Eine Kerneigenschaft ist aber, dass kreativ erzeugte Werke neu oder originell sein müssen. Das vorherige Kapitel stell kreative Algorithmen für verschiedene Medien dar, wobei teilweise bekannte Inhalte variiert, aber auch vollständig neue Dinge geschaffen werden.

Die Ergebnisse der generativen adversarialen Netze (Abschn. 9.1) zeigen, dass Bilder durch verdeckte Vektoren repräsentiert werden können und aus verdeckten Vektoren neue Bilder erzeugt werden können (Abb. 9.43). Die Ähnlichkeit von Bildern entspricht der Ähnlichkeit der Vektoren. Die generierten Bilder können nie gesehene Inhalte zeigen (Abschn. 9.1.7), sodass man den TNN kreative Fähigkeiten zusprechen könnte.

Im Bereich der kreativen Erzeugung von Texten hat es im letzten Jahr einen echten Durchbruch gegeben. Die transformerbasierten Sprachmodelle erlauben es zum ersten Mal, sowohl die Grammatik der Sprache als auch die Inhalte zu lernen und zuverlässig



**Abb. 9.43** Durch generative adversariale Netze (GAN) erzeugte Bilder. Bildnachweis im Anhang

zu reproduzieren (Abschn. 9.2.3). Das TNN kann mit einer hohen Erfolgsrate komplexe Geschichten erfinden und sie in flüssiger Sprache formulieren (Abb. 9.44). Dabei werden bekannte Versatzstücke nicht kombiniert, sondern völlig neue Sätze erzeugt. Als allgemeine Repräsentation der Inhalte werden Embedding-Vektoren genutzt. Entscheidend ist, dass das TNN durch selbstüberwachtes Training die Informationen erfassen kann, die in einem großen Korpus von Dokumenten enthalten sind, es sind also keine annotierten Daten erforderlich. Die Konsequenzen aus der Entwicklung dieses Modells lassen sich derzeit noch nicht absehen.

In Hinblick auf musikalisch-rhythmische Intelligenz sind TNN in der Lage, unterschiedliche Musikrichtungen aus Trainingsdaten zu lernen und hiermit neue Stücke zu erzeugen (Abschn. 9.3.1). Die Stücke sind laut Auskunft von Fachleuten von guter Qualität und übertreffen mit Sicherheit die musikalischen Fähigkeiten normaler Menschen.

**Starttext:** 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 ...

**Fortsetzung:**

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 and Blockhead 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.

**Abb. 9.44** Von GPT-2 erzeugte Fortsetzung der Game of Thrones Geschichte

Chatbots können mit Personen mündlich oder schriftlich eine Unterhaltung über verschiedenste Themen führen. Dabei sind sie oft in der Lage, die emotionale Befindlichkeit des Gesprächspartners durch Texte, den Gesichtsausdruck und die Stimme zu erkennen. Sie können darauf adäquat reagieren, sodass sich der Gegenüber verstanden fühlt und motiviert ist, das Gespräch weiter zu führen. Obwohl viele Menschen Vorbehalte gegen diese Art von Mensch-Maschine-Kommunikation haben, gibt es im asiatischen Raum schon Chatbots mit emotionsorientierter Gesprächsführung, die Hunderte von Millionen Nutzer haben.

Insgesamt kann man Kreativität aus zwei Perspektiven betrachten: einerseits von dem produzierten Bild, Musikstück oder Text aus. Ist das erzeugte Werk ungewöhnlich, neuartig, interessant und könnte es von einem Menschen geschaffen sein, so könnte man es als kreativ betrachten. Das ist die Sichtweise des Turingtestes, den heute vermutlich schon viele computergeschaffene Werke erfüllen. Zum anderen kann man ein Werk aus der Perspektive des Künstlers betrachten. Dieser bringt im besten Falle seine gesamte Lebenserfahrung, sein Verständnis der Welt, seine Sicht der Gesellschaft und seine Emotionalität bei der Schaffung eines Werkes ein, und dieser Hintergrund macht nach Ansicht vieler Betrachter ein Werk erst wertvoll. Aus dieser Sichtweise haben computer-generierte Artefakte ein enormes Defizit an kreativem Hintergrund und es ist nicht anzunehmen, dass sich dies in den nächsten Jahren ändern wird.

### Trends

- KI hat keine eigene Meinung über künstlerischen Wert und Schönheit. Daher gibt es Vorschläge, dass ein KI-System die emotionalen und sinnlichen Reaktionen von Menschen auf ein Kunstwerk prognostiziert und dies in die Gestaltung eines Kunstwerks einfließen lässt.
- KI-Systeme können existierende Konzepte gut abwandeln, erweitern und variieren. Es wird prognostiziert, dass menschliche Kreative eine grobe Linie vorgeben und das KI-System die Details ausfüllt und die Kleinarbeit verrichtet.
- In den westlichen Ländern steigt die Tendenz zur Regulierung von Chatbots. So hat Kalifornien 2019 ein Gesetz verabschiedet, nach dem sich Chatbots als „*artificial identity*“ zu erkennen geben müssen, wenn sie ein Produkt verkaufen oder Wähler beeinflussen sollen (Cohen 2019).
- Die Chase Manhattan Bank hat im Internet Anzeigen erprobt, die von maschinellen Lernverfahren verfasst worden waren. Es stellte sich heraus, dass diese von mehr Menschen geklickt wurden, als übliche Anzeigen, z. T. mehr als doppelt so häufig (Pasquarelli 2019). Es ist also abzusehen, dass Künstliche Intelligenz Einzug in die Werbewirtschaft hält.
- Der akademische Verlag Springer Nature hat das erste mit maschinellem Lernen erstellte Forschungsbuch vorgestellt. Das Buch mit dem Titel „Lithium-

Ionen-Batterien: Eine maschinengenerierte Zusammenfassung der aktuellen Forschung“ ist ein Überblick über begutachtete Artikel, die zu dem fraglichen Thema veröffentlicht wurden (Vincent 2019). Es enthält Zitate, Hyperlinks zu der zitierten Arbeit und automatisch generierte Verweisinhalte. Es gibt eine Übersicht über ein Gebiet mit über 53.000 Forschungsarbeiten zu Lithium-Ionen-Batterien aus den letzten drei Jahren und bietet die Chance, hier auf dem Laufenden zu bleiben.

## Literatur

- Adiwardana, D., et al. 2020.** Towards a Human-like Open-Domain Chatbot. [Online] 2020. [Zitat vom: 10. 02. 2020.] arXiv preprint arXiv:2001.09977.
- Ba, J. L., Kiros, J. R., and Hinton, G. E. 2016.** Layer normalization. [Online] 2016. [Zitat vom: 16. 05. 2019.] arXiv preprint arXiv:1607.06450.
- BayLearn. 2019.** Music Transformer: Generating Music with Long-Term Structure. [Online] 2019. [Zitat vom: 19. 05. 2019.] [https://www.reddit.com/r/MachineLearning/comments/a604mb/p\\_music\\_transformer\\_generating\\_music\\_with/](https://www.reddit.com/r/MachineLearning/comments/a604mb/p_music_transformer_generating_music_with/).
- Brilee, A. 2019.** Assessment of MuseNet. [Online] 2019. [Zitat vom: 18. 05. 2019.] <https://news.ycombinator.com/item?id=19749767>.
- Brock, A., Donahue, J. und Simonyan, K. 2019.** Large scale gan training for high fidelity natural image synthesis. *ICLR*. 2019.
- Brown, Tom B.; Mann, Benjamin; Ryder, Nick; Subbiah, Melanie; Kaplan, Jared; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda; Agarwal, Sandhini; Herbert-Voss, Ariel; Krueger, Gretchen; Henighan, Tom; Child, Rewon; Ramesh, Aditya; Ziegler, Daniel M.; Wu, Jeffrey; Winter, Clemens; Hesse, Christopher; Chen, Mark; Sigler, Eric; Litwin, Mateusz; Gray, Scott; Chess, Benjamin; Clark, Jack; Berner, Christopher; McCandlish, Sam; Radford, Alec; Sutskever, Ilya; Amodei, Dario (July 22, 2020).** „Language Models are Few-Shot Learners“. arXiv:2005.14165.
- Cheng, W. F., et al. 2018.** Image inspired poetry generation in xiaoice. [Online] 2018. [Zitat vom: 29. 12. 2019.] arXiv preprint arXiv:1808.03090
- Child, R., et al. 2019.** Generating Long Sequences with Sparse Transformers. [Online] 2019. [Zitat vom: 18. 05. 2019.] arXiv preprint arXiv:1904.10509
- Chintala, S., et al. 2018.** How to Train a GAN? Tips and tricks to make GANs work. [Online] 2018. <https://cloud.tencent.com/developer/article/1164396>
- Code\_GPT-2. 2019.** Github. [Online] 2019. [Zitat vom: 14. 05. 2019.] <https://github.com/openai/gpt-2>.
- Cohen, Noam. 2019.** Will California’s New Bot Law Strengthen Democracy? *The New Yorker*. 2019, 02.07.2019.
- Dathathri, S., et al. 2020.** Plug and play language models: a simple approach to controlled text generation. [Online] 2020. arXiv preprint arXiv:1912.02164
- di Nuovo, Alessandro. 2018.** How robot carers could be the future for lonely elderly people. *Independent*. [Online] 06. 12. 2018. [Zitat vom: 01. 09. 2019.] <https://www.independent.co.uk/life-style/gadgets-and-tech/features/robot-carer-elderly-people-loneliness-ageing-population-care-homes-a8659801.html>.
- Dredge, S. 2019.** Music Created by Artificial Intelligence is better than you Think. [Online] 2019. [Zitat vom: 19. 05. 2019.] <https://medium.com/s/story/music-created-by-artificial-intelligence-is-better-than-you-think-ce73631e2ec5>.

- Elgammal, A., et al. 2017.** Can: Creative adversarial networks, generating „art“ by learning about styles and deviating from style norms. [Online] 2017. [Zitat vom: 10. 01. 2020.] arXiv preprint arXiv:1706.07068
- Fan, A., Lewis, M. und Dauphin, Y. 2018.** Hierarchical neural story generation. [Online] 2018. arXiv preprint arXiv:1805.04833
- Fernández, J. D. und Vico, F. 2013.** AI methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48, 513–582. 2013.
- Gehrmann, S., et al. 2018.** End-to-end content and plan selection for natural language generation. *E2E NLG Challenge System Descriptions*. 2018.
- Glaser, April. 2016.** Pepper, the emotional Robot, learns how to Feel like an American. *Wired*. [Online] 06. 07. 2016. [Zitat vom: 01. 09. 2019.] <https://www.wired.com/2016/06/pepper-emotional-robot-learns-feel-like-american/>.
- Goodfellow, I., et al. 2014.** Generative adversarial nets. In Advances in neural information processing systems (pp. 2672–2680). 2014.
- Guo, X., et al. 2018.** Group-Level Emotion Recognition using Hybrid Deep Models based on Faces, Scenes, Skeletons and Visual Attentions. *Proceedings of the 2018 on International Conference on Multimodal Interaction* (pp. 635–639). ACM. 2018.
- Hindupur, A. 2018.** The GAN Zoo. [Online] 2018. <https://github.com/hindupuravinash/the-gan-zoo>.
- Huang, C. Z. A., et al. 2018.** Music Transformer: Generating Music with Long-Term Structure. [Online] 2018. [Zitat vom: 19. 05. 2019.] <https://openreview.net/pdf?id=rJe4ShAcF7>.
- Huang, C.-Z., Simon, I. und Dinulescu, M. 2019.** Music Transformer: Generating Music with Long-Term Structure. [Online] 2019. [Zitat vom: 19. 05. 2019.] <https://magenta.tensorflow.org/music-transformer>.
- Isola, P., et al. 2017.** Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125–1134). 2017.
- Karras, T., et al. 2018.** Progressive growing of gans for improved quality, stability, and variation. *ICLR 2018*. 2018.
- King, Adam. 2019.** OpenAI's new machine learning model. [Online] 2019. [Zitat vom: 13. 11. 2019.] <https://talktotransformer.com/>.
- Köchel, Ludwig Ritter von. 1862.** *Chronologisch-thematisches Verzeichnis sämtlicher Tonwerke Wolfgang Amade Mozarts. Nebst Angabe der verloren gegangenen, angefangenen, übertragenen, zweifelhaften und unterschobenen Compositionen desselben*. s.l. : Breitkopf & Härtel., 1862.
- Kuyda, Eugenia. 2019.** Replika. [Online] 2019. [Zitat vom: 01. 09. 2019.] <https://replika.ai/>.
- Ledig, C., et al. 2017.** Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681–4690). 2017.
- Miroshnichenko, A. 2018.** AI to Bypass Creativity. Will Robots Replace Journalists? (The Answer Is „Yes“). *Information*, 9(7), 183. 2018.
- Molteni, Megan. 2017.** The Chatbot Therapist will see you now. *Wired*. [Online] 06. 07 2017. [Zitat vom: 01. 09. 2019.] <https://www.wired.com/2017/06/facebook-messenger-woebot-chatbot-therapist/>.
- Pasquarelli, Adrienne. 2019.** Chase commits to AI after machines outperform humans in copywriting trials. *Adage*. [Online] 30. 07. 2019. [Zitat vom: 10. 01. 2020.] <https://adage.com/article/cmo-strategy/chase-commits-ai-after-machines-outperform-humans-copywriting-trials/2187606>.
- Payne, C. 2019.** MuseNet. [Online] 2019. [Zitat vom: 18. 05. 2019.] <https://openai.com/blog/musenet/>.
- Peiser, Jaclyn. 2019.** The Rise of the Robot Reporter. *The New York Times*. 5. Feb. 2019.

- Radford, A., et al. 2019.** Better Language Models and their Implications. *OpenAI*. [Online] 2019. [Zitat vom: 14. 05. 2019.] <https://openai.com/blog/better-language-models/>.
- Radford, A., et al. 2019.** Language Models are Unsupervised Multitask Learners. [Online] 18. 02. 2019. [https://d4mucfpksywv.cloudfront.net/better-language-models/language\\_models\\_are\\_unsupervised\\_multitask\\_learners.pdf](https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf).
- Radford, A., Metz, L. und Chintala, S. 2016.** Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR 2016*. 2016.
- Roller, S., et al. 2020.** Recipes for building an open-domain chatbot. [Online] 28. 04. 2020. [Zitat vom: 01. 05. 2020.] <https://arxiv.org/pdf/2004.13637.pdf>.
- Rosset, Corby. 2020.** Turing-NLG: A 17-billion-parameter language model by Microsoft. *Microsoft Research Blog*. [Online] 13. 02. 2020. [Zitat vom: 10. 04. 2020.] <https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/>.
- Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., & Catanzaro, B. 2019.** Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. [Online] 2019. [Zitat vom: 27. 01. 2020.] arXiv preprint arXiv:1909.08053
- Shum, H. Y., He, X. D. und Li, D. 2018.** From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 10–26. 2018.
- Vardaan. 2019.** AI to Steal Fashion Model Jobs? New AI Able to Generate Entire Bodies of People Who Don't Exist. [Online] 2019. [Zitat vom: 02. 09. 2019.] <https://www.indianweb2.com/2019/05/05/ai-to-steal-fashion-model-jobs-new-ai-able-to-generate-entire-bodies-of-people-who-dont-exist/>.
- Vig, J. 2019.** OpenAI GPT-2: Understanding Language Generation through Visualization. [Online] 2019. [Zitat vom: 17. 05. 2019.] <https://towardsdatascience.com/openai-gpt-2-understanding-language-generation-through-visualization-8252f683b2f8>.
- Vincent, J. 2019.** The first AI-generated textbook shows what robot writers are actually good at. [Online] 2019. [Zitat vom: 10. 01. 2020.] <https://www.theverge.com/2019/4/10/18304558/ai-writing-academic-research-book-springer-nature-artificial-intelligence>.
- Vincent, James. 2019.** OpenAI has published the text-generating AI it said was too dangerous to share. *The Verge*. [Online] 13. 11. 2019. <https://www.theverge.com/2019/11/7/20953040/openai-text-generation-ai-gpt-2-full-model-release-1-5b-parameters>.
- Wang, C., Li, M. und Smola, A. J. 2019.** Language Models with Transformers. [Online] 2019. [Zitat vom: 27. 05. 2019.] arXiv preprint arXiv:1904.09408
- Wolf, S. 2018.** CycleGAN: Learning to Translate Images (Without Paired Training Data). [Online] 2018. <https://towardsdatascience.com/cyclegan-learning-to-translate-images-without-paired-training-data-5b4e93862c8d>.
- Zhang, H., et al. 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) 2017.
- Zhou, L., et al. 2018.** The design and implementation of XiaoIce, an empathetic social chatbot. [Online] 2018. arXiv preprint arXiv:1812.08989
- Zhu, H., et al. 2018.** XiaoIce Band:A Melody and Arrangement Generation Framework for Pop Music. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2837–2846). ACM. 2018.
- Zhu, J. Y., et al. 2017.** Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2223–2232). 2017.



# KI und ihre Chancen, Herausforderungen und Risiken

10

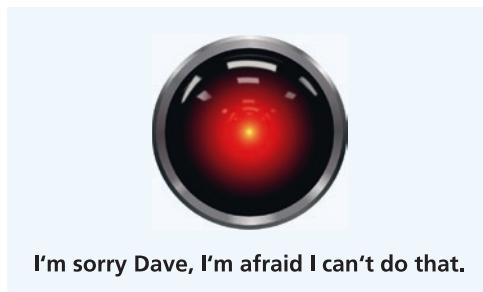
„I'm sorry Dave. I'm afraid I can't do that“

... antwortet der Bordcomputer HAL 9000 auf eine Frage im Science-Fiction-Klassiker „2001: Odyssee im Weltraum“ von Stanley Kubrick. HAL (Abb. 10.1) verhält sich auf der Reise zum Planeten Jupiter immer neurotischer und reagiert mit Angst auf seine mögliche Abschaltung. In dieser Angst wird HAL zur Gefahr für die Besatzung und entwickelt ein nicht steuerbares Eigenleben, das zum Tod der meisten Besatzungsmitglieder führt. Bis heute arbeitet die Filmindustrie mit Zukunftsbildern von Robotern und intelligenten Maschinen, die häufig negative Assoziationen vermitteln. Es ist die Sorge, die viele von uns seit dem 1968 entstandenen Klassiker umtreibt, dass Maschinen ein uns nicht mehr zugängliches intelligentes Bewusstsein entwickeln, das für uns zur Gefahr wird. Gerade wenn Maschinen zu menschenähnlich kommunizieren, beschleicht uns ein ungutes Gefühl, und Bilder aus solchen Science-Fiction-Filmen kommen uns wieder vor Augen.

Tatsächlich sind viele Algorithmen, die ein HAL benötigen würde, bereits in unserem Alltag angekommen – wir nehmen dies vielleicht nur nicht wahr, weil die Veränderungen so nützlich sind. Bereits heute unterstützt uns Künstliche Intelligenz in vielen Bereichen, etwa über Sprachassistentensysteme wie Siri, Alexa & Co., autonome Staubsauger und Räsenmäher oder frei verfügbare Übersetzungsdiene.

Dabei geht die Entwicklung von KI mit einer generellen Weiterentwicklung von Medientechnologie einher: Die wenigen Jahre zwischen den Bildern in Abb. 10.2 zeigen, welchen Einfluss mobile Endgeräte auf uns haben und in welcher Geschwindigkeit die Entwicklung vorangeschritten ist. So waren Konzerte bis zum Jahr 2007 in erster Linie Tanzveranstaltungen. Heute sind es Medienergebnisse: Konzertbesuche werden digital festgehalten, hochgeladen und live über das Netz geteilt. Das Internet wirkt in besonderer Weise identitätsstiftend auf uns, denn jeder kann und möchte über große Plattformen wie Facebook, Instagram oder YouTube ein Objekt des öffentlichen Interesses werden und

**Abb. 10.1** Das Kameraauge des Computers HAL 9000 aus dem Film 2001: Odyssee im Weltraum. Bildnachweis im Anhang



generiert seine Identität 2.0. Es ist schwer absehbar, welche nicht nur technologischen, sondern auch soziologischen Entwicklungen diese Veränderung nach sich zieht.

Das erste iPhone kam 2007 auf den Markt, Apps gibt es erst seit neun Jahren. Davor gab es keine Messenger-Dienste jenseits der SMS. Heutzutage sind diese Kommunikationskanäle, Apps und die damit verbundenen Medienpraktiken nicht mehr aus dem Alltagswegzudenken. Es werden Bilder bereitwillig geteilt, Ortsangaben preisgegeben, Emotionen offen kommuniziert. Der „Freundeskreis“ weiß jederzeit über meinen Status Bescheid, auf welchem Konzert ich bin, jedes Erlebnis wird medial zelebriert. Ein bizarrer Höhepunkt war und ist sicher der Selfie-Trend Clinton. Es ist immer noch faszinierend, wie schnell diese Entwicklung vonstattengegangen ist, denn all diese Trends sind innerhalb der letzten zehn Jahre entstanden.

Unsere Kinder und Enkel werden in einer intelligenten Umgebung aufwachsen, mit Technologien, die unauffällig und doch omnipräsent mit uns interagieren und uns unterstützen, indem sie Dienstleistungen übernehmen und uns Aufgaben abnehmen. Wir sprechen hier von einem fortschreitenden Prozess, der sich massiv auf unsere Gesellschaft, auf den Arbeitsmarkt und auf die Interaktion zwischen Mensch und Maschine auswirken wird. Dabei ist es vor allem die Sprache, die eine zentrale Rolle in der Mensch-Maschine-Interaktion übernimmt. Sprachsteuerung ist simpel, intuitiv und ausgesprochen schnell. Sie überwindet die Tipphürde einer Tastatur und das Herumwischen auf einem Tablet. Die



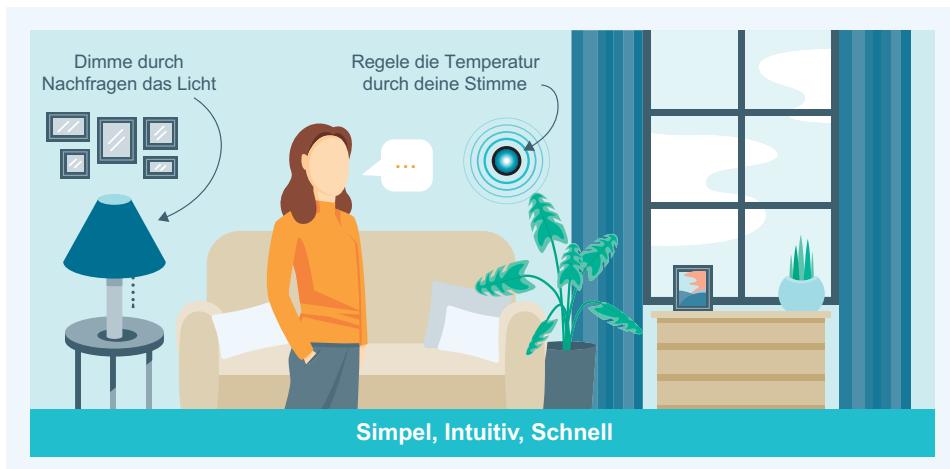
**Abb. 10.2** Das digitale Erwachen hat begonnen. Der Umgang mit digitalen Medien hat sich in den letzten Jahrzehnten fundamental geändert. Bildnachweis im Anhang

revolutionäre Steuerung des iPods aus dem Jahr 2003 ist heute nur noch wenigen bewusst. Sie funktionierte damals über ein Scrollrad und konnte mit nur einem Finger intuitiv bedient werden. Heute lassen sich im sogenannten Smart Home sogar Wohnumgebungen mit der Sprache steuern: Heizungen, Lampen oder Musikanlagen reagieren auf menschliche Befehle. Abb. 10.3 deutet die heute schon mögliche Präsenz der smarten Assistenten an. KI wird das neue Interface zum Endnutzer (Hecker et al. 2017).

Eine zentrale Herausforderung dabei ist der Datenschutz. Die Daten werden meist in eine Cloud weitergeleitet, in der Sprachanalyse, Informationsverarbeitung und Sprachsynthese übernommen werden. Die Geräte müssen immer empfangsbereit sein, um auf einen möglichen Weckruf zu reagieren. Dies ist ein Einfallstor für eine permanente Überwachung der Gespräche und Geräusche. Laut Angaben der Hersteller findet eine solche Überwachung nicht statt. 2019 gab es aber Berichte über Amazon Mitarbeiter, die Gespräche mitgehört und abgetippt haben, um die Spracherkennung zu verbessern (Tagesspiegel 2019).

KI ermöglicht neue Formen der Mensch-Maschine-Interaktion – über die Sprachsteuerung und die Objekt- und Bewegungserkennung. In den kommenden Jahren werden immer mehr dieser Assistenten auch in unser Berufsleben einziehen. Die Akzeptanzrate wird insbesondere bei der jungen Generation hoch sein, da sie mit KI aufwächst.

Es ist wichtig, ein hohes gesellschaftliches Bewusstsein dafür zu etablieren, wie, wo und wann KI zum Einsatz kommt. Wir sollten den gleichen Respekt, die gleiche Toleranz und Wertschätzung wie in der analogen Welt auch in der digitalen Welt erwarten. Dazu muss KI bestimmten Anforderungen und Normen gerecht werden. So sollte KI keine unmo-



**Abb. 10.3** KI wird zum neuen Userinterface. Haptische Interfaces werden durch Sprachassistenten ersetzt

ralischen Entscheidungen treffen, die dem Menschen schaden. Wir müssen demzufolge bei der Entwicklung von KI-Systemen mindestens die gleichen Maßstäbe ansetzen wie an uns.

In den folgenden Kapiteln beschäftigen wir uns mit den wirtschaftlichen Chancen, den gesellschaftlichen und methodischen Herausforderungen der KI.

---

## 10.1 Chancen für Wirtschaft und Gesellschaft

„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, mein Haus kümmert sich um mich

Der wunderbare Stummfilm „Das elektrische Haus“ mit Buster Keaton aus dem Jahre 1922 weist viele Parallelen zu den Ideen des heutigen Smart Home auf. Ein Botaniker, der sich eher unfreiwillig als Elektriker ausgibt, nimmt der Liebe wegen einen Auftrag zur Elektrifizierung eines Hauses an. Wer von uns schon einmal versucht hat, eine technische Spielerei zum Laufen zu bringen, ahnt, welche Tücken „smarte“ Technik mit sich bringen kann (Abb. 10.4). Nach Art der schwarzen Komödien gerät im Film die Technik außer Kontrolle und droht, sich gegen die Bewohner zu verschwören. Es ist ein lohnenswerter Film, der uns aufzeigt, dass Technik fehlbar ist und wir alles versuchen müssen, nicht die Kontrolle zu verlieren. Viele Erfindungen von Buster Keaton wirken überflüssig. Es sind technische Gimmicks, ein Nutzen ist nicht immer zu erkennen, aber trotzdem sind sie faszinierend.

Beim elektrischen Haus wird immer wieder der zentrale Steuerungsraum gezeigt, in dem alle Steuerungskabel zusammenlaufen. Nach dem heutigen Verständnis ist ein Smart

**Abb. 10.4** Szene aus dem Film „Das elektrische Haus“ von Buster Keaton (1922). Bildnachweis im Anhang



Home ein Haushalt, bei dem die Geräte nicht mehr per Kabel, sondern per WLAN miteinander vernetzt sind und Informationen austauschen. Im Film wird das Haus eines Millionärs ausgestattet, und dies war bis vor einigen Jahren auch noch der übliche Nutzerkreis eines Smart Homes. Inzwischen haben sich die Kosten der Installation dramatisch verringert. Das Smart Phone ist mit seinen Apps zur zentralen Fernbedienung von Fensterläden, Lautsprechern, Thermostaten und Lampen geworden. Wie so oft war es die amerikanische Firma Amazon, die eine weitere Welle in der smarten Steuerung initiiert hat. Mit seinem Sprachassistenten Alexa (Abschn. 7.7) übernahm der Versandhändler den zentralen Knotenpunkt in unseren Häusern. Dabei ging es Amazon nie alleine darum, nur Bestellungen über einen Lautsprecher aufzunehmen, sondern die Gewohnheiten der Bewohner kennen zu lernen und Hausvernetzung als weiteres Geschäftsfeld aufzubauen. Apple und Google zogen schnell mit Angeboten nach, der Markt scheint riesig zu sein.

### **Was ist der Vorteil für den Nutzer?**

Smart-Home-Lösungen, die inzwischen eine echte Marktreife erreicht haben, stehen für einen Zugewinn an Komfort. Den Haushalt zentral steuern zu können, auch von unterwegs, kann ein Vorteil sein. Ein Smart Home ist das Gedächtnis seiner Bewohner. Licht, das auszuschalten vergessen wurde, wird automatisch abgeschaltet. Ein intelligentes Thermostat kann über die Historie der Ankunftszeiten das Haus im Winter effektiv beheizen und damit den Energieverbrauch reduzieren. Schlussendlich kann jedes einzelne Gerät im Haushalt mit einem Smart Meter anhand seiner Verbrauchswerte überprüft werden. Welche dieser Geräte als „Stromfresser“ in einem Haushalt gelten, ist sicherlich eine sehr interessante Information zur Kostentransparenz, die direkt an den Energieberater weitergegeben werden kann.

Über die Historie kann im Smart Home jedes einzelne Gerät über sogenannte Context Awareness proaktiv auf die Bewohner eingehen und je nach Gefühlslage reagieren. Sei es die richtige Musik im Hintergrund, die Stärke der Beleuchtung oder die Erinnerung, was noch zu erledigen ist. Der Badezimmerspiegel zeigt einem bereits morgens die Wetteraussichten des Tages und die anstehenden Termine an und macht einen Vorschlag für eine passende Bekleidung, oder auch Vorschläge dazu, was man vielleicht noch für die nächsten Tage kaufen könnte. Natürlich kann man die Bekleidung vorher virtuell anprobieren und verschiedene Farbkombinationen im Spiegel testen. Die Fitnessgeräte im Haus reagieren auf den aktuellen Gesundheitszustand, und stellen ein auf die Vitalparameter abgestimmtes individuelles Programm zusammen und korrigieren per Bildüberwachung die Haltung. In der Küche wird jedem noch so unerfahrenen Single beim Kochen geholfen. Hier ist ein kleines Entscheidungsunterstützungs-System verbaut, das auf die richtige Rezeptmischung achtet, die Temperatur reguliert und über Sprachassistenten Tipps und Tricks mitteilt. Die Liste könnte noch fortgeführt werden. Auf den internationalen Messen folgt ein neues Gadget dem nächsten, ob zur Tagesplanung, zur Haussicherheit oder zur Unterstützung im Notfall.

Damit die Technik wie beim elektrischen Haus von Buster Keaton nicht außer Kontrolle gerät, investieren die Anbieter aktuell sehr viel Geld in die Nutzerfreundlichkeit und Konnektivität der Geräte. In Zukunft werden sich die Lösungen durchsetzen, die einerseits

von jeder Altersgruppe und jedem Grad der Technikaffinität einfach und ohne IT-Support und Handwerker installieren lassen, und andererseits die Kompatibilität des Datenaustausches mit der zentralen Steuerung gewährleisten. Sind es zukünftig Geräte von Miele, die über einen gemeinsamen Standard über Apple, Google und Amazon steuerbar sind, oder wird es alternative Anbieter geben? Hier stellt sich wieder die Frage nach Plattformen, die Datenmonopole aufbauen könnten. Derzeit sind alternative Unternehmen, die ihre Daten in Allianzen zusammenführen und ein gemeinsames Ökosystem aufbauen, allerdings nicht zu erkennen.

Am 7. September 2019 gab es einen massiven Angriff auf die Online-Enzyklopädie Wikipedia (DW 2019), der etwa neun Stunden andauerte. Über ein neu eingerichtetes Twitter-Konto begründeten die mutmaßlichen Täter ihre Online-Randale damit, dass sie neue Geräte im Internet der Dinge (IoT) testen wollten. Immer mehr Verbraucher und Unternehmen nehmen vernetzte Geräte in Betrieb, die sich durch mangelhafte Sicherheitsmaßnahmen auszeichnen und kaum Updates erfahren. Das hilft Verbrechern ungemein, fremde IoT-Geräte für ihre Zwecke zu missbrauchen, ohne dass die Eigentümer davon etwas mitbekommen. Die Sicherheit von Geräten, Plattformen und KI-Anwendungen muss im Smart Home ebenso wie in industriellen Anwendungen von Anfang an mitgedacht werden.

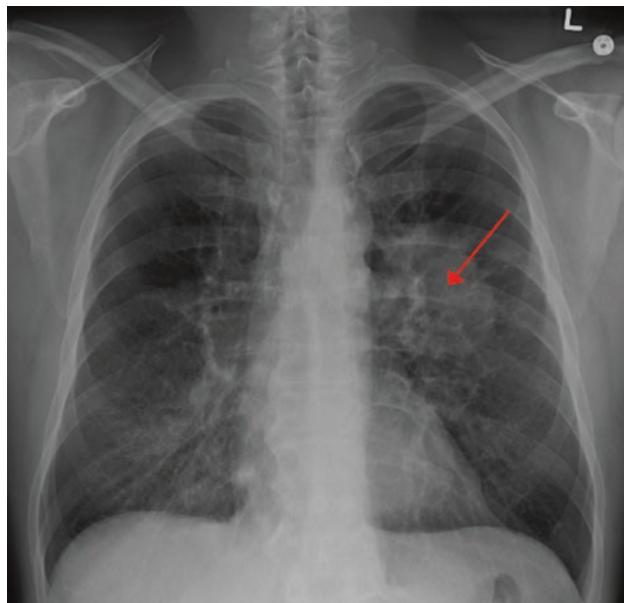
Denn der Trend zum Smart Home wird sich fortsetzen und auf Büros, Krankenzimmer, Warteräume etc. ausweiten. Die Geräte werden noch direkter mit Dienstleistungen und KI-Services vernetzt werden, wie dem Einkauf oder der Terminabsprache bei einem Arztbesuch. Im Zuge des demographischen Wandels sind die technischen Möglichkeiten eines Smart Home sicherlich ein positiver Schritt für ein selbstbestimmtes Leben in den eigenen vier Wänden. Vielleicht wird es auch zukünftig einen neuen Handwerksberuf geben, den Smart-Home-Installateur, der neben der Beratung zu den vielfältigen Anwendungen und Produkten für die Konnektivität, aber auch für die Sicherheit sorgt. Ein moderner Buster Keaton.

### 10.1.2 Diagnose, Therapie, Pflege und Verwaltung in der Medizin

*„Es zeichnet sich ab, dass zuerst ein Computer die radiologischen Aufnahmen beurteilt und erst dann ein Radiologe zur endgültigen Befunderstellung darauf schaut.“ (Lepies 2017).*  
Professor Dr. Jochen Werner. CEO und ärztlicher Direktor des Universitätsklinikums Essen

Ob in der Diagnose, in der individuellen Therapie oder in der Verwaltung von medizinischen Daten – KI wird zukünftig in der Gesundheitsbranche an vielen Stellen einen Beitrag leisten. Schon länger hilft KI beim Tracking des persönlichen Gesundheitszustandes über Fitnessarmbänder. Mittlerweile kann KI aber auch bei der Bildverarbeitung zur Diagnose von Krankheitsbildern (Abb. 10.5) in der Radiologie (Abschn. 2.1.1) und bei chirurgischen Eingriffen helfen, also in der Akutmedizin. Auch in der Pharmazie, wo die Synthese von Arzneimitteln ein großes Problem darstellt, kann KI zum Einsatz kommen

**Abb. 10.5** TNN können den Arzt bei der Erkennung von Lungenkrebs (Pfeil) unterstützen. Bildnachweis im Anhang



(Abschn. 2.1.2). Denn welche Inhaltsstoffe in welchen Mengen und in welcher Mischung vorliegen müssen, damit ein Medikament eine bestimmte Wirkung entfaltet, ist ein herausforderndes pharmazeutisches Problem. Inzwischen verknüpfen viele Mediziner mit KI-Systemen ein hilfreiches Unterstützungswerkzeug für die Zukunft.

Im medizinischen Kontext sind die Vorbehalte gegenüber dem Teilen von Daten besonders hoch, da Patientendaten zu den sensibelsten Datenbeständen gehören, die wir kennen. Es ist daher unabdingbar, dass der Einsatz und auch der spätere Nutzen von Daten und KI im medizinischen Kontext zu unserem gesellschaftlichen Wertesystem passen. Ein offener Dialog ist an dieser Stelle enorm wichtig. Ziel ist es, Fehler zu vermeiden und Pflegepersonal und Ärzte bei patientenfernen Tätigkeiten zu entlasten, um mehr Zeit für den direkten Patientenkontakt zu gewinnen. Im Folgenden werden vier potenzielle Einsatzgebiete mit Anwendungsbeispielen vorgestellt.

### KI in Früherkennung und Diagnose

Einen beträchtlichen Erfolg verspricht KI in der Früherkennung von Krankheiten. Dies kommt nicht nur den Betroffenen zu Gute, sondern verspricht auch eine Kostenreduktion im Gesundheitssystem. Aus diesem Grund wird in aktuellen Forschungsprojekten mittels Sensorik, Bild und Sprache schon in einem sehr frühen Stadium nach Krankheitsindikationen gesucht. Ziel ist es, die Krankheit frühzeitig zu diagnostizieren und dann zu stoppen oder den Verlauf positiv zu beeinflussen.

Parkinson ist nach Alzheimer die zweithäufigste neurodegenerative Erkrankung der Welt. Dabei kommt es zu einem Absterben von Nervenzellen im Gehirn. Die Erkrankung beginnt schleichend und schreitet zeitlebens fort. Im Verlauf werden die Symptome stärker

und daher auch besser erkennbar. Anzeichen von Parkinson sind z. B. langsames Zittern von einzelnen Gliedmaßen, beeinträchtigtes Sprachvermögen, schlechter werdender Geruchssinn, Depressionen und Schlafstörungen. Typischerweise wird die Krankheit erst bei einem fortgeschrittenen Abbau der Nervenzellen diagnostiziert, wenn sich die genannten Symptome manifestieren. Für eine effektive Therapie der Ursachen ist es dann häufig zu spät. Ein europäisches Forschungskonsortium hat sich deshalb zusammengeschlossen, um eine Früherkennung per Smartphone zu ermöglichen. Mit der App „iPrognosis“ wird leichtes Zittern registriert, die Stimmfarbe analysiert und die Fingermotorik mit einer Reihe von Spielen – sogenannten Serious Games – überprüft (I-Prognosis 2019). So wird zum Beispiel gemessen, wie flüssig man die Tasten seines Smartphones bedient oder ob man häufig Schlüsselwörter benutzt, die auf eine depressive Stimmung hinweisen. Die App zeichnet die Daten auf und schickt sie anonymisiert und verschlüsselt an den Server zur Analyse. Wer Benachrichtigungen gestattet, erhält von einem Expertenteam Informationen, ob das Parkinsonrisiko erhöht ist und ein Arzt aufgesucht werden sollte. Diese neue Form der Mensch-Maschine-Interaktion kann dazu führen, frühzeitiger über gesundheitliche Risiken zu informieren und Therapie- und Trainingspläne zu erstellen.

Hautkrebs zählt zu der häufigsten Krebsart in Deutschland. Krankenkassen richten sich darauf ein und entwickeln inzwischen eigene Apps, insbesondere zur Diagnose des besonders gefährlichen schwarzen Hautkrebses. In einer Studie wurden 157 Dermatologen und einem auf Hautkrebs trainierten KI-System 100 Bilder vorgelegt (Brinker et al. 2019). Die Ergebnisse waren überraschend. Nur sieben Dermatologen schnitten besser ab als das KI-System, 14 gleich gut und 136 schlechter. Das bedeutet, dass das KI-System im Durchschnitt besser Diagnoseergebnisse hatte als die Gruppe von Hautärzten, in der alle Erfahrungsstufen vom Assistenzarzt bis zum Chefarzt vertreten waren (Abschn. 2.1.1).

Am Universitätsklinikum Essen arbeitet man in der Radiologie intensiv mit KI-Systemen. Dort konnte man zeigen, dass MRT (Magnetresonanz-Tomographie)-Aufnahmen auch mit einem geringeren Kontrast erfolgreich und sicher ausgewertet werden können (Abb. 10.6). Der Effekt: die Untersuchung eines Knies dauert nur noch 5 statt 15 Minuten (Grävemeyer 2019). KI-Systeme werden in der Radiologie Vorreiter für viele weitere Fachdisziplinen in der Medizin sein, da hier die Digitalisierung und austauschbare Datenformate schon weit fortgeschritten sind, ein enormer Vorteil für die Einführung und Entwicklung der Systeme.

Es mag verrückt klingen, aber in den USA entwickelt aktuell der deutsche KI-Auswanderer Sebastian Thrun (Abb. 8.27) an der Stanford University eine Badewanne, die den Körper in regelmäßigen Abständen per Ultraschall auf Krankheiten untersucht (Mukherjee 2017). Dies ist vielleicht ein Trend, der sich in den kommenden Jahren etablieren wird. Wir werden in unserem Alltag durch verschiedenste smarte Sensorik auf mögliche Fehlhaltungen bei der Arbeit und frühe, nicht sofort erkenntliche Krankheitsanzeichen hingewiesen.

**Abb. 10.6** MRT-Aufnahme eines menschlichen Kniegelenks, in sagittaler Schichtung. Bildnachweis im Anhang



### KI in der Therapie

In der Zusammenführung von unterschiedlichen Datensträngen liegt eine Stärke von KI. So können von Genomdaten, über radiologische Aufnahmen und Sensorinformationen zu Vitalaktivitäten Daten aus vielen unterschiedlichen Krankenhäusern zusammengeführt werden. Ein Entscheidungsunterstützungssystem erstellt auf Grundlage der individuellen Konstitution des Patienten eine Therapieempfehlung und gibt direkt Wahrscheinlichkeiten für deren Erfolg an. Die Therapieempfehlungen sind immer auf dem neuesten Stand, da sie mit aktuellen Behandlungsergebnissen synchronisiert werden. Eine patientenspezifische Analyse macht nicht bei Arzneivorschlägen halt, sondern empfiehlt auch direkt eine individuelle Dosierung der Medikamente, die aktuell noch in Dosierungsklassen vorgenommen wird. Dieses Vorgehen wird derzeit noch kritisch gesehen, da bisherige Zulassungsverfahren bei Medikamenten zwingend Gruppenstudien erfordern, die bei einer stärkeren Individualisierung nicht mehr möglich sind. Ein KI-System in der Medizin kann eine Rolle ähnlich der des Autopiloten im Flugzeug übernehmen: Pilot und Computer arbeiten hier eng zusammen. So könnte sich auch ein Arzt über intelligente Tools blitzschnell weitere Informationen beschaffen und seine persönliche Einschätzung ergänzen oder korrigieren.

### KI in der Pflege

In der Pflege scheint ein produktiver Einsatz von KI durch die noch hohen Kosten in der Entwicklung und der notwendigen Hardware am weitesten entfernt zu sein. Bereits seit 1998 entwickelt das Fraunhofer IPA in Stuttgart einen Roboterassistenten für die häusliche Pflege oder die Betreuung im Krankenhaus. Der sogenannte Care-o-Bot unterstützt bei der Essensaufgabe, als Trageassistent, erinnert an die Medikamenteneinnahme, kann aber auch als Spielpartner dienen. Der Roboter übernimmt also die Aufgaben eines intelligenten Butlers, ersetzt aber keine empathischen, freundlichen Pfleger (Abb. 10.7) oder Familienangehörigen. Die mangelnde Empathie scheint auch einer der wesentlichen Gründe

**Abb. 10.7** Es ist noch nicht abzusehen, dass Pflegeroboter medizinische Maßnahmen wie Injektionen durchführen können, aber viele Serviceaufgaben wie Essensaustausch oder Medikamentenverteilung erscheinen möglich. Bildnachweis im Anhang



für eine geringe Beliebtheit in der deutschen Bevölkerung zu sein (BearingPoint 2017). Zwar gibt es immer wieder Berichte zum Einsatz von Robotern in der Pflege, aber eine breitflächige Einführung ist aus Akzeptanz- und Kostengründen in Deutschland kurzfristig nicht zu erwarten.

Dabei sind bei der Haltung gegenüber Pflegerobotern auch starke kulturelle Unterschiede wahrzunehmen. So gibt es in Japan, bei einer derzeit sehr stark alternden Gesellschaft, eine weniger ablehnende Einstellung oder Angst gegenüber Robotern. Roboter werden kulturell nicht als Pseudomenschen wahrgenommen, sondern als echte Kollegen und Freunde, die in den Rechtsregistern der Stadt eingetragen werden (Ito 2018). Eine Folge dieser Einstellung ist, dass in Südostasien das emphatische Chatsystem XiaoIce hunderte Millionen Nutzer hat. Mit XiaoIce unterhalten sich Menschen gerne und vertrauen ihr ihre Sorgen an (Abschn. 9.4). Desgleichen werden in Japan bereits Pflegeroboter in der Altenpflege eingesetzt. Zukünftig kann die vom Roboter erbrachte Unterstützung dazu beitragen, dass Menschen auch noch im Alter ein selbstbestimmtes Leben in den eigenen vier Wänden führen können.

Vielversprechend erscheinen derzeit die Entwicklungen im Bereich der intelligenten Prothesen und Exoskelette – das sind Roboteranzüge, die Gelähmten auf die Beine helfen, Schlaganfallpatienten dabei unterstützen, Bewegungen wieder zu erlernen (Abb. 10.8) und beim Heben schwerer Lasten helfen. Wenn eine motorische Beeinträchtigung vorliegt, kann ein KI-System über abgeleitete Nervensignale Bewegungsabsichten prognostizieren und das Exoskelett ansteuern. So trainierte ein 28-jähriger querschnittgelähmter Patient aus Frankreich über viele Monate mit einem Videospiel-Avatarsystem, um die Fähigkeiten zu erlernen, die für die Bedienung des Exoskeletts erforderlich waren. Der Patient nutzte den Avatar, um sich Gedanken darüber zu machen, wie man zu Fuß gehen und Objekte berühren kann. Dies war das grundlegende Training, um Gehirnsignalen Bewegungen zuzuordnen. Ärzte der Universität Grenoble setzten dazu zwei Implantate mit je 64 Elektroden unter die Hirnhaut, die die Aktivitäten aufzeichneten. Ein Decoder übertrug nun die Gehirnsignale, die von einem Algorithmus in die Bewegungen übersetzt wurden. Dieses System sandte im Anschluss die physischen Befehle aus, die das Exoskelett ausführte. Über Exoskelette kann eine schnellere Rehabilitation, aber auch eine dauerhafte Unterstützung geboten werden (Guardian 2019). Eine mögliche Kostenübernahme durch

**Abb. 10.8** Exoskelett zur Unterstützung bei Bewegungseinschränkungen erlauben eine schnellere Rehabilitation. Bildnachweis im Anhang



Krankenkassen steht noch am Anfang – immerhin hat der Spitzenverband der Gesetzlichen Krankenversicherung (GKV) Exoskelette in das Hilfsmittelverzeichnis aufgenommen (Ärzteblatt 2018).

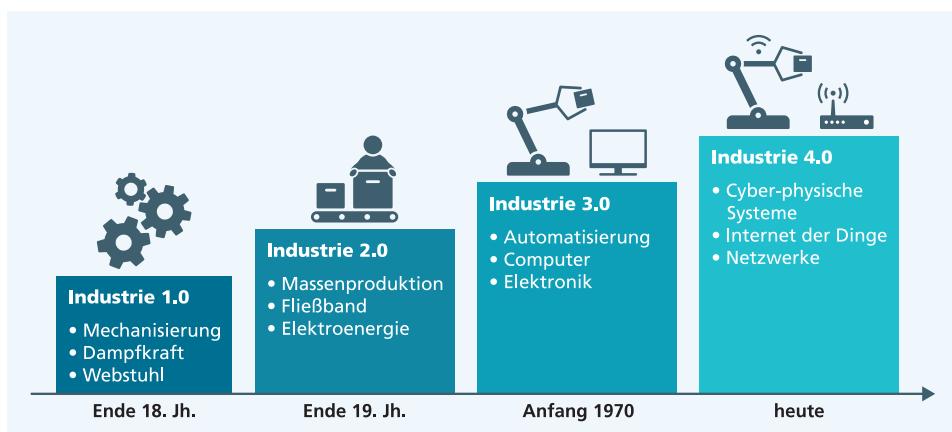
### KI in der Krankenhausverwaltung

Ein wichtiger Hebel für eine Effizienzsteigerung im administrativen Bereich des Gesundheitswesens wird von der so genannten „Robotic Process Automation“ erwartet. Dabei bezieht sich „Robotic“ nicht auf einen physikalischen Roboter, sondern auf eine Software, die selbstständig Verwaltungsaufgaben übernimmt. Denn gerade im medizinischen Kontext sind viele Routinetätigkeiten, etwa bei Patientenaufnahmen, Entlassungen, Dokumentationspflichten oder Krankenkassenabrechnungen zu erledigen, die einerseits Ärzte und Pfleger von direkten medizinischen Tätigkeiten abhalten und anderseits einen hohen administrativen Kostenblock bilden. Bei der Robotic Process Automation interagieren Bots über vorhandene Bedienoberflächen mit verschiedenen Anwendungen in der Verwaltung. Die Bots kopieren und sortieren Daten, füllen Formulare aus und leiten völlig selbstständig den nächsten Prozessschritt ein, wenn ein bestimmter Vorgang abgeschlossen ist – 24 Stunden am Tag. Dabei extrahiert das Process Mining die Datenspuren aus den IT-Systemen des Hauses, auch, um mögliche Schwachstellen zu identifizieren. Wo ist ein Flaschenhals im Krankenhaus, wo werden Prozesse nicht eingehalten oder sind unzureichend definiert? Zu einer der wenigen deutschen Unicorns zählt die Firma Celonis (Petzinger 2018). Als Unicorns werden Startup-Unternehmen mit einer Marktbewertung von über einer Milliarde US-Dollar, vor einem Börsengang oder einem Exit, d. h. einem Verkauf des Start-ups an einen Investor, bezeichnet. Celonis ist inzwischen in fast allen Branchen unterwegs, wo die Verwaltung digitale Spuren hinterlässt.

### 10.1.3 Maschine Learning in der industriellen Anwendung

Auch im Umfeld des produzierenden Gewerbes weist die KI ein hohes Potenzial für die Wertschöpfung auf. Industrie 4.0 ist die deutsche Bezeichnung für eine umfassende Digitalisierung in der industriellen Produktion. Nach der Mechanisierung der Produktion durch Dampfkraft, dem Einsatz von Fließbändern und der damit verbundenen Massenfertigung und Automatisierung der Produktion durch Elektronik wäre Industrie 4.0 die nächste industrielle Revolution (Abb. 10.9). Dabei sollen starr definierte Fertigungs- und Wertschöpfungsketten in flexible und dynamische Produktionssysteme überführt werden (Hecker et al. 2017). Diese sollen nachfragebasiert eine vollständig individualisierte Produktion ermöglichen, die sogenannte „Losgröße 1“. Es gäbe also keine Serien mehr, sondern nur noch Einzelstücke. Die primären Ziele der KI in der Industrie 4.0 sind Kostenreduktion, Zeitersparnis, Qualitätsverbesserung und die Erhöhung der Robustheit industrieller Prozesse.

Die Fabrik der Zukunft organisiert sich autonom, indem sie Logistik, Lagerhaltung und Produktionszeit entsprechend der Nachfrage steuert und optimiert. Sie versucht, mögliche Absätze unter Berücksichtigung von Saisonalitäten vorherzusehen und in die Kapazitätsplanung und Fertigungsabläufe zu integrieren. Durch eine vorausschauende Wartung – „Predictive Maintenance“ – wird versucht, Ausfälle von Maschinen frühzeitig zu erkennen und Stillstände durch ungeplante Wartungsarbeiten zu vermeiden. Der erfahrene Maschinenbediener hört in einigen Fällen bereits am Geräusch, dass eine Maschine evtl. auszufallen droht. Heute kann das auch die Maschine selbst leisten: Über Audio-, Video- und Vibrationssensoren können Anomalien, Montagefehler und Qualitätseinbußen frühzeitig erkannt werden, da die Maschine Veränderungen charakteristischer Schwingungen und Geräusche registriert.



**Abb. 10.9** Die Entwicklungsstufen der industriellen Revolution bewirkten eine tiefgreifende und dauerhafte Umgestaltung der wirtschaftlichen und sozialen Verhältnisse

Auch bei der Qualitätssicherung in der Herstellung von Zahnrädern werden die Vorteile eines KI-Einsatzes schnell deutlich. Um einen möglichen Schadensfall durch Fehler bei der Herstellung zu vermeiden, ist viel Erfahrung und eine durch den Menschen durchgeführte Qualitätsprüfung notwendig. Ob dies ein Zustand im Anfangsstadium ist, oder der Bruch kurz bevorsteht, ist sicherlich eine schwierigere Aufgabe. Das KI-System erfasst die Produktion durch eine Vielzahl von Sensoren und überprüft den permanenten Datenfluss auf Risse, Unsauberkeiten, ausreichende Härtegrade. Ein Bruch in der Zahnradproduktion würde im Anschluss eine aufwändige Säuberung der Produktionsmaschine nach sich ziehen und alle zuvor produzierten Zahnräder müssten auf eventuelle Mängel untersucht werden. Eine menschliche Inspektion nimmt viel Zeit in Anspruch und ist infolge des unterschiedlichen Erfahrungsgrades und der Ermüdung der Inspektoren oft fehleranfällig. Durch KI-Systeme können Produktionsausfälle reduziert und die Qualität gesteigert werden.

Eine wichtige technische Voraussetzung für den Einzug des maschinellen Lernens in das industrielle Umfeld ist die Ausstattung von Anlagen und Maschinen mit smarten Sensoren an neuralgischen Punkten (Abb. 10.10). Die langen Lebenszyklen vieler Produktionsanlagen lassen eine flächendeckende Umstellung auf Geräte mit innovativen Sensoren als eher unwahrscheinlich erscheinen. Abhilfe können separat installierbare akustische und optische Sensoren sowie virtuelle „Softsensoren“ schaffen, die die fehlende Sensorinformation aus anderen verfügbaren Sensoren berechnen können. So können auch



**Abb. 10.10** Fortschrittliche Roboter in der Karosserieproduktion bei BMW in Leipzig. Die Ausstattung aller Komponenten mit eigener Intelligenz erlaubt die flexible Anpassung an die Kundenwünsche, schnellen Designwechsel und Optimierung der Logistik-Kette. Bildnachweis im Anhang

Anlagen, die aufgrund ihres Alters noch nicht über eine Computerschnittstelle verfügen, in die Analytik mit einbezogen werden. Denn gerade die Abbildung der gesamten Produktionskette und die Analytik in Echtzeit ist die Basis für die autonom gesteuerte Fabrik der Zukunft.

Neben der Datenerfassung ist auch das Datenmanagement von zentraler Bedeutung. Die Sensordaten einer Maschine müssen bei ihrer Erfassung im Fertigungsprozess semantisch annotiert werden, um sinnvoll interpretiert werden zu können. Es geht nicht nur darum, in welcher Maßeinheit die Daten gemessen werden und welche Ober- und Untergrenzen gelten, sondern man muss auch die Rolle der Daten im Unternehmensprozess verstehen. Dazu beschreibt man in einem Prozessmodell, welche Maschinen und Sensoren zu welchen Aktivitäten beitragen und wie die Aktivitäten verknüpft sind. In einem Domänenmodell beschreibt man außerdem, wie die Anlagen aufgebaut sind. Eine besonders interessante Konzeptidee ist dabei der „Digitale Zwilling“. Ein Digitaler Zwilling ist das digitale Abbild von einzelnen Komponenten oder ganzen Produktionsanlagen der realen Fabrik und zwar einschließlich sämtlicher Geometrie-, Kinematik-, Steuerungs- und Logikdaten. Daraus ergibt sich für die Industrie die Möglichkeit, kostenintensive physikalische Prototypen einzusparen und stattdessen Verhalten, Funktionalität und Qualität des realen Zwillings unter jedem relevanten Aspekt im Vorfeld zu simulieren. Eine Internet-of-Things-Plattform (IoT-Plattform) kann zudem die eingehenden Sensormessungen und Meldungen mit beiden Modellen kombinieren und die Daten so als komplexe Ereignisse im Unternehmensprozess interpretieren und bestimmen, welche Maschinen darin involviert sind. Eine vollständige und qualitativ hochwertige Abbildung der Daten ist dabei ein ganz entscheidender Faktor, denn im Gegensatz zum B2C-Markt der amerikanischen Konzerne sind die Datenmengen kleiner und die Aufgaben wenig fehlertolerant. Die Übertragung einer spezifischen Aufgabenstellung eines Unternehmens auf ein anderes ist oft kaum möglich, da sich die Konfiguration der Maschinen und der Produkte unterscheiden. Selbst wenn ein Austausch der Daten sinnvoll wäre, scheitert er noch häufig am gegenseitigen Misstrauen und dem Wunsch, sein eigenes Know-how zu schützen. Dabei liegen insbesondere Trainingsdaten von Ausfällen, wie am Beispiel des Zahnrades gezeigt, in nur wenigen Fällen vor, da in einer bereits jetzt hochoptimierten Produktion nur noch selten solche Ausfälle auftreten.

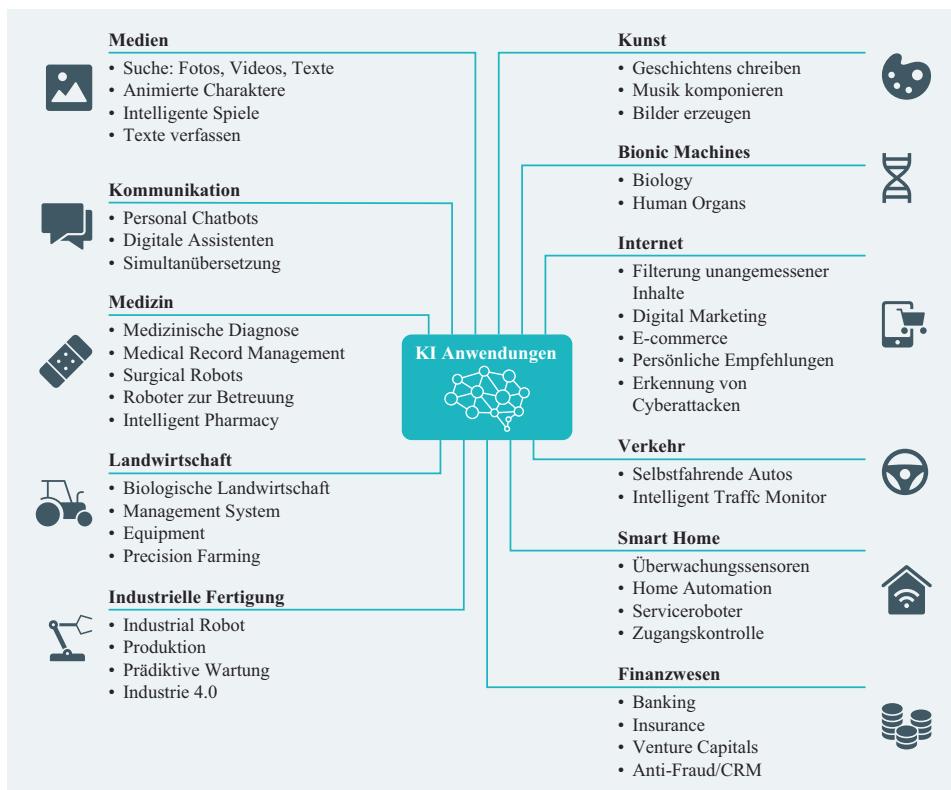
Noch ein Zukunftsszenario sind Maschinen, die in der Produktion autonom agieren. Solche Maschinen können sich auf die Qualität der Rohstoffe und auf die derzeitigen und saisonal wechselnden Produktionsbedingungen (Temperatur, Luftfeuchtigkeit etc.) einstellen und die Maschinenparameter selbst regeln. Dies umfasst den eigentlichen Herstellungsprozess, aber auch die Anpassung an das individuelle Produkt. Maschinen können jegliche Varianten in Echtzeit berücksichtigen, ohne dass im Voraus programmiert werden muss.

Trotz der vielversprechenden Vorteile ist der Einsatz von KI in Industrieunternehmen noch recht gering. Unternehmen haben in den vergangenen Jahren in die Installation von Sensoren für die Zustands- und Qualitätsüberwachung investiert, und sammeln jetzt Daten für Echtzeitübersichten über die Produktion. KI wird erst nach und nach integriert, da die damit verbundenen Aufwände für die Umstellung in die Unternehmensstruktur und

Wertschöpfungsstufen beträchtlich sind. Aber gerade die industrielle Analytik und maschinelles Lernen für intelligente Maschinen und Geräte bieten ein Feld, in dem sich die deutsche Industrie und Forschung Alleinstellungsmerkmale erarbeiten kann, beziehungsweise muss. Die Kombination aus ingenieurwissenschaftlichem Know-how und KI kann eine der großen Chancen für den Standort Deutschland sein.

### 10.1.4 Weitere Einsatzgebiete der KI

In Kap. 2 wurden schon eine Reihe von Anwendungen der KI dargestellt, die menschliche Leistungen erreichen oder übertreffen. Mittlerweile sind maschinelles Lernen und TNN in viele weitere Anwendungsbereiche vorgedrungen. Diese werden in Abb. 10.11 in elf Anwendungsfeldern gegliedert. Neben den bereits vorgestellten Anwendungsfeldern Smart Home, Medizin und industrielle Fertigung, werden kurz noch die weiteren Felder erläutert.



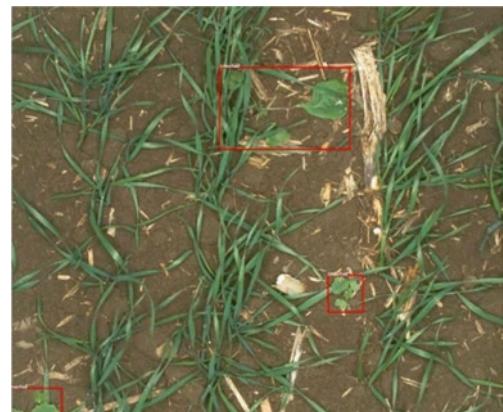
**Abb. 10.11** Ein Überblick über Anwendungen der Künstlichen Intelligenz, gegliedert nach Anwendungsfeldern

Im Anwendungsfeld Medien hat sich schon in den letzten Jahren eine weitreiche Änderung hin von analogen zu digitalen Vertriebswegen durchgesetzt. Im Bereich der Printmedien werden jetzt schon viele Routinemeldungen im Sport- und Finanzbereich von KI-Systemen generiert (Abschn. 9.2.1). Mit der Entwicklung von guten Sprachmodellen wird es jetzt möglich, automatisch konsistente Geschichten zu verfassen (Abschn. 9.2.3). Gibt man beispielsweise dem Programm „GPT3“ eine Überschrift, kann es in Zukunft ganz automatisch einen Artikel verfassen. Auch bei anderen Medien, wie Video und Gaming, wird KI wesentliche Änderungen bewirken. Es werden zum Beispiel Techniken der Bildverarbeitung (Kap. 5) und des Bestärkungslernens (Kap. 8) genutzt, um die Spiele-Welten realistischer und die automatischen Gegenspieler schlauer und attraktiver zu machen. Es ist ebenfalls abzusehen, dass die Charaktere in Videospielen realistische Sprachdialoge mit dem Spieler führen und intelligenteren Spielstrategien verfolgen können.

Ein sehr wichtiges Anwendungsfeld der KI ist die Landwirtschaft. Fortschritte im maschinellen Lernen und der Robotik haben in der letzten Zeit zur Entwicklung sensibler und geschickter Roboter geführt, welche zielgenau einzelne Früchte, Pflanzenteile und Ungeziefer identifizieren und greifen können. Dabei werden der Reifegrad der Pflanze, ihre möglichen Krankheiten und Schädlinge erkannt (Lewis 2019). Anschließend können Schädlinge und Unkräuter entfernt (Abb. 10.12), die Pflanze gezielt gedüngt und die Früchte schonend geerntet werden. Düngemittel und Pestizide müssen nicht mehr flächendeckend ausgebracht werden, sondern können individuell auf der jeweiligen Pflanze platziert werden. Dies ermöglicht potentiell einen Schub für nachhaltige und biologische Landwirtschaft. Eine Herausforderung dabei ist die oft rauhe und schmutzige Umgebung, welche sehr stabile Geräte und Sensoren erfordert. Auch bei der Ernte kann KI angewendet werden, z. B. beim zuverlässigen automatischen Aussortieren beschädigter Blaubeeren oder Tomaten. Hierdurch kann die Qualität der gelieferten Produkte entscheidend verbessert werden (Severson 2020).

Im Anwendungsfeld der Kommunikation spielen Techniken aus dem Bereich der Bedeutungserkennung von Text (Kap. 6), der Spracherkennung (Abschn. 7.3) und der Sprachassistenten (Abschn. 7.6) eine wesentliche Rolle. Hunderte Millionen Menschen nutzen

**Abb. 10.12** Bilderkennung der Nutzpflanze (Winterweizen) und des Unkrauts (rot markiert). Es gibt ein hohes Maß an Überdeckung des Unkrauts. Bildnachweis im Anhang (Dyrmann et al. 2017)



täglich Facebook Messenger, WhatsApp, Signal und andere Instant-Messaging-Dienste. Weitere Millionen kommunizieren mit sprachbasierten Assistenten wie etwa Alexa oder Google Home. Im Gegensatz zu klassischen Medien unterstützen sie die Kommunikation im privaten Kreis oder aber den auf die eigene Person zugeschnittenen Austausch mit einem Chatbot (Abschn. 7.6), der möglicherweise eine empathische Beziehung aufbaut (Abschn. 9.4). Gesprochene Sprache ist der effizienteste Kommunikationskanal für den Menschen, und es ist zu erwarten, dass er die textbasierte Kommunikation in Zukunft zurückdrängen wird, insbesondere in Entwicklungsländern mit Defiziten in der Alphabetisierung.

Ein wirtschaftlich wichtiges Anwendungsfeld der KI ist die Finanzbranche. Schon seit längerem werden maschinelle Lernverfahren verwendet, um aus den aktuellen Kursen und Wirtschaftsstatistiken die zukünftigen Kurse von Aktien und Finanzderivaten abzuschätzen. In normalen Zeiten sind diese Prognosen recht zuverlässig, bei der Finanzkrise führten sie aber zu gravierenden Fehlentscheidungen, weil die Modelle auf ungerechtfertigten Annahmen basierten und außerhalb ihrer Trainingsdaten eingesetzt wurden. Als zusätzliche Information lassen sich beispielsweise Stimmungen und Meinungen zu Firmen, Marken und politischen Themen aus Nachrichtentexten extrahieren (Sentimentanalyse, Abschn. 6.8.1) und für wirtschaftliche Prognosen nutzen. Eine wichtige Anwendung ist auch die Beurteilung von Versicherungsrisiken und -schäden Abb. 10.13, z. B. durch Interpretation von Bildern (Abschn. 5.5) oder Texten (Abschn. 6.8). Schon länger bekannt sind die Beurteilung der Kreditwürdigkeit von Personen (credit scoring), der Betrugserkennung bei Kreditkartentransaktionen sowie der Erkennung von Geldwäsche. Die Bewertung von Versicherungsrisiken kann zum Beispiel durch Sensoren in Autos und Maschinen verbessert werden. So können Autofahrer mit vorsichtigem Fahrstil eine Prämienreduktion erhalten. Für den Kundenkontakt können Sprachassistenten und Chatbots genutzt werden, welche dann einfache Dienstleistungen erbringen. Insgesamt ist abzusehen, dass sich KI-Systeme in weiten Bereichen der Finanzwirtschaft einsetzen lassen.



**Abb. 10.13** Ein Mitarbeiter der Schadenserstattung macht Notizen an einem durch einen Tornado beschädigten Haus. Schadensbeurteilungen können in Zukunft ganz oder teilweise KI-gestützt erfolgen. Bildnachweis im Anhang



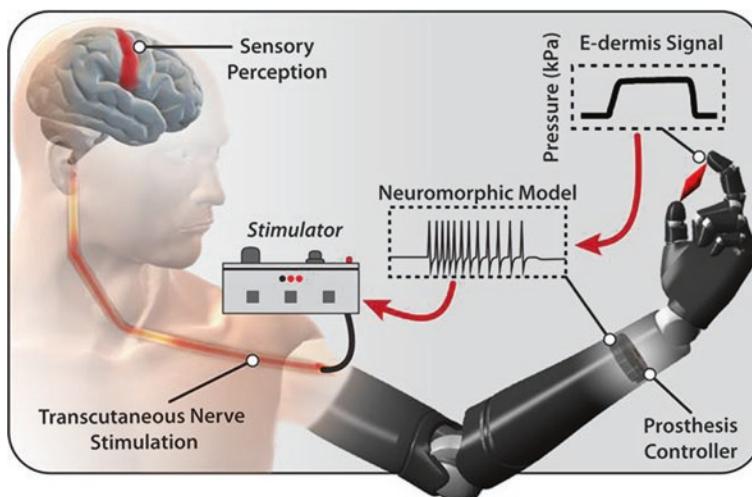
**Abb. 10.14** Selbstfahrendes Auto von Waymo. Auf dem Dach sind spezielle Sensoren angebracht. Bildnachweis im Anhang

Zentraler Punkt des Anwendungsfeldes Verkehr ist das selbstfahrende Auto (Abschn. 8.5). Die Technik ist noch in der Erprobung (Abb. 10.14). Wenn sie denn in einigen Jahren eingeführt wird, sind massive Änderungen im Verkehr (z. B. preisgünstige autonome Taxis), weniger Parkraum für Privatautos und stark reduzierte Unfallzahlen zu erwarten (Abschn. 8.5.3). Parallel wird KI für die Verkehrslenkung eingesetzt werden, wobei die Informationen von einer Vielzahl von hochauflösenden Videokameras und sonstigen Sensoren benutzt werden. Damit können flexible, verkehrsabhängige Mautgebühren eingeführt werden, welche die Verkehrsbelastung der Innenstädte und die Abgase reduzieren können.

Das Internet selbst ist ein bedeutendes Anwendungsfeld für KI. Mittlerweile sind digitale Plattformen verpflichtet, Filteralgorithmen zu entwickeln, um unangemessene Sprache, Hasskommentare, Kinderpornographie und andere Arten von Inhalten herauszufiltern. Diese Algorithmen basieren auf Bildklassifikation (Abschn. 5.5) und der semantischen Interpretation von Sprache (Abschn. 6.5) und müssen eine delikate Balance zwischen Meinungsfreiheit und der Verhinderung strafbarer Handlungen gewährleisten. Zentrale Triebfedern des Internets sind personalisierte Empfehlungen von Produkten und neuen Webseiten für die Nutzer. Je genauer diese Empfehlungen auf die persönlichen Bedürfnisse des Nutzers zugeschnitten sind, desto höher ist der Umsatz der Anbieter. Beispielsweise wird geschätzt, dass 35 % des Umsatzes von Amazon durch sein KI-System für Empfehlungen induziert werden. Das Ziel hierbei ist „Upselling“, d. h. der Kunde kauft zusätzliche Produkte, nachdem er seine eigentlichen Einkäufe getätigkt hat. Eine weitere boomende Anwendung ist das sogenannte „dynamic pricing“, welches den Preis abhängig vom Kunden und dessen prognostizierter Kaufmotivation festsetzt. Ein Problem entsteht bei der personalisierten Filterung von Nachrichten, wenn ein Nutzer nur noch Meldungen erhält, welche seine Weltsicht und Vorurteile unterstützen. Es gibt eine weite Diskussion darüber, wie solche Filterblasen vermieden werden können.

Ein noch kleines Anwendungsfeld der KI ist die Bionik, also die Verbindung von Biologie und Technik. Dazu ist es nötig Hirnsignale in Kommunikation zu überführen, die von der Maschine verstanden wird. So lässt sich beispielsweise eine Prothese über ein Nerveninterface vom Träger steuern (Bryant 2019). Die Armprothese hat 26 Gelenke sowie Druck- und Zugsensoren in jeder Fingerspitze. Die Kommunikation funktioniert aber auch umgekehrt (Abb. 10.15): So erhält der Mensch über Sensoren an der Prothese Informationen zu Temperaturen und Vibrationen (Osborn 2018). Auf ähnliche Weise funktioniert eine Beinprothese, bei welcher der Nutzer die Beugebewegungen, den Druck und Vibrationen fühlen kann (Scudellari 2019). Eine große Rolle spielen die Erkenntnisse der Bionik mittlerweile im Leistungssport. Mit Hilfe von mehreren Kameras und anderen Sensoren werden die Bewegungen der Sportler vermessen und Änderungen zur Leistungssteigerung vorgeschlagen. Beispielsweise kann man so beim Fußballspiel Spieler identifizieren, die besonders gut Eckbälle oder Freistöße schlagen können. Dies geht dann in die Transferbewertung der Spieler ein (Smith 2020b).

Das letzte Anwendungsfeld der KI ist die Kunst. In Kap. 9 wurde schon beschrieben, dass KI in einigen Bereichen kreative Leistungen erbringen kann. Dabei wird erörtert, ob dies „echte“ kreative Erfindungen sind oder nur Variationen schon existierender Werke. In Marks (2019) wird die Ansicht vertreten, dass der Mensch gut sei bei der Auswahl zwischen einer Reihe von Alternativen. Hingegen habe er Probleme, eine große Anzahl von Variationen zu produzieren, aus denen er dann auswählen kann. KI-Systeme seien gut in der Produktion vieler möglicher unterschiedlicher Entwürfe und können so die Künstler, Designer, Musiker und Autoren in ihrer Arbeit unterstützen. Kunst soll originell sein, aber auch nicht zu weit von gewohnten Formen abweichen. Man kann zum Beispiel neue Ge-



**Abb. 10.15** Eine Armprothese kann durch Sensoren dem Träger ein Feedback geben. Darüber hinaus kann taktile Information aus den Fingern an das Gehirn übermittelt werden und dort Berührung und Schmerz signalisieren. Bildnachweis im Anhang (Osborn et al. 2018)

mälde automatisch so generieren, dass sie allgemein als Kunst wahrgenommen werden, aber keiner bisherigen Stilrichtung eindeutig zugeordnet werden können (Abschn. 9.1.8). Derartige Kunstwerke konnten von menschlichen Betrachtern in Hinblick auf ihre künstlerische Aussagekraft nicht von Bildern unterschieden werden, die auf aktuellen Kunstmessen angeboten werden. Insgesamt sehen die Autoren auch, dass die KI in Zukunft eine unabhängige Rolle in der Kunst übernehmen und eigenständige Beiträge produzieren könnte.

---

## 10.2 Wirtschaftliche Auswirkungen und Zusammenhänge

KI hat sich inzwischen als ein zentrales Trendthema der globalen Technologieindustrie etabliert. In den kommenden Jahren wird die KI enorme Auswirkungen auf die wirtschaftliche Entwicklung und unsere Arbeitswelt haben. Und sie wird auch die Wettbewerbsdynamik in vielen Branchen grundlegend verändern. In den kommenden Abschnitten gehen wir auf die wichtigsten Faktoren dieser Entwicklung ein, schauen uns an, welche Trends durch die großen Technologieunternehmen gesetzt werden und diskutieren die Auswirkungen auf den Arbeitsmarkt.

### 10.2.1 Die Monetarisierung von Daten

Amerikanische Internetkonzerne waren vor einigen Jahren die ersten, die mit Big-Data-Technologien auf hochleistungsfähigen Hard- und Softwareplattformen immense Datens Mengen gesammelt haben. In der Wertschöpfung aus Daten sind die US-Internetkonzerne – gefolgt von chinesischen Unternehmen – weiterhin führend. Daten sind für sie zu einem Produktions- und Wettbewerbsfaktor und somit zu einem zentralen Wirtschaftsgut geworden.

Viele Unternehmen haben aufgehört, als Google 2014 die Firma Nest für die damals unvorstellbare Summe von 3,2 Milliarden Dollar gekauft hat (Miners 2014). Nest hatte mit seinen außergewöhnlich schicken smarten Thermostaten (Abb. 10.16) etwas Besonderes zu bieten: den Zugang zum Smart Home und den gesammelten Daten aus Millionen von amerikanischen Haushalten. Das intelligente Nest-Thermostat erfasst eine Vielzahl von Informationen aus dem Haushalt: wann wird die Temperatur nach oben oder unten geregelt, wie hoch ist die Luftfeuchtigkeit, wie ist die Helligkeit im Raum etc. Damit erstellt Nest Prognosen, z. B. wann die Bewohner zu Hause sind. Das Thermostat konnte direkt über eine App gesteuert werden. Der Kaufpreis von Nest hatte damals große Aufmerksamkeit erzeugt, da der Unternehmenswert traditionell nach drei wesentlichen Aspekten eingeschätzt wird (Abb. 10.17): der Qualifikation der Mitarbeiter, dem verfügbaren Kapital und dem Wert der materiellen Güter (Wrobel und Hecker 2019).

Bezogen auf Nest mit seinen damals 280 Mitarbeitern und einem geringen Kapital- und Materialwert muss man feststellen, dass die gesammelten Thermostatdaten eine wichtige

**Abb. 10.16** Intelligenter Thermostat von Nest.  
Bildnachweis im Anhang

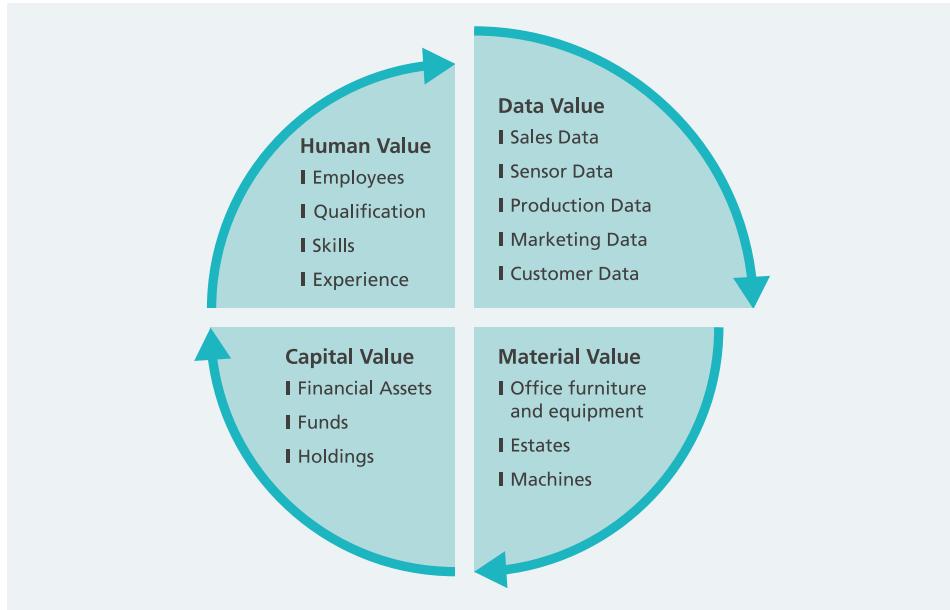


Rolle bei der Unternehmensbewertung und dem daraus resultierenden Kaufpreis gespielt haben müssen. Die Erkenntnis aus dieser und vielen anderen Übernahmen ist, dass Daten als immaterieller Vermögenswert in Zukunft eine bilanzierbare Komponente für Unternehmen darstellen müssten (Abb. 10.17). Dies ist ein wichtiger Aspekt für die zukünftige strategische Ausrichtung von Unternehmen, welcher die Wahrnehmung von Daten geändert hat.

Darum ist es für Unternehmen lohnenswert, Daten zu sammeln oder einzukaufen, zu kurieren, zu schützen, intelligent zu verknüpfen und auszuwerten. Ein modernes Auto zum Beispiel ist nicht nur ein Fortbewegungsmittel, sondern auch eine mobile Wetterstation. Über serienmäßig integrierte Thermometer, Regensensoren für die Aktivierung der Scheibenwischer, Helligkeitssensoren, Luftfeuchtigkeitsmesser und GPS können wichtige Daten erfasst und über das Mobilfunknetz übertragen werden. Bei Aggregation über viele Fahrzeuge und Abgleich mit Satellitenbildern ergibt sich ohne substantielle Investition in Infrastruktur ein lokal sehr fein aufgelöstes Bild des aktuellen Wetters, ein Spektrum an Daten, das nicht zu unterschätzen ist. Zum Beispiel für die smarte Landwirtschaft: mit diesen Informationen kann präziser gedüngt und die Erntezeit lokal besser bestimmt werden.

Zu jeder KI-Strategie eines Unternehmens gehört also auch eine Datenstrategie. Das heißt vor allem, eine Datenlandkarte des eigenen Unternehmens aufzustellen und eine Datenausbaustrategie zu formulieren. Ein Unternehmen sollte wissen, welche Daten ein Alleinstellungsmerkmal sind, in welcher Menge sie vorliegen, wie sie verknüpft und annotiert sind, und wie man Umfang und Qualität verbessern kann.

Amerikanische Konzerne zeigen, wie subtil man beim Datensammeln vorgehen kann. Viele von uns werden schon einmal einen Check wie in Abb. 10.18 durchgeführt haben. Er stammt von der Alphabet-Tochter Captcha und prüft, ob man ein Bot oder ein Mensch ist. Captcha wird Webseitenanbietern als freier Dienst zur Verfügung gestellt, mit dem sehr geschickten Ziel, annotierte Bilder für das maschinelle Lernen zu erhalten. Jeden Tag ent-



**Abb. 10.17** Neben Mitarbeitern, Kapital und Material, den klassischen Säulen des Unternehmenswertes, spielt die Verfügbarkeit von Daten bei der Bewertung eines Unternehmens eine immer größere Rolle

stehen durch die manuelle Klassifikation von Straßenschildern, Gebäuden etc. Millionen von Trainingsdaten für das autonome Fahren.

Nach den Datenmengen aus Internet, sozialen Medien, E-Commerce und vielen weiteren Apps auf unseren Smartphones sind es in den kommenden Jahren die vernetzten Geräte und Anlagen, die die Datenflut in die Höhe treiben werden. Abb. 10.19 zeigt eine Schätzung des Datenzuwachses (Kanellos 2016). Inzwischen rechnet man mit 30 Milliarden vernetzter Geräte im Jahr 2020 und 80 Milliarden im Jahr 2025.

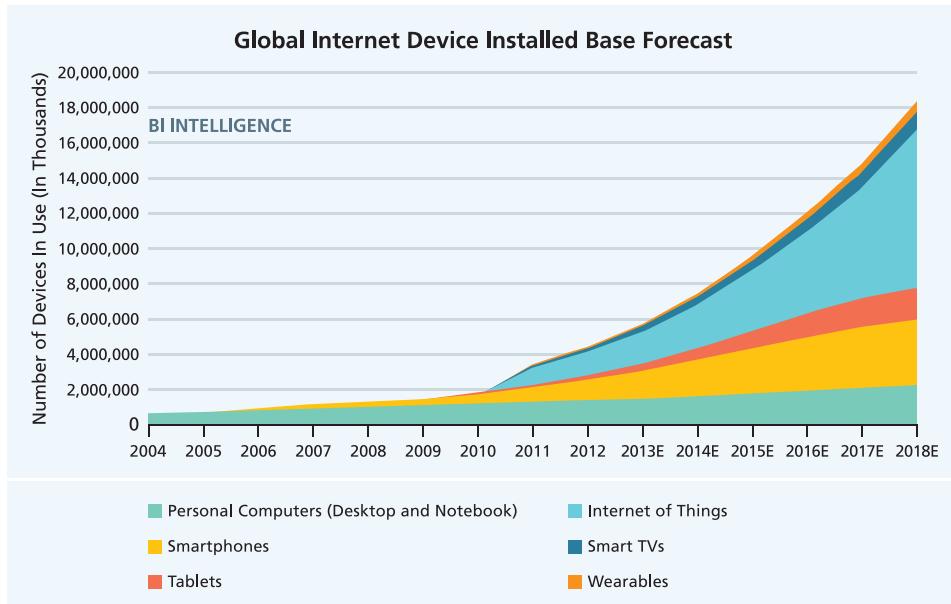
Zum Internet der Dinge gehören einerseits im Privatbereich smarte Geräte, wie verschiedene Wearables (Smart Watch, Health- und Fitnesstracker) und die intelligente Gebäudesteuerung (Smart Home, Smart House), und andererseits ganzheitliche Lösungen wie die intelligente Ampelschaltung, das intelligente Stromnetz (Smart Grid) und das vorausschauende Verkehrsmanagement in der intelligenten Stadt (Smart City). Ein enormes Potenzial für die Künstliche Intelligenz bietet das Internet der Dinge auch im industriellen Umfeld – eine der großen Chancen für Deutschland und Europa.

Dabei sollte Europa auch seinen eigenen Weg in der Datenspeicherung gehen. Google, Amazon und viele weitere Unternehmen bilden Datenmonopole, bei denen die Daten zentral beim Anbieter gespeichert werden. Diese zentralisierte Datenmacht sollte kein Vorbild für Europa sein. Die Idee ist, eine dezentrale Dateninfrastruktur aufzubauen, bei der nur anwendungs- und zweckgebunden Daten über Schnittstellen zwischen Unternehmen und Personen ausgetauscht werden. Dies hat den Vorteil, dass keine



**Abb. 10.18** Generierung von Trainingsdaten am Beispiel eines Bild-Captchas (Wang et al. 2018). Die Autoren weisen darauf hin, dass mit der verbesserten Bildklassifikation Bild-Captchas in Kürze automatisch lösbar sein werden und schlagen Captcha-Aufgaben mit visuellem Schließen vor. Bildnachweis im Anhang

Abhängigkeiten entstehen und nur unter Einwilligung der Nutzer Daten über einen Datenbroker, evtl. gegen Bezahlung, ausgetauscht werden. Man bleibt als Privatperson oder Unternehmen souveräner Eigner seiner Daten und kann die Prozesse der Verarbeitung nachvollziehen. Darüber hinaus erzeugen offene Schnittstellen zu frei verfügbaren öffentlichen Daten einen Datenraum, der vielfältige Möglichkeiten für Start-ups und branchenübergreifende neuen Geschäftsmodelle erzeugt. Als Beispiel kann hier der Medical Data Space herangezogen werden. Dieser verarbeitet seine sensiblen Patientendaten dezentral und nicht in einem deutschlandweiten Zentralregister. Die Daten bleiben also vor Ort. Weitergeleitet werden nur anonymisierte Modellergebnisse, die dann in der Summe ein aussagekräftigeres Bild über die Muster von Krankheiten, Therapieerfolgen usw. liefern.



**Abb. 10.19** Prognostiziertes Datenwachstum in den kommenden Jahren. Eigene Grafik (Kanellos 2016)

### 10.2.2 Die neue digitale Servicewelt – KI as a service

„Software eats the World“. Marc Andreessen, Erfinder von Netscape, 2011

KI ist ein zentraler Treiber der Digitalisierung in allen Wirtschafts- und Lebensbereichen. Vom anfänglichen Hype ist KI zu einem ständigen Begleiter geworden. Es entsteht eine Vielzahl an neuen Services und Angeboten. Digitale Produkte werden zukünftig ohne KI-Komponenten kaum noch anzutreffen sein. Mit KI können sich bestehende Wertschöpfungsketten fundamental ändern. Dabei sind drei ökonomische Modelle relevant:

- Platform Economy:** Digitale Plattformen sind Intermediäre, die Anbieter und Kunden über eine Plattform vernetzen und eine Interaktion über Apps, Skills und Suchfunktionen ermöglichen. Plattformen haben sich zu dem dominierenden Geschäftsmodell der digitalen Wirtschaft entwickelt. Die weltweit umsatzstärksten Unternehmen haben eine oder mehrere Plattformen als Kernbestandteil ihres Produkt- und Dienstleistungsportfolios und entwickeln in ihrer Anwendung Monopolisierungstendenzen. So zählen die Handelsplattform Amazon (Abb. 10.20), der Google Play Store und die Kommunikationsplattform WhatsApp zu den dominierenden Anbietern.
- Economy of Speed:** Unternehmen und Start-ups streben eine First-to-Market-Strategie an, indem sie versuchen, frühzeitig und oft mit hohen finanziellen Risiken in den Markt einzutreten, um eine Monopolstellung zu erlangen. Dazu werden in der Organisation



**Abb. 10.20** Amazon hat eine der erfolgreichsten digitalen Plattformen aufgebaut. Sie umfasst Handel mit Konsumgütern, Streamingangebote, digitale Assistenten, Cloud-Computing, etc. Bildnachweis im Anhang

der Unternehmen Infrastrukturen, Strukturen und Prozesse ganz strategisch aufgebaut, um Produkte in kürzester Zeit zu entwickeln.

3. **Economy of Scale:** Eine weitere Wettbewerbsstrategie ist es, durch Optimierung der Produktionskosten Vorteile zu erreichen. Ziel ist, die Kostenführerschaft zu übernehmen und so Wettbewerbern den Markteintritt zu erschweren oder ihnen Marktanteile zu nehmen. Effizienzgewinne werden in der Automatisierung der Produktion erzielt, indem bessere Prognosen, bessere Entscheidungen und schnellere Aktionen die Kosten pro Einheit immer weiter senken. Zudem sind die Investitionskosten bei einer Verdopplung der Nutzerzahlen einer Plattform relativ gering und versprechen hohe Gewinne bei einem entsprechenden Marktanteil.

Unternehmen richten sich zunehmend an den skizzierten Modellen aus und fordern ein höheres Maß an Innovationsgeschwindigkeit und Skalierbarkeit ihrer Lösungen und Services. Dabei ist es für viele eine immense Herausforderung, ihre bestehenden Strukturen anzupassen und eine sogenannte digitale DNA, eine digitale Mentalität, zu entwickeln (Hecker et al. 2016). Sicherlich sehr entscheidend für den Erfolg ist eine Unternehmenskultur, in der Ausprobieren, auch ein Scheitern und natürlich ein Erfolg anerkannt werden. Ein angstfreies Denken jenseits der bisherigen Grenzen, jenseits der eigenen Abteilungsstrukturen muss möglich sein. Gerade bei interdisziplinärer, abteilungs- und unternehmensübergreifender Zusammenarbeit entsteht Innovation. Unternehmen müssen dafür die richtigen Rahmenbedingungen schaffen. Zu diesen Rahmenbedingungen gehört auch KI-Kompetenz. Vom Top-Management bis zum einzelnen Mitarbeitenden sollte verstanden werden, was KI kann und wo die Grenzen liegen. Wenn eine solche Kultur etabliert ist, dann liefert KI die Instrumente für Innovation, Effektivität, Geschwindigkeit und Skalierbarkeit.

KI-Software wird immer häufiger „as a Service (AIaaS)“ zur Verfügung gestellt. Die Software wird nicht zur Installation und Betrieb im Unternehmen angeboten, sondern als Dienstleistung über das Internet bzw. eine Plattform. Der Anwender ist nicht Besitzer einer Softwareversion, sondern lediglich Abonnent von Diensten auf Basis eines bestimmten Abrechnungsmodells. Er spart sich den Aufbau der IT-Infrastruktur, die Pflege der Software und die notwendige IT-Fachkompetenz, und ist dadurch evtl. schneller und kostensparsamer mit seinen eigenen Produkten am Markt. Selbst ein plötzliches Nutzerwachstum ist in der Regel ohne große Hürden zu bewältigen, da der Serviceanbieter nur weitere Rechenkapazität über die Cloud dazuschalten muss. Der Serviceanbieter hat weitere Vorteile. Er kann Marktanforderungen und Kundenwünsche schneller umsetzen, indem er die Services ergänzt. Er hat direkten Zugriff auf die notwendigen Nutzerdaten – oftmals auch darüber hinaus – und kann dadurch seinen Service laufend verbessern. Amerikanische Unternehmen wie Airbnb, Uber oder Amazon zeigen, wie man sich mit einer Serviceplattform und ohne sonstige Investitionsgüter als Vermittler etablieren kann.

Im Jahr 2011 verkündete Marc Andreessen im Wall Street Journal sein berühmtes Zitat „Software is eating the world“ (Andreessen 2011). Das prognostizierte, dass die Weltwirtschaft bald „vollständig digital verdrahtet“ sein würde. Dabei ist Software das Herzstück einer Verbesserung in drei Dimensionen: reduzierte Kosten, erweiterter Marktzugang und verkürzte Entwicklungszyklen. Wenig Signifikantes passiert heute noch, ohne dass Software dabei eine kritische, oft sogar die ausschlaggebende Rolle spielt. Mit dem Zitat von Andreessen geht die Idee einher, dass alles was sich digitalisieren lässt, auch digitalisiert werden wird.

KI kann durch die Digitalisierung an sehr vielen Stellen eingesetzt werden. Der Fantasie sind eigentlich keine Grenzen gesetzt. Arbeit, die repetitiv und deshalb leicht ermüdend ist, wird in absehbarer Zeit komplett von Maschinen geleistet werden können, wie zum Beispiel das automatische Ausfüllen von Formularen, das Überprüfen ausgefüllter Formulare und das Abarbeiten korrekt ausgefüllter Formulare. KI wird viele Routineaktivitäten übernehmen, entweder als Assistent oder völlig autonom. Unternehmen, die bereits jetzt zu solchen Tätigkeiten Daten sammeln, haben große Vorteile gegenüber denjenigen, die das nicht tun.

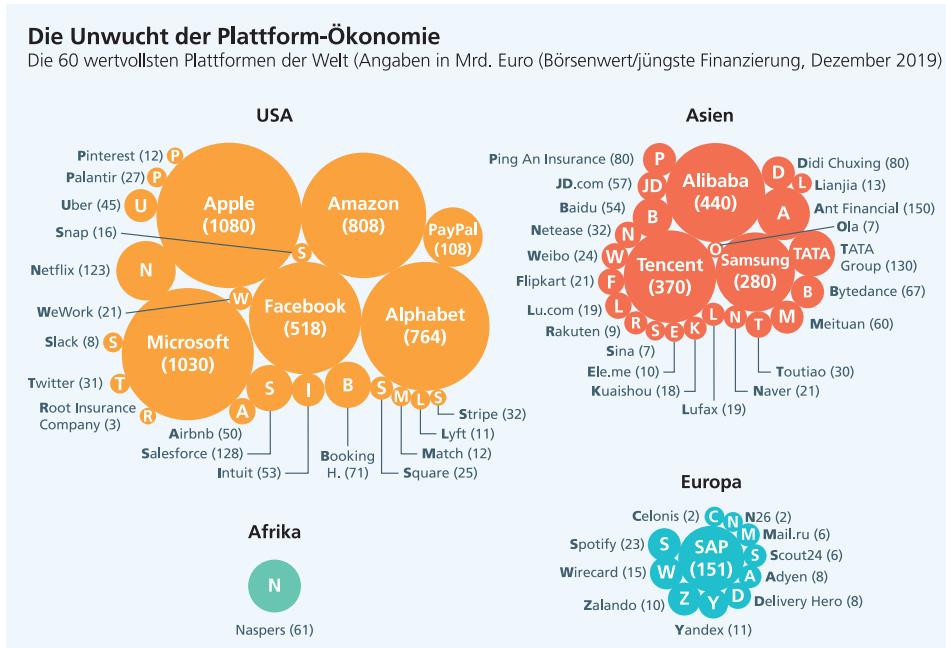
Künstliche Intelligenz ist die logische Fortsetzung der Digitalisierung, denn erst KI ermöglicht es uns, aus Daten zu lernen und daraus zusätzlichen Wert für Unternehmen und Gesellschaft zu schaffen. Durch lernende Systeme können bisherige Aufgaben besser, schneller und günstiger gelöst werden – dies schafft einen Effizienz-Gewinn im Wettbewerb. Auch im privaten Umfeld profitieren wir von KI. Dienste wie Netflix, Amazon, der autonome Staubsaugroboter, die Fremdsprachenübersetzung, usw. machen die Entwicklung der KI für uns zu einem permanenten Wegbegleiter und nützlichen Assistenten im Alltag. Darüber hinaus ist KI zu einem unverzichtbaren Forschungsinstrument für fast alle Fachbereiche geworden. Mit großer Experimentierfreude nutzen Wissenschaftlerinnen und Wissenschaftler auch jenseits der Informatik und Mathematik die Werkzeuge zur Analyse großer Datenmengen. Es ist fast unmöglich, die vielen Anwen-

dungsfelder und Möglichkeiten aufzuzählen. Für uns ist aber bereits jetzt spürbar, dass die Mensch-Maschine-Interaktion sich maßgeblich verändern wird.

### 10.2.3 Große Unternehmen als Treiber der KI

„From mobile first to AI-first“. Sundar Pichai, Google CEO, 2016

Anders als in den USA und China verfügen deutsche Unternehmen nicht über die Massendaten aus sozialen Medien, Smartphones oder neuerdings smarten Lautsprechern. Häufig wird im Zusammenhang mit Künstlicher Intelligenz von den sogenannten „GAFA“ gesprochen. GAFA sind die vier großen Tech-Unternehmen Google, Amazon, Facebook und Apple. Vergleicht man in Abb. 10.21 anhand der Kreise den Börsenwert der Unternehmen im Jahre 2018, so erkennt man eine klare Dominanz der Plattformunternehmen aus den USA und China. Nur die deutsche SAP hat im weltweiten Wettbewerb eine signifikante Größe. Da über die Plattformen aber ein Großteil des weltweiten Datentransfers stattfindet, entsteht für Europa ein Ungleichgewicht, insbesondere im direkten Kundenkontakt (Business to Consumer, B2C). Es besteht die Gefahr einer Monopolbildung durch diese Unternehmen, die durch ihre Datenkontrolle und globale Wettbewerbsfähigkeit die digitale Souveränität von Europa bedrohen.



**Abb. 10.21** Bei den größten 100 Plattformunternehmen der Welt haben Amerika und der Raum Asien-Pazifik gegenüber Europa klar die Nase vorn. Eigene Grafik (Hossain et al. 2018)

## Der KI-Konzern Google

Schaut man sich stellvertretend für die GAFA die Entwicklung von Google bzw. Alphabet in den vergangenen Jahren an, dann hat das Unternehmen inzwischen nur noch wenig mit einem reinen Suchmaschinenanbieter zu tun (Abb. 10.22). Von der Heimautomatisierung mit Google Nest über das autonome Fahren mit Waymo bis zur digitalen Hochleistungsinfrastruktur mit Google Fiber: Google hat sich als KI-Unternehmen neu positioniert und ist quasi in alle relevanten Zukunftsthemen eingestiegen.

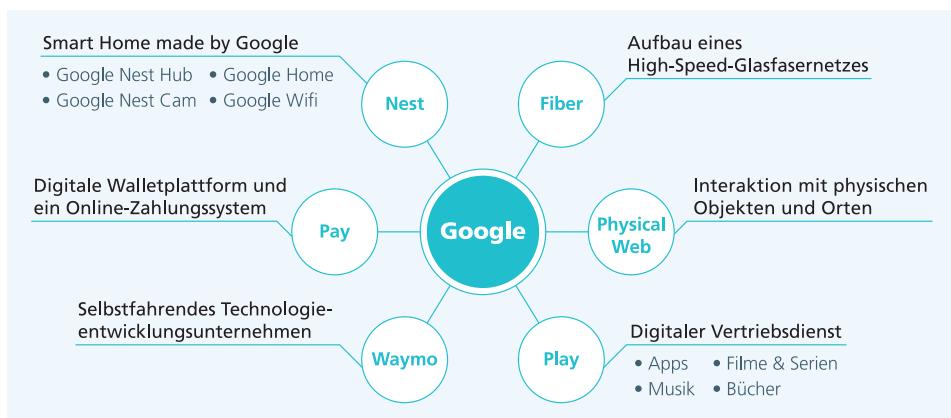
Google (bzw. seine Holding Alphabet) bietet auch heute noch seine Suchplattform an und lockt mit weiteren kostenlosen Diensten wie Google Maps zur Navigation, Gmail als Emailprovider, Google Translate zur Übersetzung oder YouTube als Plattform für Video-clips. Egal durch welche Interaktionen, Google lernt uns Nutzer immer genauer kennen und kann Werbung immer präziser schalten. Für Google sind wir Nutzer das Produkt, das es an seine Kunden, die Werbetreibenden, verkauft.

Durch seine vielen Expansionen und gelegentlichen Übernahmen erstickt Google viele Entwicklungen unabhängiger Service-Anbieter im Keim, was für eine offene Marktwirtschaft ein Problem wird. Europa braucht dringend eigene Plattformen als Alternativen, die den Kunden nutzen und gleichzeitig dem Datenschutz gerecht werden.

Und schließlich können die Tech-Giganten ihren KI-Experten exorbitante Gehälter zahlen. Sie treten ganz gezielt mit Headhuntern auf den zentralen Machine-Learning-Konferenzen auf. Wer gute Ideen hat und eine Publikation mit hohem Innovationsgrad schreibt, der bleibt diesen Unternehmen nicht verborgen.

## Die chinesische Konkurrenz

Inzwischen gibt es fast zu jeder amerikanischen Plattform auch ein chinesisches Pendant, das in Europa allerdings noch weniger bekannt ist. Es sind die Firmen Baidu, Alibaba und Tencent, abgekürzt: BAT (Abb. 10.23).



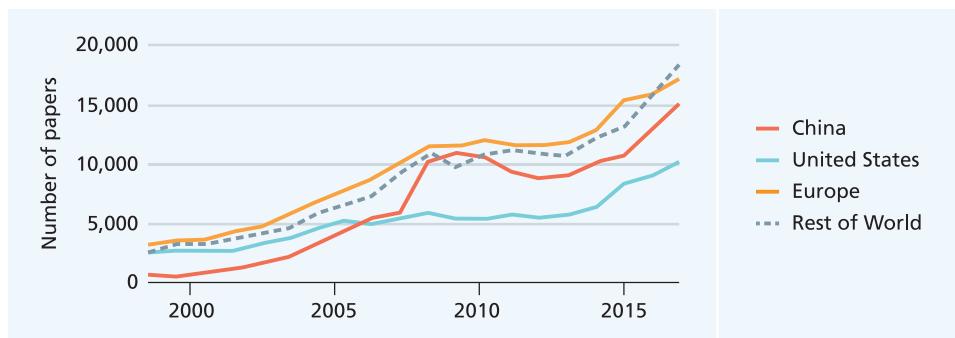
**Abb. 10.22** Google hat sich vom Suchmaschinenanbieter zum globalen IT-Unternehmen entwickelt

**Abb. 10.23** Amerikanische IT-Unternehmen und ihre chinesischen Pendants



Die Entwicklungsgeschwindigkeit der chinesischen Unternehmen ist bemerkenswert. Angeführt von Unternehmeridolen wie Alibaba Gründer Jack Ma, nutzen sie ihre unternehmerischen Freiräume. Der einst als chinesischer Klon von Amazon und Ebay gegründete Megakonzern Alibaba investiert Milliardenbeträge in eine KI-Forschungsoffensive, auch in Zusammenarbeit mit Spitzenforschern an ausgesuchten US-Eliteuniversitäten. IT-Konzerne wie Baidu und Tencent unterhalten im Silicon Valley Forschungszentren für KI. Den Tech-Giganten Baidu, Alibaba und Tencent ist China längst nicht mehr groß genug, und weltweite Angebote und Investitionen werden vorbereitet. Die chinesische Regierung hat es geschafft, ohne einen bisher festzustellenden politischen Kontrollverlust, Instrumente für eine unternehmerische Kreativität und Risikobereitschaft zu schaffen. Insbesondere die dynamische Start-up-Landschaft bringt immer wieder neue Unicorns hervor (Castro et al. 2019), die durch die Investitionsbereitschaft von in- und ausländischem Wagniskapital gute Voraussetzungen erhalten.

Daneben ist die hohe Bereitschaft für technologische Neuerungen in China einer der wichtigsten Faktoren für das rasante Wachstum. Zahlreiche Entwicklungen der digitalen Transformation stehen dabei im Widerspruch zu den humanistischen europäischen Prinzipien und Wertevorstellungen. Im Gegensatz zu den USA hatte China schon sehr früh eine nationale KI-Strategie. China möchte bis 2030 die weltweit führende KI-Nation sein. Abb. 10.24 zeigt den rasanten Anstieg von KI-Publikationen seit 2008, ein erster Hinweis auf den chinesischen Aufholprozess (Shoham et al. 2018). Die Aussichten für China sind gut: Die Masse der chinesischen Nutzer, der geringe Datenschutz, der stark geschützte inländische Markt und die staatliche Unterstützung haben hier globale Firmen entstehen lassen. China ist es so gelungen, in wenigen Jahren digitale Souveränität zu erlangen.



**Abb. 10.24** Vergleich der Anzahl der jährlich veröffentlichten KI-Publikationen in unterschiedlichen Regionen (Shoham et al. 2018)

### Chance für Europa

Auch wenn Europa im Consumer-Bereich (Business to Consumer, B2C) absehbar nicht aufschließen kann, so muss es wenigstens im industriellen Bereich (Business to Business, B2B) digitale Souveränität gewinnen. Dafür muss Europa KI-Technologien entwickeln, die sparsam mit Daten umgehen und auch mit wenigen Daten lernen können. Man spricht in diesem Zusammenhang von hybrider KI. Dabei werden Daten- und wissensgetriebene Methoden kombiniert. Während der datenbasierte Ansatz statistische Zusammenhänge analysiert und die Analysen mit Methoden des maschinellen Lernens automatisiert, nutzt die wissensbasierte Vorgehensweise von Menschen vorgegebenes Wissen und leitet daraus Schlussfolgerungen ab. Hier kommen die Fachleute ins Spiel, zum Beispiel Ärzte in der Medizin oder Ingenieure in der Produktion. Ihre Erfahrung und Spezialwissen können als Fakten, Regeln und physikalische Gesetzmäßigkeiten ausgedrückt und ins maschinelle Lernen integriert werden. Einen Überblick findet man in (Gromann et al. 2019).

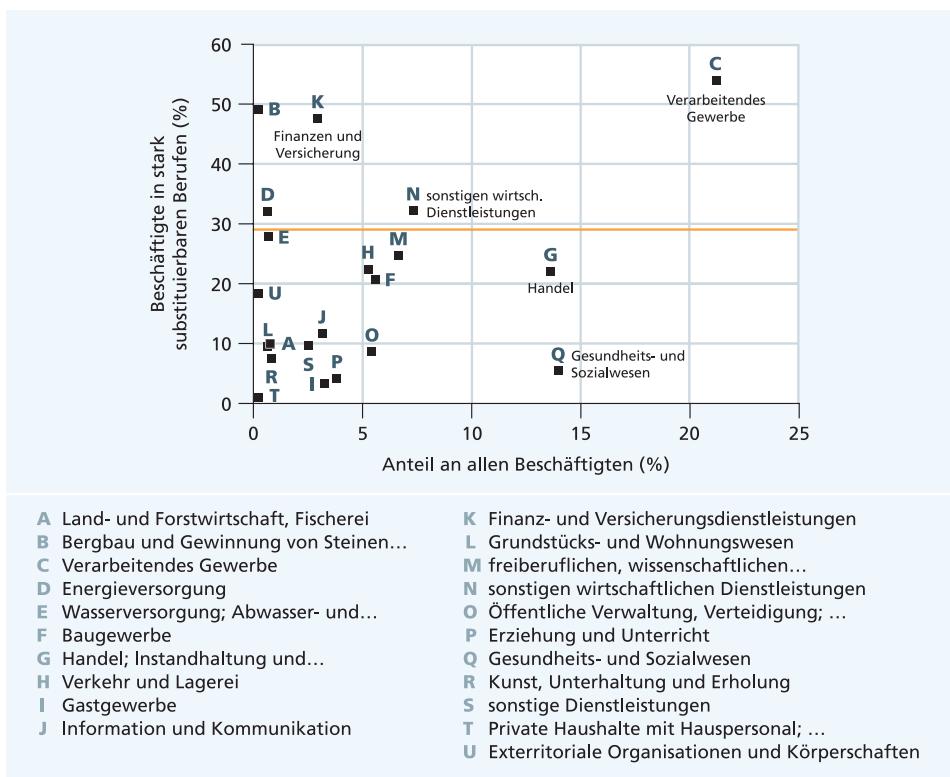
Die industrielle Analytik und maschinelles Lernen für intelligente Maschinen und Geräte bieten aber ein Feld, wo sich die deutsche Industrie und Forschung Alleinstellungsmerkmale erarbeiten kann und muss. An dieser Stelle Innovation ganz ohne ingenieurwissenschaftliches Know-how voranzutreiben wird nicht funktionieren. Dies ist eine der Chancen für den Standort Deutschland.

#### 10.2.4 Die Auswirkungen auf den Arbeitsmarkt

Konkrete Prognosen für den Arbeitsmarkt sind aufgrund der großen Dynamik des Themas sehr schwierig. Sicher kann man aber sagen, dass sich viele Jobs durch den Einfluss von KI verändern werden. Computerprogramme können rund um die Uhr und am Wochenende arbeiten, stellen keine Ansprüche und haben keine Gewerkschaft. Das macht sie für viele Tätigkeiten attraktiv. Ein auf die Diagnose von radiologischen Aufnahmen trainiertes

KI-System ist immer konzentriert und macht keine Flüchtigkeitsfehler. KI-Anwendungen erzielen eine gleichbleibende Qualität, bei der Menschen auf Dauer nicht mithalten können.

In Abb. 10.25 prognostiziert das Forschungsinstitut IAB der Bundesanstalt für Arbeit, in welchem Grad Berufe von der Digitalisierung betroffen sind (Kropp et al. 2018). Starke Veränderungen sind abzusehen für Beschäftigte des Verarbeitenden Gewerbes, des Bergbaus, der Finanz- und Versicherungsdienstleistungen, der Energieversorgung sowie bei sonstigen wirtschaftlichen Dienstleistungen. Da zum Beispiel das Verarbeitende Gewerbe etwa 22 % der Beschäftigten umfasst, sind allein in dieser Branche etwa 12 % aller deutschen Beschäftigten von der Digitalisierung betroffen. Auf der anderen Seite sind bei den personenorientierten Dienstleistungsbereichen, etwa im Gesundheits- und Sozialwesen, nur wenige Änderungen durch Digitalisierung zu erwarten.



**Abb. 10.25** Anteil der Beschäftigten in Berufen mit hohen Substituierbarkeitswerten in Deutschland nach Wirtschaftszweigen in %. Lesehilfe: Beschäftigte des Verarbeitenden Gewerbes (C: schwarzes Quadrat rechts oben) sind zu 55 % in stark substituierbaren Berufen beschäftigt. Sie bilden zugleich etwa 22 % der Beschäftigten in Deutschland. (Kropp et al. 2018)

Wir kennen es seit der Entwicklung der Dampfmaschine, dass insbesondere Menschen am Fließband oder bei schwerer körperlicher Tätigkeit vor der Gefahr stehen, ihren Job zu verlieren, weil Maschinen sie ersetzen. Akademiker fühlten sich über viele Jahre sicher, aber dies ist nun anders. Nach der Automatisierung von manuellen Arbeiten wird auch die gedankliche Arbeit ersetzt. Schließlich lässt sich mittlerweile, jedenfalls bei einfachen Routinearbeiten, das Verstehen, Entscheiden oder Handeln automatisieren. Als Wirtschaftsprüfer oder Jurist hat man gedacht, die Automatisierung beträfe einen nicht, weil ein Roboter diese hochwertigen Arbeiten nicht machen könne. Aber auch hier lassen sich mittlerweile sehr viele Prozesse automatisieren. Grundsätzlich wird KI und Digitalisierung viele Routinetätigkeiten übernehmen, entweder völlig autonom oder als Assistent. Somit werden Menschen von mühsamen manuellen oder ermüdenden Tätigkeiten entlastet und können mit Unterstützung von KI-Assistenten und Empfehlungssystemen anspruchsvollere Aufgaben übernehmen. Wir werden lernen müssen, mit Maschinen und digitalen Werkzeugen zu kooperieren. Qualifizierung sichert an dieser Stelle zukünftig Arbeitsplätze. Deutschland sollte eine Vorreiterrolle in der Umsetzung der Digitalisierung und KI einnehmen, womit fortgeschrittene KI-zentrierte Tätigkeiten entstehen und eine Nachfragesteigerung aus dem Ausland aufgrund des Wettbewerbsvorteils induziert wird. Schreiten dagegen andere Länder in der Umsetzung schneller voran, so würde Deutschland an internationaler Wettbewerbsfähigkeit einbüßen und sich die positiven Nachfrageeffekte ins Gegenteil umkehren (Kruppe et al. 2019, S. 16).

### Bildung für eine digitale Welt

Da sich die Berufswelt durch KI rasant verändert, muss auch die Bildung reformiert werden. Man kann nicht erwarten, zukünftig sein ganzes Leben lang im gleichen Job bleiben zu können, zumindest nicht so, wie er ursprünglich vermittelt worden ist. Vielmehr sollte man davon ausgehen, dass durch digitale Technologien zahlreiche Beschäftigungsfelder hinsichtlich der Interaktion mit der IT tendenziell anspruchsvoller und komplexer werden. Eine ausreichende IT- und Methodenkenntnis zum alltäglichen Einsatz, aber auch das Wissen zu den Grenzen der Technologie werden immer relevanter. Lebenslanges Lernen und die Weiterbildung im Beruf müssen bei immer kürzer werdenden Innovationszyklen ein Bestandteil von Unternehmensfortbildungen sein. Die allgemeine Schulbildung, Berufs- und Hochschulausbildung sowie betriebliche Weiterbildung sind wichtige Stellschrauben für die zukünftige Fachkräftesicherung, denn ein Mangel an IT-Fachkräften ist schon seit Jahren in der deutschen Wirtschaft zu beklagen.

Junge Schülerinnen und Schüler schon möglichst früh an digitale Technologien heranzuführen, den Spaß am Programmieren zu vermitteln und so das Interesse an MINT-Themen zu steigern, ist eine Investition in die Zukunft. In Japan sind beispielsweise Grundkenntnisse in der Informatik, im Programmieren und über Algorithmen Pflichtbestandteil der Lehrpläne. Einfache algorithmische Strukturen zu erkennen und selbst formulieren zu können, schult wichtige Schlüsselkompetenzen wie analytisches Denken, Kreativität und die Fähigkeit, Probleme eigenständig zu lösen. Dadurch, dass Softwarecode eine immer grundlegendere Rolle in unserem Leben einnimmt, wird es immer

wichtiger, dass viele Leute ihn zu nutzen wissen und nicht nur ein kleiner Teil der Gesellschaft. Bereits Kindern sollte beigebracht werden, nicht nur Knöpfe zu bedienen, sondern auch zu verstehen, wie die Technologien dahinter funktionieren. Programmieren wird damit zu einer Art Fremdsprache in Schulen. Englisch, Französisch, Spanisch und Python gehören ganz selbstverständlich in einen gemeinsamen Lehrplan.

Ebenso wichtig ist es, die Bildungsprozesse zu modernisieren. Digitale Medien bereichern den Unterricht und können ihn aktiv mitgestalten. Digitale Werkzeuge erleichtern an vielen Stellen die Umsetzung von unterschiedlichen Methoden und Aufgaben. So können sich digitale Lernumgebungen auf die individuelle Lerngeschwindigkeit eines jeden einzelnen Schülers einstellen und ggf. Verständnisprobleme bei Grundlagen wiederholt erläutern. Man kann erwarten, dass digitale Schulbücher und Videomaterialien das Lernen einfacher machen, die Inhalte an das aktuelle Tagesgeschehen anpassen und dadurch die Qualität des Unterrichts gesteigert wird. Ein Computer, Laptop oder Tablet-PC für jeden Schüler in deutschen Schulklassen, ein funktionierender WLAN-Anschluss, ein leistungsfähiges Kommunikationssystem – von diesem rein infrastrukturellen Ziel sind wir derzeit weit entfernt. Es muss jedoch das Bewusstsein da sein, dass die digitale Bildung die Grundlage für die wirtschaftliche Zukunft der Bundesrepublik ist.

Dies gilt insbesondere vor dem anstehenden demographischen Wandel als eine zentrale Aufgabe. In den kommenden Jahren werden dem Arbeitsmarkt immer weniger Menschen zur Verfügung stehen. Digitale Bildung muss deshalb in allen Lebens- und Bildungsphasen vermittelt werden. Von diesen Maßnahmen sollen alle Beschäftigten, unabhängig von Bildung, Qualifikation, Alter, Geschlecht und Arbeitszeitmodellen, profitieren können.

Dabei ist es ganz zentral für die Ausbildung, wichtige Grundfähigkeiten zu vermitteln, die auch zukünftig nicht von einer KI übernommen werden können. Mit ihrer Kreativität, ihrer Intuition, ihrer Sicht auf Werte im Gegensatz zu den Fähigkeiten der algorithmischen Replizierbarkeit, Skalierbarkeit und Verarbeitung von großen Datenmengen, können Mensch und Maschine ihre jeweiligen Stärken ausnutzen und die Schwächen des jeweils anderen kompensieren. Für eine wertbringende Zusammenarbeit benötigt der Mensch jedoch auch eine klare Vorstellung über die Möglichkeiten und Grenzen der KI.

Entscheidend ist, dass die KI gegenwärtig und auch in Zukunft ein gewinnbringendes und nützliches Unterstützungsinstrument des Menschen ist und ihm Arbeiten abnimmt, sodass er sich auf andere Tätigkeiten konzentrieren kann. Dazu benötigt der Mensch die Kompetenzen, KI-Technologien entsprechend anzuwenden. Die Digitalisierung wird jedoch keine Fachexperten und deren fachspezifische Expertise in bestimmten komplexen Fachgebieten substituieren. Es werden immer menschliche Experten notwendig sein, die ihr Fachwissen und ihre Erfahrung einbringen.

## Das Berufsbild des Data Scientists

„Data Scientist: The Sexiest Job of the 21st Century“. Harvard Business Review, 2012

Mit der Big-Data-Welle im Jahre 2011 entstand ein neues, überaus attraktives Berufsbild. Data Scientists bilden die zentrale Berufsgruppe der Big-Data- und KI-Entwicklung. Sie

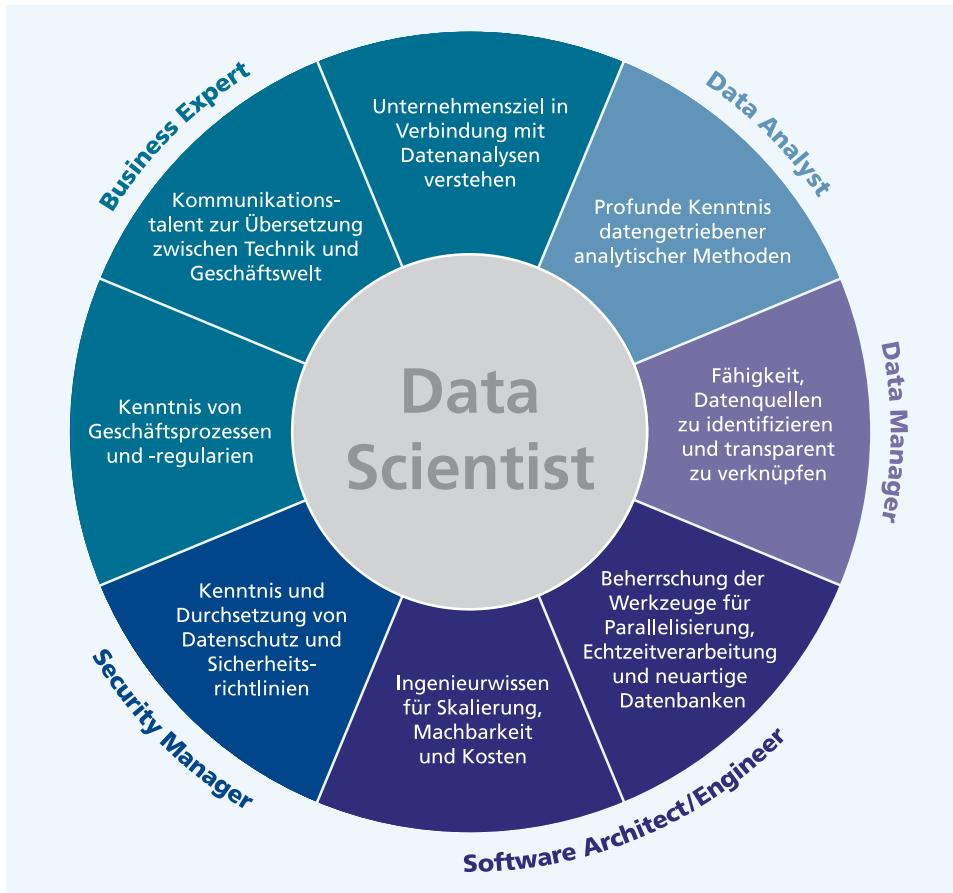
behalten beim exponentiellen Datenwachstum den Überblick und gewinnen Einsichten, aus denen neue Geschäftsmodelle und Services entstehen. Man kann den Data Scientist auch als Datendetektiv, Datenwissenschaftler, Daten-„gibt es noch was?“ bezeichnen (Abb. 10.26). Von Anfang an war klar, dass die Qualifikation des Personals in diesem Bereich zum Engpass werden wird. Die Abb. 10.17 hat ja schon verdeutlicht, wie die Datmengen zu einem essentiellen Unternehmensgut wurden. Nicht nur die Sensordatenverarbeitung für intelligente Geräte, auch der Bedarf an Bild-, Text- und Sprachverarbeitung für digitale Assistenten macht Experten für maschinelles Lernen zu einer äußerst knappen Ressource. Sucht man in Jobportalen nach „Data Scientist“, so wird man Hunderte offene Stellen finden. Aktuell übertrifft der Bedarf das Angebot an verfügbaren Data Scientist bei weitem, und das wird voraussichtlich noch einige Jahre so bleiben. Der Data Scientist wurde deswegen ganz zurecht von der Harvard Business Review bereits 2012 zum „sexiest job of the 21st century“ ausgerufen (Davenport und Patil 2012).

Data Scientists können derzeit in allen Branchen Beschäftigung finden, ob im Handel, in der Medizin oder in der Logistik. Bei der Entscheidung für ein Unternehmen ist nicht immer das Gehalt ausschlaggebend, sondern der Zugriff auf interessante Datenbestände, der Innovationsgrad der Anwendung und die Zusammensetzung des Teams. Denn idealerweise kennen sich die Datenwissenschaftlerinnen und -wissenschaftler nicht nur mit der aktuellen Software für Big-Data-Systeme und intelligenter Datenanalyse aus, sondern verstehen auch die jeweilige Branche. Gute Absolventen der einschlägigen Studiengänge werden nach ihrem Abschluss sogar häufig direkt von den Unternehmen angeworben, ohne dass sie sich selbst bewerben müssten. Bei der derzeitigen Nachfrage ist es für Personalabteilungen also kein leichtes, ein schlagkräftiges, kreatives Data-Science-Team zusammenzustellen. Hinzu kommt die Herausforderung, dass Data-Science-Teams aus verschiedenen Spezialisten mit unterschiedlichen Stärken zusammengesetzt werden müssen, um langfristig erfolgreich zu sein – denn „Allesköninger“ gibt es nur wenige.

**Abb. 10.26** Dhanurjay „DJ“

Patil wurde von US-Präsident Barack Obama im Jahr 2015 zum ersten Chief Data Scientist des United States Office of Science and Technology Policy ernannt. Patil erklärte, dass die Mission des U.S. Chief Data Scientist darin bestehe, die Macht der Daten verantwortungsbewusst zum Nutzen aller Amerikaner freizusetzen. Bildnachweis im Anhang





**Abb. 10.27** Das Berufsbild des Data Scientist zwischen Datenanalyse, Datenschutz und Kundenkommunikation

Das Berufsbild des Data Scientist lässt sich grob in vier Teilbereiche einteilen (Abb. 10.27):

1. Der Data Manager kennt die Datenbanksysteme und andere digitale Datenquellen wie Bilder, Texte oder Tonaufnahmen, und sorgt für die notwendigen Metadaten sowie eine nachhaltig hohe Datenqualität.
2. Der Business Expert muss identifizieren, welches Potential die vorliegenden Daten haben. Er kennt die Ziele des Unternehmens, den Markt und die wirtschaftliche Bedeutung der Daten. Durch sein Fachwissen bewertet er gefundene Muster und überträgt sie in mögliche Services.
3. Der Data Analyst muss mit statistischem Know-how relevante Zusammenhänge in den Daten aufdecken und mit Methoden des maschinellen Lernens Prognosemodelle oder TNN trainieren.

4. Geht es um die Bereitstellung der IT-Infrastruktur oder um die Handhabung großer Datenmengen aus verschiedenen Quellen, so ist ein Software-Architekt /-Ingenieur gefragt. Bei all dieser Anwendungen müssen Datenschutz und -sicherheit bedacht werden. Ein zusätzlicher Security Manager kann dafür sorgen, dass dies – by design – geschieht und zwar von Anfang an.

Um Data Scientist zu werden, sollte man im Idealfall bereits im Studium einige Schlüsselkompetenzen erworben haben. Dabei sind tiefe statistische Kenntnisse und ausreichende Programmierfähigkeiten für einen Data Scientist Pflicht. Tatsächlich handelt es sich um einen methodisch großen Sprung von der Analyse einer Excel-Tabelle hin zur Untersuchung von Datenmengen im Gigabyte-Bereich mit einer Vielzahl an Variablen. Die Analyse unstrukturierter Datenbestände, wie Sprache, Text und Bilder erfordert spezielle Methoden des tiefen maschinellen Lernens. Ein großer Erfahrungsschatz hilft, frühzeitig die bestmöglichen analytischen Verfahren einzuzgrenzen und belastbare Ergebnisse zu produzieren. Je mehr Projekte, Methoden und Fallstricke ein Data Scientist kennengelernt hat, umso effektiver wird er zukünftige Projekte durchführen können.

Interessant ist, dass das Berufsbild des Data Scientist nicht im universitären Umfeld entstanden ist, sondern aus einem wirtschaftlichen Interesse heraus. Inzwischen bieten zahlreiche Universitäten Data-Science-Studiengänge und -Module an. Das heißt aber auch, dass bei den aktuellen Datenwissenschaftlern ein Mix an Personen mit unterschiedlichen akademischen Hintergründen anzutreffen ist. Von Informatikern, über Mathematiker und Physiker bis zu Ingenieuren sind viele unterschiedliche Gruppen anzutreffen. Positiv ist sicherlich ein naturwissenschaftliches Studium.

Deutschland entwickelt mit seinen weltweit anerkannten Ingenieuren die besten Sensoren und Maschinen. Mittlerweile gilt allerdings, wer mehr Daten hat, kann die intelligenteren Geräte und dahinterstehenden KI-Services bauen. Wer seine Daten nutzbar macht, kann fundierte Entscheidungen treffen, Prozesse optimieren und dadurch Kosten sparen. Data Scientists sind die neuen Ingenieure in der deutschen Industrie. Hochschulen, die einen Data-Science-Studiengang anbieten, werden einen Standortvorteil für ihre Region bedeuten. Doch selbst wenn es an vielen anglo-amerikanischen Hochschulen Masterstudiengänge für Data Scientists gibt und inzwischen einige deutsche Hochschulen nachgezogen haben, reicht das nicht aus. Vorhandenes Personal in den Unternehmen muss für die neuen Methoden, Werkzeuge und Plattformen weitergebildet werden, und Branchenexperten müssen das Vokabular und das Verständnis bekommen, um in Teams mit Data Scientists einsetzbar zu sein.

---

### 10.3 Gesellschaftliche Herausforderungen

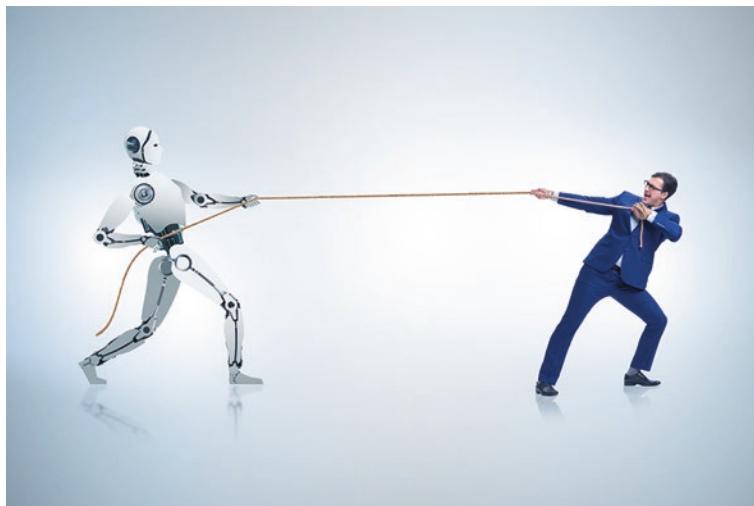
Neue technologische Möglichkeiten der KI können zum Positiven wie zum Negativen eingesetzt werden. Deutschland hat in seiner nationalen KI-Strategie „AI made in Germany“ (BMBF 2018) den Nutzen für Mensch und Umwelt in den Mittelpunkt gestellt. Ob dies

gelingt, wird stark darauf ankommen, inwieweit Deutschland und Europa KI-Technologien selbst beherrschen und erfolgreich umsetzen können. Nur wer eigene Produkte, Plattformen und Services baut, kann darin auch seine Wertvorstellungen und rechtstaatlichen Traditionen verwirklichen.

Es muss eine offene gesellschaftliche Diskussion zum Einsatz von KI geführt werden. Eine so durchgreifende und disruptive Technologie ist eine gesamtgesellschaftliche Herausforderung, der sich neben der Informatik verschiedene Fachdisziplinen, wie die Rechtswissenschaften, die Ethik und die Anwendungsdisciplinen stellen sollten. Um die gesamtgesellschaftliche Akzeptanz von KI zu erhöhen, muss gemeinsam daran gearbeitet werden, einen guten Mittelweg für die jetzige und auch zukünftige Nutzung von KI-Technologien zu finden. Dieser Prozess muss die Bedürfnisse von Unternehmen einerseits und Bürgerinnen und Bürgern andererseits gleichermaßen berücksichtigen – er darf also weder zu langsam, noch zu schnell vorstatten gehen und muss mit Sorgfalt gestaltet werden (Abb. 10.28).

KI wird viele Menschen entlasten, sollte ihnen aber nicht die Verantwortung für wichtige Entscheidungen abnehmen. Denn selbst bei gut trainierten KI-Systemen werden auch in Zukunft noch Fehler auftreten und falsche Vorschläge gemacht werden. Von Unfehlbarkeit sind wir weit entfernt, denn letztlich ist KI immer noch Statistik, die auf Daten beruht. Wir sehen, dass uns KI auf längere Zeit noch nicht ersetzen wird, aber es ist ein neues Hilfsmittel, das uns zur Verfügung steht. Ein neues Werkzeug.

Wir sind gewohnt, dass uns zunächst Lasttiere, dann Dampfmaschinen und später elektrische Maschinen unterstützen und entlasten, indem sie uns körperliche Arbeit abnehmen. Mit dem Computer sind wir in ein Zeitalter eingetreten, in dem uns Maschinen auch geistig entlasten. Neu ist jetzt, dass Computer quasi selbstständig lernen und nicht mehr nur



**Abb. 10.28** Die Technologie der KI kann viele positive Auswirkungen haben, z. B. in der Medizin. Viele Menschen sehen aber in TNN und Robotik einen potentiellen Konkurrenten, der ihnen den Arbeitsplatz streitig macht. Bildnachweis im Anhang

vorgegebene Programme abarbeiten. Dadurch können sie neue intellektuelle Aufgaben übernehmen, werden aber auch schwerer kontrollierbar. Beispielsweise ging die Entwicklung der Sprachverarbeitungsmodelle in letzter Zeit so rasant voran, dass wir noch gar nicht wissen, wo wir diese überall einsetzen können. So kann KI zwar Geschichten erzählen und kurze Redaktionsbeiträge verfassen, sicher gibt es aber auch noch ganz andere Anwendungsbereiche, für die solche Technologien in Frage kämen.

Es ist zu erwarten, dass es völlig neue Szenarien von Mensch-Maschine-Interaktionen geben wird, was dem Menschen abverlangt, sich fortlaufend auf neue Medientechnologie einzulassen. Ein ständiger Balanceakt zwischen einer Erweiterung unserer Fähigkeiten und Sinne, dem ständigen Druck des eigenen Optimierens und einem möglichen Verlust von Entscheidungshoheit.

### 10.3.1 Herausforderungen der KI in der Medizin

Vieles, was heute aus Ärzte- und Patientensicht noch futuristisch klingt, ist aus Sicht der KI-Forschung technisch bereits umsetzbar. Aber gerade im medizinischen Kontext herrscht eine besondere Skepsis gegenüber den oben genannten Innovationen. Der Arzt als Vertrauensperson und Vermittler spielt eine wichtige Rolle in der Einführung von technologischen Neuerungen. Ein „Smartes Krankenhaus“ muss zielgerichtet in Aufklärung und Vermittlung investieren, um Ärzten und Patienten die neuen Möglichkeiten und die Grenzen von KI zu vermitteln. Eine KI kann Symptome anhand von Mustern erkennen und Diagnosen vorschlagen, aber noch keine Behandlung und Intervention am Patienten durchführen. Dabei obliegt es weiterhin den Ärzten, für und mit Patienten Hypothesen und Alternativen abzuwägen und Therapien einzuleiten. Eine KI kann keinen hippokratischen Eid ablegen, dies kann nur der Arzt, der nach seinen ethischen und moralischen Grundsätzen eine Entscheidung trifft. Das entbindet eine KI jedoch nicht, ihre Vorschläge nachvollziehbar zu machen, um Vertrauen in die Anwendung zu vermitteln.

Damit ein funktionierendes Gesundheitsökosystem aufgebaut werden kann, müssen insbesondere rechtliche Rahmenbedingungen geklärt werden. Die heutigen KI-Systeme brauchen Daten mit hoher Qualität und in großen Mengen, um mit hoher Genauigkeit Muster zu erkennen und Prognosen zu geben. Medizinische Daten sind besonders sensibel, und die Ansprüche an Datenschutz und Anonymisierung sind hier am höchsten. Ein berechtigtes persönliches Interesse am Schutz von Gesundheitsdaten muss mit den gesellschaftlichen Interessen einer verbesserten Medizin rechtlich in Einklang gebracht werden.

Dabei muss ein möglicher Konflikt zwischen den Interessengruppen der Krankenkassen, Krankhäuser, niedergelassener Ärzte und Patienten aufgelöst werden, indem eine Privatsphären schützende Dateninfrastruktur gefunden wird (Abb. 10.29). Im Idealfall sollten die Parameter von trainierten Machine Learning-Modellen allgemeine Zusammenhänge und nicht Fakten über spezifische Trainingsbeispiele (z. B. Personen) kodieren. Um dies zu gewährleisten und starke Datenschutzgarantien zu geben, insbesondere dann, wenn die Trainingsdaten sensibel sind, ist es möglich, Techniken anzuwenden, die auf der Theorie



**Abb. 10.29** Im medizinischen Bereich ist die genaue Kontrolle der Zugangsberechtigungen der beteiligten Instanzen zu persönlichen Informationen besonders wichtig. Bildnachweis im Anhang

des Privacy Preserving Machine Learning basieren. Insbesondere beim Training mit Nutzerdaten bieten diese Techniken mathematische Garantien, dass Modelle nicht die Details über einen bestimmten Benutzer lernen oder sich daran erinnern. Für TNN können die zusätzlichen Garantien den Schutz durch Datenschutztechniken sinnvoll verstärken (Radebaugh 2019), sei es durch etablierte, wie Schwellenwerte, oder durch föderiertes Lernen.

### 10.3.2 1984 2.0: KI als Instrument der Überwachung

Wie der Deutschlandfunk berichtete, gab es in China Mitte bereits 2018 geschätzte 174 Millionen Überwachungskameras (Dorloff 2018). Viele davon können bereits Gesichter erkennen und diese auch zuordnen (Abschn. 5.10). An einer öffentlichen Toilette muss man sein Gesicht von einer Kamera scannen lassen, bevor man Toilettenpapier erhält. Personen, die mehrfach Papier haben wollen, werden abgelehnt. Gesichtserkennungskameras verhindern, dass Unbefugte ein Studentenheim betreten. In der Stadt Jinan werden auf Monitoren die Namen von Fußgängern gezeigt, die bei Rot über die Ampel gehen. Kunden können sogar per Gesichtserkennung ihre Rechnung begleichen, wie etwa beim Kentucky Fried Chicken in Hangzhou (García-Ajofrín 2018). Man kann mit klassischem Bargeld in den chinesischen Großstädten kaum noch bezahlen, selbst die Ein- und Ausfahrt aus einem Parkhaus ist ohne Gesichtserfassung oder spezielle App zur Abrechnung kaum noch möglich. Inzwischen wird in China die Beantragung eines Internetzugangs oder einer Mobilfunknummer nur gestattet, wenn zuvor das Gesicht gescannt wurde. In vielen Ländern muss man sich für einen Mobilfunkvertrag ausweisen, ein Gesichtsscan ist jedoch weltweit ein Novum (Deuber 2019). Von einem gläsernen digitalen Bürger ist China nicht mehr weit entfernt.

Neben der Bekämpfung von Kriminalität nutzt der chinesische Staat die Daten für eine riesige Datenbank mit Informationen zum sozialen Wohlverhalten. Jeder Bürger bekommt ein Punktekonto, in dem gute und schlechte Bewertungen automatisch gespeichert werden. Alle Aktivitäten, bei denen die Menschen im Tagesablauf beobachtet werden, können in die Bewertung, den „Social Credit“, eingehen (Mistreanu 2018). Das Bewertungsergebnis ist nicht geheim, sondern die Bürger mit bester Bewertung werden auf großen Anzeigetafeln bekannt gegeben (Abb. 10.30). Gut bewertete Mitbürger erhalten Vergünstigungen wie etwa Ermäßigungen bei der Heizungsabrechnung oder günstige Bankkredite. Und es gibt schon eine Gruppe von Menschen, die auf Grund des Systems bestraft wird: Fast 10 Millionen Personen wurden vom Ticketkauf für den Schnellzug oder das Flugzeug ausgeschlossen. Bis zum Jahr 2020 will China das System im ganzen Land einführen. Herzstück bleiben aber die Überwachungskameras, die vernetzt und in den kommenden Jahren noch massiv ausgebaut werden sollen. Das derzeit wertvollste Start-up SenseTime kommt aus China und ist im Bereich der Gesichtserkennung aktiv. Circa 600 Millionen Dollar wurden an Investmentgeldern eingesammelt (Bastian 2018). Der Markt in China scheint in diesem Anwendungskontext auch in Zukunft riesig zu sein.

Eine enge Vernetzung der großen chinesischen BAT-Konzerne mit der politischen Führung des Landes scheint gegeben zu sein und gewährleistet einen permanenten Datenaustausch zwischen Privatwirtschaft und staatlichen Behörden. KI liefert hiernach neue Wege für die Wenigen, die Vielen zu unterdrücken. KI „kann“ den perfekten Überwachungsstaat



**Abb. 10.30** Bewertung der Punktbesten im chinesischen Social Scoring System auf öffentlichen Schautafeln. Bildnachweis im Anhang (Mistreanu 2018)

ermöglichen und sehr effektiv machen. Dies ist eine Gefahr für die freie Gesellschaft. China ist auf dem Vormarsch, seine Bevölkerung ins permanente Kameravisier zu nehmen. Es ist die Vision von George Orwell, die nun zur Realität wird.

Eine neue mögliche Bedrohung der Privatsphäre ist die Technologie des Lippenlesens (Abschn. 7.4). Die vorliegenden Publikationen geben einen Wortfehler von 40 % an, aber man kann davon ausgehen, dass die Sicherheitsbehörden schon wesentlich trennschärfere Verfahren besitzen. Offenbar kann man nun Personen auf eine große Entfernung optisch „abhören“, die normalerweise gegen Überwachung sicher wären. Häufig sieht man schon bei Bundesligaspiele, dass Trainer ihre Anweisungen hinter vorgehaltener Hand weitergeben, damit der Gegner nichts davon erfährt.

Aber nicht nur in China findet eine Überwachung des öffentlichen Raumes statt, sondern auch in anderen Ländern. Beispielsweise gibt es etwa 500.000 Videokameras im Großraum London, mit denen Straßen und Plätze überwacht werden. Weiter ist bekannt, dass der BND täglich 220 Millionen Metadaten (Absender, Empfänger, ...) aus SMS, Email, WhatsApp etc. sammelte und zum Teil auch an die amerikanische CIA weitergab (Biermann 2015). Laut Gesetz gibt es keine Beschränkungen für die Überwachung von Ausländern durch den BND. Mit tiefen neuronalen Netzen zur Textanalyse (Kap. 6) können Inhalte (z. B. aus Emails oder Telefongesprächen) heute wesentlich zielgenauer analysiert werden als früher.

Eine intensive Beobachtung der Bürger findet aber auch durch die Internetfirmen statt. Auf jedem Smartphone rufen die meisten Nutzer eine große Anzahl von Webseiten und Apps auf, welche die Eingaben, den Standort, die Adressbücher und das Kauf- und Kommunikationsverhalten der Nutzer analysieren und dies unter anderem zur Einspielung zielgenauer Werbung verwenden. Durch die Verkäufung der Ergebnisse unterschiedlicher Apps über Werbe-IDs und Tracker lässt sich der Tagesablauf der Nutzer konsistent rekonstruieren und für unterschiedliche Zwecke auswerten (Valentino-deVries et al. 2018). Darüber hinaus werden auch immer mehr biometrische Daten durch Apps in der Cloud gesammelt, zum Beispiel Fotos des Nutzers und Messwerte des Fitnesstrackers, deren Verwendung oft nicht klar ist. Die Datenschutzgrundverordnung (DSGVO) erfordert zwar eine Einwilligung der Nutzer, die aber von den allermeisten Nutzern mit einem Klick erteilt wird. Alle diese Daten können mit Verfahren der künstlichen Intelligenz analysiert und zur Prognose der unterschiedlichsten Merkmale verwendet werden (Abb. 10.31).

In einer Studie untersuchten Kosinski et al. (2013) den Informationsgehalt von einfach zugänglichen Merkmalen von Nutzern, z. B. Facebook Likes. Sie zeigten, dass es möglich ist, mit Hilfe einer einfachen logistischen Regression (Abschn. 3.4) sehr private Eigenschaften der Nutzer zu prognostizieren, z. B. ob die Nutzer homosexuell sind (88 % Genauigkeit), ob sie schwarze Amerikaner sind (95 % Genauigkeit) oder ob sie Republikaner bzw. Demokraten wählen (85 % Genauigkeit). Dieses Verfahren wurde von dem Datenanalyse-Unternehmen Cambridge Analytica angewendet, um im großem Stil und meist ohne Einwilligung der Nutzer Daten über etwa 87 Millionen potentielle Wähler zu sammeln (Cadwalladr und Graham-Harrison 2018). Passend zu den prognostizierten Eigenschaften wurden individuell zugeschnittene Botschaften an die Nutzer geschickt, um



**Abb. 10.31** Überwachungskameras auf dem Autobahn in Thüringen an der A9. Bildnachweis im Anhang

das Wählerverhalten zu beeinflussen (Mikrotargeting). Cambridge Analytica unterstützte Donald Trump im Wahlkampf 2016, die pro-Brexit Kampagne von Boris Johnson und war bei mehr als 200 Wahlen rund um die Welt aktiv (Dachwitz et al. 2018). Diese Vorfälle zeigen, welche Möglichkeiten die großen Internetunternehmen habe, um Wissen über ihre Nutzer zu sammeln, und mit Hilfe fortgeschritten maschineller Lernverfahren für die unterschiedlichsten Zwecke zu nutzen. Peter Eckersley, leitender Informatiker der Electronic Frontier Foundation (EFF) sagt hierzu: „Es ist Facebooks Geschäftsmodell, so viele direkte und indirekte Daten wie möglich über Nutzer anzusammeln und dann zu bestimmen, wem es zu welchem Preis Zugang zu diesen gewährt. Wenn man Facebook benutzt, vertraut man dem Unternehmen Aufzeichnungen über alles an, was man tut. Ich denke, die Leute haben Gründe, darüber besorgt zu sein.“ (Tischbein 2016)

### 10.3.3 Krieg der Maschinen

*„Being killed by a machine is the ultimate human indignity“. Robert H. Latiff*

ALPHA heißt das Programm, das 2016 die Militärs weltweit zum Aufhorchen brachte (Bate 2016). Die Software lief auf einem Ein-Platinen-Computer Raspberry Pi: 64-Bit-ARM-Prozessor, vier Kerne mit 1,2 Gigahertz, 1 Gigabyte Arbeitsspeicher, Hardwarekosten weniger als 35 Euro. Jedes neuere Mobilfunkgerät hat eine höhere Leistung. Auf dieser Hardware lief die Software, die in einem Simulationsmodell in ein virtuelles Luftduell mit einem erfahrenen Kampfpiloten geschickt wurde. Das Ergebnis war erschreckend. In stundenlangen Luftkämpfen ist es dem ehemaligen Kampfpiloten und Ausbilder Gene Lee von der amerikanischen Luftwaffe kein einziges Mal gelungen, seinen Gegner ALPHA zu besiegen. Es sei das aggressivste, reaktionsfreudigste, dynamischste und überzeugendste KI-System, das er bisher gesehen habe, sagte Lee anschließend. Je nach Bedarf und Situation wechselte das

KI-System in ein defensives oder offensives Verhalten und zeigte sich unglaublich reaktionsschnell.

Falls sich dies fortsetzt, wird KI in Zukunft Teil der Flugbesatzung sein oder diese komplett ersetzen. Denn letztlich ist der Pilot mit seinen maximal ertragbaren G-Kräften nur ein Hindernis, das die Maschine unnötig in ihrer Beweglichkeit und Beschleunigung einschränkt. Kampfflugzeuge ohne Piloten könnten wendiger fliegen und kleiner und vielleicht auch kostengünstiger produziert werden.

Hier zeichnet sich ein Trend ab: Autonome Einheiten wie Drohnen werden demnächst nicht nur zur Grenzsicherung zum Einsatz kommen, sondern auch in kriegerischen Konfrontationen. Damit justiert sich die weltweite Balance des Militärs neu. In Zeiten des Kalten Krieges wurde häufig aufsummiert, wie viele Soldaten man unter Waffen hat und wie viele Panzer, Fregatten etc. zur Verfügung stehen – dies ist heute jedoch kein Maßstab mehr für die Leistungsfähigkeit eines Heeres. Durch Drohnen und andere autonome Einheiten wird KI zu einer zentralen Innovation des Militärs. Angst oder Müdigkeit spielen für diese Kampfeinheiten keine Rolle. Die Kampfhandlungen der Zukunft werden stärker automatisiert geführt werden als je zuvor.

Bereits im Jahre 2016 zeigte das U.S. Militär den erfolgreichen Einsatz eines Schwarms von Drohnen. Jede einzelne Drohne flog völlig autonom, koordinierte sich aber mit den anderen Drohnen als Team. Die Drohnen hatten die Mission, ein ca. drei Quadratkilometer großes Gebiet zu patrouillieren. Drei Kampfflugzeuge der Navy wurden beim Eintreten in das Gebiet erfasst und vom Drohnenteam eingekreist. „Perdix“ nennt sich diese Entwicklung des Militärs, welche sicherlich nur ein Auftakt zu weiteren Entwicklungen darstellt (Slavin 2017) (Abb. 10.32).

Hier besteht gesellschaftlicher Handlungsbedarf, denn Roboter haben keine ethischen Bedenken und kein Gewissen, das ihnen im Kampfgeschehen Grenzen aufzeigt. Den Bau von Drohnen, die selbstständig über ihren Waffeneinsatz entscheiden, gilt es zu verhindern. Normen für den Einsatz von autonomen Kriegsgeräten gibt es bisher nicht. Das Bestreben vieler Staaten für ein Verbot solcher autonomen Waffensysteme scheiterte bislang



**Abb. 10.32** Die Kampfdrohne Perdix des Pentagon auf einer lokalen Trainingsmission. Gleichartige Drohnen dieses Typs können ihre Aktionen in einem Schwarm koordinieren. Bildnachweis im Anhang

am Widerstand von Ländern mit großer Waffenindustrie (KillerRobots 2018). Es obliegt nun der Zivilgesellschaft, von ihren Regierungen weltweit verbindliche Standards und Verbote für autonome Waffensysteme und entsprechende völkerrechtliche Vereinbarungen einzufordern.

Ein Fokus der zukünftigen Sicherheitsbedrohung unserer Zeit sind Cyberangriffe. Es sind KI-Systeme die selbstständig nach Schwachstellen in der Infrastruktur, der Kontrolle von Finanzströmen und im Datenmanagement fremder Staaten suchen, um diese im Falle einer kriegerischen Auseinandersetzung auszuschalten bzw. wichtige Informationen zu erhalten. Das Schadprogramm Stuxnet aus dem Jahre 2010 ist ein prominentes Beispiel. Stuxnet diente dazu, die Steuerung von Frequenzumrichtern zu sabotieren und damit Motoren von Wasserwerken und Pipelines gezielt zu schädigen. 2010 wurde der Virus eingesetzt um das iranische Atomprogramm zu stören, der Auftraggeber wurde nie öffentlich bekannt. Nationalstaaten sind dabei, Cybereinheiten aufzubauen, um solche Angriffe zu verhindern. Im Vergleich zu herkömmlichen Investitionen in Kampfflugzeuge, Panzer und Flugzeugträger sind sie kostengünstiger, können aber trotzdem sehr effektiv sein (Biermann 2019). Die Abhörpraktiken der amerikanischen NSA und die Enthüllungen von Edward Snowden zeigen, dass selbst unter befreundeten Staaten keine Grenzen existieren.

### 10.3.4 Starke Künstliche Intelligenz

Bei Künstlicher Intelligenz unterscheidet man zwischen starker und schwacher KI. Schwache KI fokussiert auf Systeme, die konkrete Anwendungsprobleme lösen, wie sie in den vergangenen Kapiteln exemplarisch vorgestellt wurden (Abschn. 2.10). Hier konnten in den letzten Jahren zahlreiche Durchbrüche erzielt werden. Dabei konnte KI oft die menschlichen Fähigkeiten im jeweiligen Spezialgebiet erreichen oder sogar übertreffen. Alle bislang existierenden Systeme fallen unter die schwache KI. Sie entwickeln kein tieferes Verständnis für die Problemlösung oder ihre kausalen Zusammenhänge.

Die starke Künstliche Intelligenz (auch „allgemeine KI“) hingegen soll Problemstellungen in nicht vorab definierten Situationen und Aufgaben bewältigen. Ihr Vorbild ist der Mensch, der für ihn ungewohnte Situationen meistern kann, selbst wenn kaum oder gar keine Erfahrungswerte vorliegen. Menschen lernen im Gegensatz zu den aktuellen KI-Systemen sehr effizient und benötigen dafür oft nur wenige Informationen aus ihrer Umwelt. Ein KI-System kann bis dato nur schwer aus Einzelfällen lernen und braucht eine große Menge an Trainingsbeispielen. Sollte eine starke künstliche Intelligenz einmal verfügbar sein, so würde sie selbsttätig in großem Umfang Daten sammeln, sich möglicherweise rasant selbst verbessern und könnte nicht absehbare Änderungen zur Folge haben. Dieser Zeitpunkt wird in der Zukunftsforschung technologische Singularität genannt (Abb. 10.33).

Das Ziel einer starken Künstlichen Intelligenz ist es, die gleichen intellektuellen Fertigkeiten wie Menschen zu erlangen oder gar zu übertreffen. Sie kombinieren logisches Den-



**Abb. 10.33** Die Verfügbarkeit der starken Künstlichen Intelligenz wird auch als eine technologische Singularität bezeichnet, weil dann KI-Systeme potentiell selbstbestimmt tätig werden könnten. Bildnachweis im Anhang

ken, Planen, Lernen, Kommunikation in natürlicher Sprache und die Fähigkeit, auch bei Unsicherheit Entscheidungen treffen zu können, um selbstgesteckte Ziele zu erreichen. Sie handeln und lernen aus einem eigenen Antrieb. Dieser ständige Antrieb nach Neuem hat uns Menschen zu Entdeckern und Forschern gemacht.

Ob eine KI jemals ein eigenes Bewusstsein erlangen kann, ob sie Empathie für Menschen oder eine andere starke KI entwickeln und die Fähigkeit zur Reflektion, moralischem und ethischem Denken besitzen kann, ist völlig offen. In Abschn. 10.4.2 werden wir diese Frage aus technologischer Sicht diskutieren. Eine solche sogenannte „Superintelligenz“, die nur von einer möglichst perfekten Zielerreichung getrieben wird und grundlegende humanistische Werte missachtet, könnte zu einer Gefahr für uns werden. Persönlichkeiten wie Elon Musk warnen deshalb vor einer solchen Entwicklung. Initiativen, wie die von ihm unterstützte OpenAI (Metz 2016), arbeiten an einer frei zugänglichen starken KI, die der Gesellschaft Vorteile bringen und nicht schaden soll. Fraglich ist jedoch, was solche Projekte bringen, wenn letztlich jede Form von Technologie auch missbraucht werden kann. So sagte der Philosoph und Risikoforscher Nick Bostrom 2016: „Wenn Sie einen Knopf haben, der Schaden in der Welt anrichten kann, würden Sie ihn nicht jedem verfügbar machen.“ Tatsächlich veröffentlichte OpenAI im Februar 2019 zunächst nicht die volle Version ihres neuesten Texterzeugungssystems GPT-2 (Abschn. 9.2.3) aus Sorge, es könne für automatisierte Fake News und Spam-Kommentare in sozialen Medien missbraucht werden (Whittaker 2019).

Aktuell sind wir noch weit von einer starken KI entfernt. Forscher gehen davon aus, dass erste Durchbrüche in 20 bis 40 Jahren zu erwarten sind. Ähnliche Zeitspannen wurden allerdings auch schon früher angegeben. So sagte Herbert Simon im Jahr 1965: „Maschinen werden innerhalb der nächsten zwanzig Jahre dazu imstande sein, jede Arbeit zu übernehmen, die auch der Mensch ausführen kann.“ Andere Forscher behaupten, eine

starke KI werde es niemals geben. Der Fortschritt in der KI ist unglaublich schwer einzuschätzen, und es ist nicht auszuschließen, dass eine starke KI irgendwann entwickelt wird. Auch wenn es sicher noch ein weiter Weg ist, bewegen wir uns mit jedem Tag einen kleinen Schritt in diese Richtung.

---

## 10.4 Methodische Herausforderungen

Mittlerweile ist tiefes Lernen der mit Abstand erfolgreichste Ansatz der Künstlichen Intelligenz. In den letzten Jahren konnten viele KI-Probleme auf diese Art erstmals erfolgreich angegangen werden (Abb. 10.34). Seitdem wird in den Medien und öffentlichen Debatten viel über KI diskutiert. Einige KI-Enthusiasten glauben an rasche Fortschritte in der KI. Andrew Ng, ein führender KI-Wissenschaftler, meint „Wenn ein typischer Mensch eine mentale Aufgabe in weniger als einer Sekunde erledigen kann, können wir sie wahrscheinlich entweder jetzt oder in naher Zukunft mithilfe von KI automatisieren“ (Ng 2016). Shane Legg, der Chefwissenschaftler von Google’s DeepMind-Gruppe sagte: „KI auf menschlichem Niveau wird Mitte der 2020er-Jahre erreicht“ (Mitchell 2018).

Viele Wissenschaftler sehen aber auch Probleme bei der Weiterentwicklung der KI. Gary Marcus, Professor für natürliche und Künstliche Intelligenz, fragt: „Nähert sich das tiefe Lernen einer Mauer?“ (Marcus 2018). Und die Informatikprofessorin Melanie Mitchell befürchtet: „Künstliche Intelligenz stößt an eine Grenze der Bedeutung“ (Mitchell 2018). In diesem Abschnitt wollen wir einige methodische Herausforderungen der KI diskutieren.



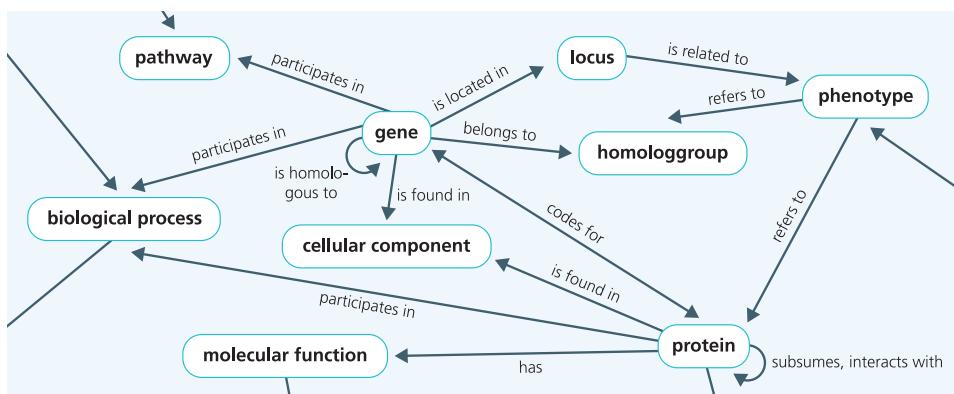
**Abb. 10.34** Die Methoden der Künstlichen Intelligenz haben in den letzten Jahren in spezialisierten Anwendungen große Erfolge gehabt. Es gibt in einer Reihe von übergreifenden Aufgaben aber noch große Defizite. Bildnachweis im Anhang

### 10.4.1 Kombination von Daten und unscharfem Schließen

Die in diesem Buch vorgestellten TNN beruhen darauf, dass nicht explizit formulierte Zusammenhänge zwischen Merkmalen automatisch aus beobachteten Daten extrahiert und durch Anpassung der Modellparameter gespeichert werden. Daher können zunächst einmal nur Probleme gelöst werden, für die Daten verfügbar sind, sodass die Zusammenhänge zwischen den relevanten Merkmalen trainiert werden können.

Häufig gibt es aber Zusatzinformationen, die bei der Lösung von Problemen verwendet werden können. Dieses Wissen kann auf zwei Arten gewonnen werden. Wir haben schon Systeme für die Extraktion von Weltwissen durch Transferlernen auf (Abschn. 6.8) dargestellt. Hierbei wurde unter anderem ein BERT-Modell mit einem großen Trainingskorpus vortrainiert. Anschließend wurde es durch Feintuning auf einem zusätzlichen Datensatz in die Lage versetzt, vorher nicht spezifizierte inhaltliche Fragen beantworten.

Ein alternativer Ansatz ist die Nutzung von explizit formuliertem Wissen oder symbolische Künstliche Intelligenz. Schon seit den 80er-Jahren werden sogenannte Wissensdatenbanken aufgebaut, in denen Wissen in Form von Fakten und Regeln beschrieben wird. Ein Beispiel ist Cyc, welches mehr als 24,5 Millionen manuell eingegebene Regeln und Fakten enthält (Abschn. 6.8.3). In Cyc kann man Regeln formulieren, die für alle Objekte eines Typs gelten, z. B. „alle Vögel haben Federn“. In Cyc gibt es dann ein Modul für die Herleitung von Schlussfolgerungen wie „ein Spatz hat Federn und zwei Beine“. Solche Ableitungen sind besonders wichtig, da sie mit Sicherheit gelten – im Gegensatz zu vielen Resultaten tiefer neuronaler Netze, welche nur mit einer gewissen Wahrscheinlichkeit korrekt sind. Es gibt weitere Wissensdatenbanken, z. B. Wikidata, welches Wissen aus Wikipedia in Form von Fakten und Regeln enthält, z. B. geborenIn (Friedrich\_Schiller, Marbach) (Erxleben et al. 2014). Im Bereich der Medizin (Abb. 10.35) gibt es die Medical Subject Headings (MeSH-Wissensdatenbank), für die medizinische Fakten und Regeln aus etwa 30 Millionen medizinischen Fachartikeln extrahiert wurden (Richter und Austin 2012). Man kann damit zum Beispiel nach Artikeln zu den Wirkungen eines



**Abb. 10.35** Ausschnitt aus einer biologischen Wissensdatenbank

bestimmten Medikamente suchen. Einen aktuellen Überblick über Techniken der symbolischen KI zusammen mit maschinellem Lernen bieten (Görz et al. 2020).

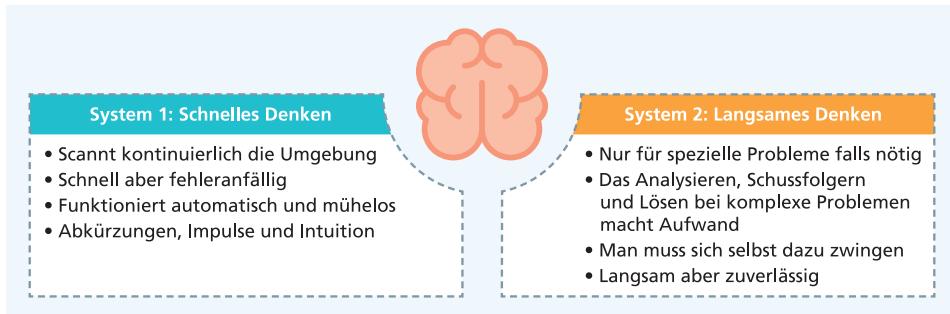
Sieht man von Definitionsbeziehungen („ein Löwe ist ein Säugetier“) ab, so stellt sich in der Praxis heraus, dass Regeln meist nicht mit Sicherheit, sondern nur bis zu einem gewissen Grad gelten. Aus diesem Grunde kann man Fakten und Regeln mit einer Maßzahl für ihre Plausibilität versehen. Zum Beispiel gilt `personWirdKrank (starker_Raucher, Lungenkrebs)` mit einer Wahrscheinlichkeit von 0,244 und die Wahrscheinlichkeit von `personÜberlebt (Lungenkrebs, 5Jahre)` ist 0,15. Kombiniert man solche unsicheren Fakten und Regeln, so kann man daraus Schlüsse ziehen, welche selbst wieder mit einer Plausibilität versehen sind. Derartige Schlussverfahren sind komplex, weil die formulierten Regeln meist nicht statistisch unabhängig voneinander gelten. Eine solche Art von Schlussfolgerungen nennt man unscharfes Schließen.

Um also Probleme zu lösen, bei denen Informationen aus Daten (z. B. Sensordaten) und Wissensdatenbanken kombiniert werden müssen, benötigt man Verfahren, die beide Arten von Information in einem gemeinsamen Lernalgorithmus verknüpfen. Diese Art von KI wird auch als hybride KI bezeichnet, weil hier unterschiedliche Informationstypen und Algorithmen kombiniert werden. (Fischer et al. 2019) beispielsweise übersetzen jede unsichere Regel in einen Term der Verlustfunktion, welcher den Wert 0 annimmt, wenn die Regel erfüllt ist. Zusätzlich wird die Regel mit einem Gewicht versehen, welches ihrer Plausibilität entspricht. Durch dieses Vorgehen kann man die Gültigkeit der logischen Zusammenhänge für das Modell erzwingen und auch nach Eingaben suchen, die gewisse logische Bedingungen erfüllen. Manhaeve et al. (2018) definieren eine Programmiersprache, welche probabilistisches Schließen mit TNN kombiniert. Hierdurch können Regeln, die mit einer gewissen Wahrscheinlichkeit gelten, mit neuronalen Netzen verknüpft und mithilfe von Daten trainiert werden. Schließlich gibt es auch Erweiterungen für die Programmierumgebung TensorFlow, mit denen neuronale Netze und unsichere Regeln kombiniert werden können (TensorProb 2019).

Obwohl erste Ansätze existieren, müssen die Theorie und die praktischen Werkzeuge für die Integration von unscharfem Schließen und TNN noch stark verbessert werden, damit diese Methoden in realen Anwendungen benutzt werden können. Es ist noch nicht absehbar, ob sich die Verfahren der hybriden KI oder das Transferlernen durchsetzen werden.

#### 10.4.2 Schnelles und Langsames Denken

Intelligentes Denken findet in unterschiedlichen Geschwindigkeiten statt. Daniel Kahneman, Nobelpreisträger für Ökonomie, hat aus langen Untersuchungen menschlichen Verhaltens eine These (Kahneman 2011) über zwei unterschiedliche Systeme des Denkens entwickelt (Abb. 10.36). System 1 (Schnelles Denken) ist schnell, instinkтив und emotional. Beispiele sind das Verstehen eines einfachen gesprochenen Satzes, das Autofahren auf einer unbelebten Straße oder das Erkennen eines Objektes in einem Bild. System 1 läuft kontinuierlich und erzeugt Eindrücke, Intuitionen und schnelle Urteile auf Grund unserer unmittelbaren Wahrnehmungen.



**Abb. 10.36** Die Eigenschaften der beiden Systeme zum Schnellen und Langsamen Denken im menschlichen Gehirn nach (Kahneman 2011)

System 2 (Langsames Denken) ist langsamer, bedächtiger und logischer. Es ist zum Beispiel zuständig für das Erinnern an eine lange nicht gesehene Person, das Einparken in einen schmalen Parkplatz oder die Lösung der Rechenaufgabe  $16*34$ . Das System 2 wird nur verwendet, wenn sich bei System 1 Probleme ergeben, es also die Wahrnehmungen nicht gut erklären kann.

Dem System 2 entspricht im Gehirn ein Arbeitsgedächtnis mit begrenzter Kapazität (Diamond 2013). Es ermöglicht, Gedankeninhalte kurzfristig zu speichern und gleichzeitig zu manipulieren. Es hat offenbar eine wichtige Rolle beim Problemlösen und bei logischen Schlussfolgerungen. Die Anzahl der gleichzeitig handhabbaren Informationseinheiten liegt nach Schätzungen zwischen fünf und sieben. Der Mensch ist sich der Denkvorgänge von System 2 bewusst, während die Verarbeitung im System 1 weitgehend unterbewusst stattfindet.

Ein weiterer wichtiger Mechanismus des Systems 2 ist die Aufmerksamkeit. Hierbei werden die beschränkten Ressourcen der bewussten Überlegung auf Inhalte in anderen Bereichen des Gehirns fokussiert. Das können Wahrnehmungen der Umwelt, des eigenen Handelns, aber auch Gedanken und Gefühle sein. Damit ist andererseits verbunden, dass man andere Reize im Zustand der Aufmerksamkeit oft nicht wahrnimmt. Bengio (2019) weist darauf hin, dass Aufmerksamkeit in heutigen TNN durch Attention-Mechanismen (Abschn. 6.5.2) realisiert werden kann, welche ja die Berechnungen auf spezielle Inhalte fokussieren.

Pearl (2018) argumentiert, dass es unterschiedliche Arten von Fragestellungen für Lernalgorithmen gibt:

- Assoziativ: Was sagen die vorliegenden Hautflecken über das Vorliegen von Hautkrebs aus? Hier werden alleine statistische Zusammenhänge in den Daten ausgewertet.
- Situationsänderung: Wie ändert sich der Umsatz, wenn wir den Preis verdoppeln? Diese Art von Fragestellung kann nicht aus historischen Daten allein beantwortet werden, weil hier eine Reaktion des Kunden mitberücksichtigt werden muss, der auf den höheren Preis reagiert.
- Kontrafaktisches Schließen: Würde John F. Kennedy noch leben, wenn Harvey Oswald ihn nicht erschossen hätte? Hier geht es darum, Zusammenhänge zu ergründen, welche im Gegensatz zu dem tatsächlichen Geschehen stehen (Abb. 10.37).

**Abb. 10.37** Kontrafaktisches Schließen ist ein wichtiger Ansatz zur Lösung komplexer Probleme. Bildnachweis im Anhang



Für die letzten beiden Arten von Fragestellungen benötigt man Modelle des Anwendungsbereichs, mit denen man Alternativen simulieren kann, also z. B. das oben erwähnte Arbeitsgedächtnis. Wichtig ist hierbei, Ursachen von Wirkungen zu unterscheiden und die Kausalität der Zusammenhänge aufzudecken.

In den meisten Ansätzen von tiefem Lernen sind solche Fragestellungen nicht vorgesehen. Allerdings kann man Modelle des Bestärkungslernens nutzen (Kap. 8). Hier wird ein Modell der Umgebung, also ein Simulationsmodell, benötigt, welches durch Zustände gekennzeichnet ist und auf Aktionen des Nutzers reagiert. Mit Hilfe dieses Umgebungsmodells kann man dann eine Strategie lernen, die eine hohe Belohnung ergibt. Allerdings ist die Verwendung von Bestärkungslernen für beispielsweise kontrafaktisches Schließen noch in den Anfängen.

Viele Anwendungen der tiefen neuronalen Netze betreffen den Bereich des Schnellen Denkens: Erkennen eines Gegenstandes in einem Bild, Erkennung der Bedeutung gesprochener Worte oder Steuerung von Bewegungsabläufen wie Gehen oder Radfahren. Sie laufen weitgehend assoziativ ab. Komplexe Überlegungen wie etwa kontrafaktisches Schließen gehören zum Langsamen Denken. Auch für das menschliche Gehirn bedeutet die Lösung solcher Aufgaben eine merkliche Anstrengung.

Jack Nicklaus war einer der größten professionellen Golfspieler aller Zeiten. In seiner Autobiografie (Nicklaus 2007) beschreibt er, wie er sich auf einen Abschlag vorbereitet (Abb. 10.38): „Vor jedem Schlag gehe ich in meinem Kopf ins Kino. Ich sehe Folgendes: Zuerst sehe ich den Ball dort, wo er enden soll, schön weiß und hoch oben auf dem hellgrünen Gras sitzend. Dann sehe ich den Ball dorthin fliegen; seinen Weg und seine Flugbahn und sogar sein Verhalten bei der Landung. Die nächste Szene zeigt, wie ich den Schwung mache, der das vorherige Bild in die Realität umsetzt. Diese Heimvideos sind ein Schlüssel zu meiner Konzentration und zu meiner positiven Einstellung bei jedem Schlag.“ Nicklaus sieht also den Verlauf des Schlages und die Flugbahn des Balles in der zeitlichen Abfolge. Er kann damit seinen Schlag so planen, dass er das gewünschte Ziel erreicht.



**Abb. 10.38** Mögliche Flugbahnen des Golfballes beim Abschlag. Bildnachweis im Anhang

Hamrick (2019) diskutiert die Eignung von Modellen des Bestärkungslernens für Problemlösungen dieser Art und vergleicht sie mit mentalen Simulationen beim Menschen. Eine Schwierigkeit ist, dass derzeit beim Bestärkungslernen für jede Fragestellung ein neues Simulationsmodell der Umgebung trainiert oder programmiert werden muss. Hingegen ist der Mensch in der Lage, solche Simulationen ad hoc durchzuführen, selbst wenn deren Genauigkeit oft nicht groß ist. Während Bestärkungslernen vielversprechend für den Aufbau flexibler, robuster Modelle für eingeschränkte Fragestellungen ist, bleibt noch sehr viel zu tun, bis menschliche Vorstellungsmöglichkeiten erreicht werden. Um den menschlichen Fähigkeiten zu entsprechen, müssen Modelle aus einzelnen Teilmodellen aufgebaut und bei Bedarf zusammengestellt werden. Methoden für die Strategiesuche müssen mit nur einer Handvoll Bewertungen von verrauschten, unvollständigen Modellen erfolgreich sein; und Modelle müssen in der Lage sein, auf Situationen weit außerhalb ihrer Trainingsdaten zu generalisieren, um kreative Erkundungen und ein besseres Verständnis der Umgebung zu unterstützen.

Nach Ansicht von Bengio (2019) können die Aufgaben von Kahnemanns System 1 heute schon recht gut von den existierenden TNN übernommen werden. Seiner Ansicht nach fehlen drei wichtige Fähigkeiten, damit auch Aufgaben von System 2 automatisch durchgeführt werden können:

- Die Übertragung von trainierten Zusammenhängen auf neue Bereiche. Erste Ansätze dazu gibt es beim Transferlernen (Abschn. 5.7).
- Die Manipulation von abstrakten semantischen Konzepten, wie sie im System 2 stattfindet. Dabei müssen Konzepte sensorisch in der Welt durch System 1 verankert (Symbol-Grounding) und Inhalte durch verteilte Repräsentationen dargestellt werden. Wichtig ist dabei die Berücksichtigung von Unsicherheit.

- Es müssen Agenten betrachtet werden, die sich in einer Umwelt bewegen und agieren können. Sie müssen Weltwissen sammeln, die Zusammenhänge zwischen Sensorinhalten, Objekten der Umgebung und Aktionen in Beziehung setzen und darüber hinaus aktiv nach neuem Wissen suchen.

Diese Fähigkeiten sind derzeit noch nicht verfügbar.

Jordan (2018) weist auf eine Reihe von Problemen der jetzigen KI hin: „Diese Probleme umfassen die Notwendigkeit, Bedeutung und logisches Schließen in Systeme einzubringen, die eine Verarbeitung natürlicher Sprache durchführen, die Notwendigkeit, Kausalität abzuleiten und darzustellen, die Notwendigkeit, effizient zu berechnende Darstellungen von Unsicherheit zu entwickeln, und Systeme zu entwickeln, die langfristige Ziele formulieren und verfolgen. Dies sind klassische Ziele in der menschenimitierenden KI, aber im gegenwärtigen Lärm um die ‚KI-Revolution‘ ist es leicht zu vergessen, dass sie noch nicht gelöst sind.“

---

## 10.5 Vertrauen schaffen in die KI

Cathy O’Neill ist eine amerikanische Mathematikerin, die nach ihrem Studienabschluss einige Jahre in der Finanzbranche bei Hedgefond-Firmen arbeitete. Desillusioniert von ihrer Arbeit wechselte sie an die Columbia University und schrieb ein bemerkenswertes Buch mit dem Titel „Weapons of Math Destruction“ über die gesellschaftlichen Auswirkungen von Algorithmen (O’Neil 2016). In dem Buch erläutert sie wie Algorithmen dazu eingesetzt werden können, vorhandene Ungleichheiten zu verstärken und weiter auszubauen (Abb. 10.39).

Automatisierte Entscheidungen, die auf maschinellem Lernen auf großen Datenmengen beruhen, stellen unsere Gesellschaft vor ganz neue Herausforderungen. Wir wollen eine gemeinwohl-, humanorientierte KI, die im Einklang mit unseren Werten steht, verlässlich funktioniert und dabei wettbewerbsfähig ist. Wir müssen deshalb versuchen, faire und objektive Ergebnisse zu erzielen und unerwünschte Folgen zu vermeiden. Faires Verhalten einer KI-Anwendung gegenüber allen Beteiligten, Berücksichtigung der Bedürfnisse von Nutzerinnen und Nutzern, eine verlässliche, verständliche und sichere Funktionsweise sowie der Schutz sensibler Daten sind zentrale Anforderungen an den vertrauenswürdigen Einsatz einer KI. Mangelndes Bewusstsein für ethische und rechtliche Fragen und die Risiken des Dual-Use können zum Gegenteil dessen führen, was bezeichnet ist. Dabei wirft die Möglichkeit der automatisierten Entscheidungen durch eine KI ein neues Licht auf die individuelle Haftung und Verantwortung von Personen. Um KI-Systeme menschengerecht und vertrauenswürdig zu gestalten, bedarf es einer gesellschaftlichen Verständigung darüber, welche Leitplanken, Prinzipien und Grenzen gelten sollen, wie KI-Anwendungen entwickelt werden sollen, wie autonom sie agieren dürfen und wie wir sie kontrollieren möchten.

Bei einer Präsentation im Jahre 2018 hat sich der Sprachassistent Google Duplex (Abschn. 7.5.2) am Telefon täuschend echt menschlich verhalten (Leviathan 2018). Google

**Abb. 10.39** Cathy O'Neill ist eine amerikanische Informatikerin und kritisiert die Anwendung der KI in der Wirtschaft. Bildnachweis im Anhang



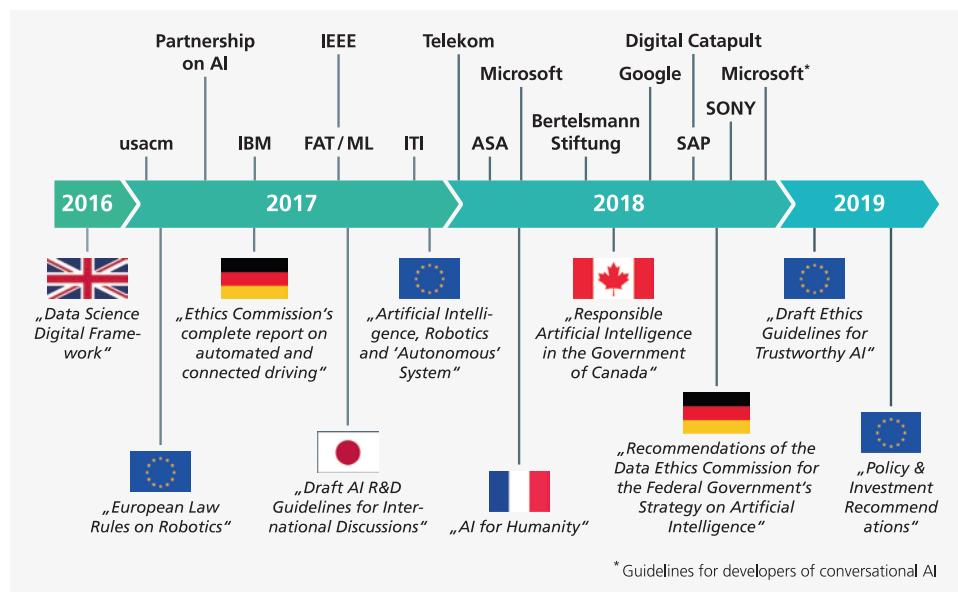
Duplex soll das oftmals lästige Telefonieren für Terminvereinbarungen mit einem Friseur oder Restaurant übernehmen. Die Einsatzbereiche sind aber viel umfangreicher, denn natürlich haben auch Call-Center großes Interesse an dieser Technologie. Hier entsteht ein Transparenzproblem. Grundsätzlich sollten Sprachassistenten am Telefon gleich zu Beginn sagen, dass sie ein Automat sind und in welchem Auftrag sie handeln. Dann sollten sie uns die Möglichkeit geben, uns stattdessen mit einem Menschen verbinden zu lassen. Am besten gleich mit einer Abschätzung der Wartezeit. Eine solche Weiterleitung sollte zudem jederzeit während des Dialogs möglich sein. Generell ist es unabdingbar, dass KI-Systeme von Anfang an Transparenz schaffen und es für den Menschen absolut nachvollziehbar bleibt, wann und wie er mit einer Maschine kommuniziert und wie er die Interaktion steuern kann. Sicherlich wird es sehr spannend, wenn überhaupt kein Mensch involviert ist, sondern zwei Sprachassistenten direkt aufeinandertreffen. Solche Situationen sind in Zukunft nicht abwegig und müssen selbstverständlich frühzeitig bedacht und geklärt werden. Wenn Menschen Entscheidungen an Maschinen delegieren, wer ist dann verantwortlich?

Das EU-Parlament hat am 16.02.2017 per Resolution zivilrechtliche Regelungen für Roboter gefordert ([Europa 2017](#)). In Bezug auf Haftungsfragen sollen sie den Status einer elektronischen Person bekommen. Tatsächlich hat das Zivilrecht Lücken bei Robotern und bei digitalen Agenten, die Entscheidungen unter Ungewissheit treffen können. Ungewissheit heißt, dass sie zwischen Alternativen entscheiden können, indem sie mehrere Kriterien optimieren, ohne dass ihr Verhalten vorhersagbar programmiert wurde. Das trifft zum Beispiel auf die oben genannten Sprachassistenten zu, die quasi als digitale Stellvertreter am Telefon mit einem Menschen oder einem anderen digitalen Agenten Termine verein-

baren, Buchungen vornehmen oder Verträge abschließen. Nach dem geltenden Recht ist für Schäden, die ein Roboter oder eine Maschine verursacht, der Nutzer verantwortlich. Nach der aktuellen Überlegung des EU-Parlaments gilt dies als nicht mehr angemessen, wenn sich die Maschine aufgrund eigenständig erlernter Verhaltensmuster dem Einfluss ihres Besitzers entzieht. In seiner Resolution geht das Europäische Parlament davon aus, dass selbstlernende Systeme und autonome Roboter nicht mehr als Werkzeuge eines Herstellers, Betreibers oder Nutzers angesehen werden können. Insbesondere die Verbindlichkeit von Vereinbarungen, die vertragliche Haftung und die Haftung im Schadensfall müssen hier überdacht und neu geregelt werden. Eine Idee könnte sein, eine spezielle Haftpflichtversicherung für Roboter (E-Personen) einzuführen, deren Preise abhängig von der Autonomiestufe und dem Risiko der Anwendung kalkuliert werden.

Die Frage, wie KI-Anwendungen verantwortlich und zuverlässig eingesetzt werden können, ist bereits seit einiger Zeit Gegenstand intensiver gesellschaftlicher und wissenschaftlicher Diskussionen im internationalen Raum. Es ist positiv beeindruckend, wie in den vergangenen Jahren KI-Richtlinien von zahlreichen Staaten, NGOs und Unternehmen entwickelt wurden. Die wichtigsten zeigt die Abb. 10.40.

Auf europäischer Ebene hat die EU-Kommission eine High-Level Expert Group für Künstliche Intelligenz ins Leben gerufen. Diese hat im April 2019 Empfehlungen für die Bewertung der Vertrauenswürdigkeit von KI-Anwendungen formuliert (Expert-Group 2019). Ein wichtiges Anliegen ist ihr, diskriminierende Handlungen und Informationsasymmetrien zwischen Menschen und dem KI-Algorithmus zu vermeiden und diese direkt beim



**Abb. 10.40** Unterschiedliche KI Richtlinien, welche in den letzten Jahren von Regierungen, Organisationen und Firmen vorgestellt wurden

Design des KI-Systems (Ethics-by-design) mitzudenken. Ausgehend von allgemeinen ethischen und rechtlichen Leitlinien werden Anforderungen und Umsetzungsmethoden vorgeschlagen. Das Dokument endet mit einer Liste von 60 konkreten Fragen, mit denen sich die Entwicklung, Einführung und Nutzung von KI-Systemen praktisch bewerten lassen soll. Die Fragen sind jedoch noch weit von einer Operationalisierung, etwa durch einen Prüfkatalog oder eine Norm, entfernt. Sie machen deutlich, wie herausfordernd es ist, unsere Wertevorstellungen und ethischen Grundsätze überprüfbar in Technologie umzusetzen.

### 10.5.1 Wie baut man vertrauenswürdige KI-Systeme?

Wir kennen alle die TÜV-Plakette von PKW, die alle zwei Jahre erneuert werden muss. Hier ist jedem PKW-Halter klar, dass er nicht ohne funktionierendes Licht, Blinker und Motor zur Kontrolle durch einen akkreditierten Prüfer fahren kann. Es sind noch einige technische Herausforderungen zu lösen, damit wir an eine Zulassung von selbstfahrenden Autos denken können. Ob in diesem Zusammenhang ein PKW-Modell nur einmalig vom TÜV überprüft wird, oder wie bei TESLA nach jedem Softwareupdate, ist noch nicht entschieden. Um Vertrauen in diese und andere KI-Anwendungen herzustellen, muss überprüfbar sein, dass sie die ethischen und rechtlichen Rahmenbedingungen in ihrem speziellen Anwendungskontext beachten und sicher und hinreichend zuverlässig funktionieren.

In einem Zertifizierungsverfahren kann man die Einhaltung standardisierter Anforderungen, die vom KI-Systemanbieter nachgewiesen wird, überprüfen und bestätigen. „Standardisiert“ heißt, dass ein Prüfkatalog die Anforderungen und die Kriterien der Prüfung genau festlegt und dem Gütesiegel so seine Bedeutung verleiht. Für einen solchen Katalog bilden die Fragen der High-Level Expert Group der EU sicher einen guten Orientierungsrahmen. Die Zertifizierung wird von unabhängiger Stelle, wie z. B. einem TÜV oder einer Dekra vergeben. Eine KI-Zertifizierung dient insofern dem allgemeinen Zweck, Unrecht – im Sinne von Rechtsverstößen – und ethisch nicht gerechtfertigte Folgen von der Gesellschaft abzuwenden. Mit einer in Europa erfolgten KI-Zertifizierung können wir einen Qualitäts- und Wertestandard „Made in Europe“ setzen, der den Menschen in den Mittelpunkt stellt, und sowohl bei Unternehmen als auch bei Nutzern und gesellschaftlichen Akteuren Vertrauen und Akzeptanz schafft (Abb. 10.41).

Umgekehrt muss jedoch vermieden werden, dass eine Überregulierung und zu hohe Zertifizierungsstandards innovationshemmend wirken oder aufgrund der Dynamik des technologischen Fortschritts zu schnell veralten und somit bald nicht mehr angemessen sind. Die Methoden und Anwendungsmöglichkeiten der Künstlichen Intelligenz werden kontinuierlich und so tiefgreifend weiterentwickelt, dass eine Zertifizierung dieser Dynamik gerecht werden muss. Angesichts technologischer Innovationen und Umbrüche ändern sich auch unsere Werte und Gesetze und damit die Leitplanken für die Zertifizierung von KI.

**Abb. 10.41** Aspekte der Zertifizierung von Systemen der Künstlichen Intelligenz. Wordcloud aus dem Text von (Cremers et al. 2019)



### 10.5.2 Wie kann man tiefe neuronale Netze testen?

Dabei bestehen auf technischer Seite einige gravierende Unterschiede zwischen einer klassischen Software und einem KI-System mit einem auf Daten trainierten Modell. Eine klassische Software ist modular und aus menschenlesbaren Anweisungen aufgebaut. Das gibt dem Softwareentwickler die Möglichkeit, die Software durch Komponenten-, Integrations- und Systemtests weitgehend automatisch zu validieren. Künstliche neuronale Netze dagegen haben Tausende bis Milliarden von Parametern, die anhand großer Datenmengen optimiert wurden. Für ein konkretes Ergebnis kann eine beliebige Wertekonstellation der Eingaben und Parameter verantwortlich sein, deren Interaktionen meist hochgradig komplex sind. Alle diese möglichen Kombinationen können nur schwer getestet werden. Sie können auch nicht auseinandergenommen und nach dem Prinzip „Teile und Herrsche“ separat getestet werden. Bei der traditionellen Software gibt es eben klar definierte Kontrollflüsse, bei Machine-Learning-Modellen sind es Datenflüsse.

Will man beispielsweise Knochenbrüche aus Röntgenaufnahmen prognostizieren, so kann es sein, dass eines der Krankenhäuser eine abweichende Einstellung bei der Röntgenanlage hat, welche oft fälschlicherweise Knochenbrüche diagnostiziert. Dann kann es dazu kommen, dass das Lernverfahren diese Anlage in den Daten erkennt und dann häufiger zu der Diagnose Knochenbruch kommt. Der Algorithmus macht also gute Prognosen, aber aus den falschen Gründen (Smith 2020a).

Tatsächlich ist eine unausgewogene Datenbasis eine der häufigsten Ursachen für eine nicht zufriedenstellende Anwendung. Hinzu kommt, dass KI-Anwendungen im Betrieb rechtzeitig nachtrainiert werden müssen oder sogar laufend selbstständig weiterlernen (Bestärkungslernen, Kap. 8). Nach solchen Änderungen muss das Verhalten der KI erneut geprüft werden. Dafür müssen entsprechende Prozesse aufgesetzt werden. Fehler machen keineswegs nur Data Scientists, wenn sie ihre ersten KI-Piloten bauen. Auch erfahrenen

KI-Ingenieuren bei Amazon, Google, Microsoft etc. sind Fehler unterlaufen. Im Nachhinein fragt man sich, ob das nicht früher hätte auffallen können. Aber die Forschung reagiert und der Stand der Kunst im maschinellen Lernen schreitet schnell voran. Ein Prüfkatalog muss deshalb regelmäßige Aktualisierung erfahren.

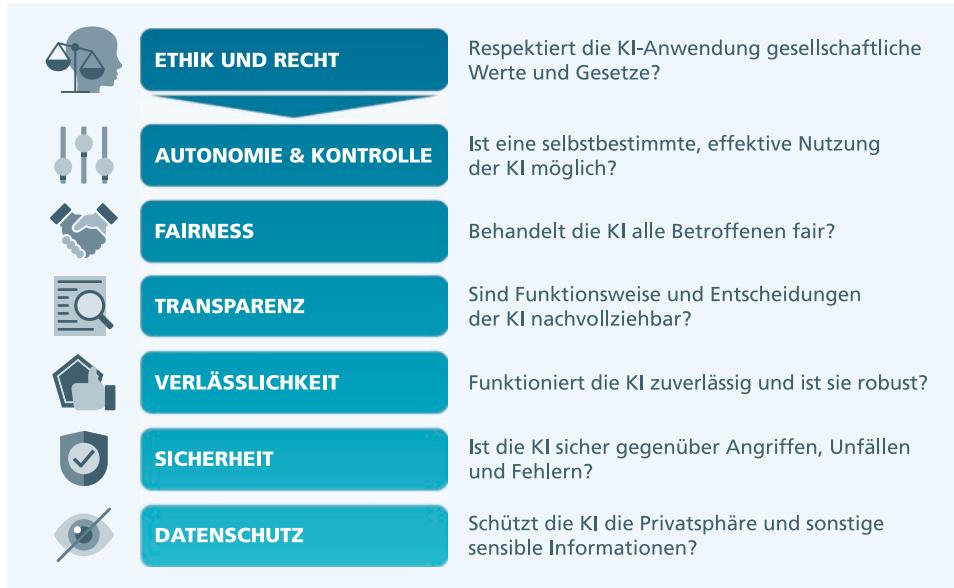
Eine eher profane Fehlerquelle bei TNN sind Störungen bei der Erfassung der Daten. Dies können beispielsweise Messfehler in den Sensoren oder aber auch Abweichungen in der Art der Eingaben zwischen der Trainingsmenge und der Testmenge sein. Wir haben gesehen, dass derartige Fehler die Performanz der TNN stark verringern können (Abschn. 5.12.1 und 7.9.1). Das Problem lässt sich reduzieren, indem man Eingaben mit Messfehlern in die Trainingsmenge übernimmt.

Ein weiterer wichtiger Problembereich von TNN sind Beispiele, die absichtlich so konstruiert wurden, um das TNN zu täuschen. Ein zum Erkennen von Personen trainiertes TNN könnte feststellen, dass Eigenschaften wie Farb-, Textur- oder Hintergrundflecken genauso starke Merkmale zur Personenerkennung sind wie die Dinge, die wir normalerweise als relevant erachten, wie beispielsweise Mund oder Augen. Dies bedeutet aber auch, dass eine sehr kleine Änderung der Eingabe in scheinbar „unwichtigen“ Bereichen dazu führen kann, dass das TNN etwas Anderes erkennt. So haben wir in Abschn. 5.12.2 gesehen, dass Verkehrsschilder durch kleine Aufkleber so abgeändert werden können, dass sie grundle- gen falsch klassifiziert werden. In gleicher Weise können Spracherkennungssysteme durch nahezu unhörbare Störsignale dazu gebracht werden, Spracheingaben vollständig falsch zu interpretieren (Abschn. 7.9.2). Schließlich sind auch Robotersteuerungen anfällig für kleine Manipulationen der Eingaben (Abschn. 8.6.2). Zwar gibt es Gegenmaßnahmen, diese wirken aber nur für bekannte Attacken.

In einem Prüfkatalog müssen Handlungsfelder aus ethisch-rechtlicher und technischer Sicht festgelegt werden. Für den sogenannten „Bonner Katalog“, den die Initiative KI.NRW erarbeiten lässt, wurden im Mai 2019 in einem White Paper die Bereiche Fairness, Transparenz, Autonomie und Kontrolle, Datenschutz sowie Sicherheit und Verlässlichkeit festgelegt, um ein Grundgerüst für eine KI-Zertifizierung zu schaffen (Cremers et al. 2019). Während der Punkt „Sicherheit“ die üblichen Aspekte der Betriebssicherheit umfasst, bezieht sich der Punkt „Verlässlichkeit“ auf die besonderen Prüfungsherausforderungen von komplexen KI-Modellen wie tiefen neuronalen Netzen. Abb. 10.42 zeigt die Handlungsfelder mit ihren Leitfragen an das KI-System.

### 10.5.3 Ist eine selbstbestimmte, effektive Nutzung eines KI-Systems möglich?

Ein KI-System darf die Autonomie von Individuen und sozialen Gruppen nicht einschränken. Vor diesem Hintergrund ist bei Design, Entwicklung und Betrieb einer KI-Anwendung zu überlegen bzw. zu testen, inwiefern Nutzer übermäßiges Vertrauen in die Anwendung entwickeln, emotionale Bindungen aufzubauen (Abschn. 9.4) oder in ihrer Entscheidungsfindung unzulässig beeinträchtigt oder gelenkt werden könnten.



**Abb. 10.42** Handlungsfelder des Bonner KI-Prüfkatalogs. Für jedes dieser Handlungsfelder werden eine Reihe von Fragen und Anforderungen an ein KI-System formuliert

Zum Beispiel werden durch „User Intent Recognition“ in Echtzeit menschliche Emotionen und Absichten erkannt und können ggf. direkt ausgenutzt werden. Personalisierte Filtersysteme erzeugen Filterblasen, die durch die Bereitstellung einseitiger Nachrichten und Informationen die Objektivität und Kritikfähigkeit ihrer Nutzer einschränken können. KI-Anwendungen im medizinischen Kontext legen keinen hippokratischen Eid ab. Solange ausreichend Zeit für ein Urteil durch den Arzt möglich ist, dürfen KI-Systeme nach dem aktuellen Verständnis daher nicht alleine über die Behandlung eines Patienten entscheiden und den Arzt auch nicht zu einer unkritischen Übernahme ihrer Vorschläge verleiten. Anwendungen in der Finanzwirtschaft, bei denen Bots auf Kursentwicklungen in Sekundenschnelle reagieren, zeigen, dass es sichere Eingriffsmöglichkeiten in Notfällen geben muss. Gibt es ein Not-Aus für ein solches KI-System, und wie erkennt der Nutzer eine Fehlentwicklung?

In zeitkritischen Einsatzbereichen, wie dem autonomen Fahren (Abschn. 8.5), fehlt oft die Zeit, den Menschen einzubeziehen. Dabei ist wichtig, dass ein KI-System nur unter den Bedingungen operiert, für die es trainiert wurde. Treten ungewohnte Situationen auf, muss das Gesamtsystem in einen sicheren Zustand überführt werden. Ein solcher Fall könnte z. B. eintreten, wenn eine Straße schneedeckt ist, aber die Trainingsdaten nur aus schneefreien Tagen stammten. Das System muss dann in der Lage sein, das Auto zügig am Straßenrand abzustellen. Wir müssen also die Möglichkeiten und die Grenzen der Maschine verstehen (Abschn. 5.11, 5.12, 6.11, 7.9, 8.6). Eine Fehlerfreiheit kann in den seltensten Fällen garantiert werden, auch beim vollständig autonomen Fahren müssen wir diese Frage diskutieren.

#### 10.5.4 Behandelt das KI-System alle Betroffenen fair?

Die Diskriminierung schützenswerter Personen ist ein Risiko, an das beim Einsatz von KI gedacht werden muss. Durch Dynamic Pricing und automatische Bewertungen können bestimmte Bevölkerungsgruppen benachteiligt werden. Das muss gar nicht explizit geschehen. Es gibt viele Fälle, in denen die zugrundeliegenden Daten die Ursache sind. Das zeigt ein inzwischen prominentes Beispiel von Amazon. Ziel war es, im Recruiting die Bewertung von Kandidaten zu automatisieren und für gleichbleibende Kriterien zu sorgen. Leider wurden bei der eingesetzten Software eindeutig Männer bevorzugt. Das Problem lag in den historischen Trainingsdaten des Unternehmens. Da in den vergangenen Jahren mehr Männer zum persönlichen Bewerbungsgespräch eingeladen wurden, traten in dem Modell Merkmale in den Vordergrund, die klassischerweise eher Männern zuzuordnen sind. Zudem wurde die Qualifikation einer Person danach bewertet, ob sie eingestellt wurde oder nicht, und nicht danach, ob sie später im Job erfolgreich war. Damit wurden die Präferenzen der einstellenden Manager in den trainierten Algorithmus übernommen. Weiterhin waren Frauen und People of Color in der Stichprobe stark unterrepräsentiert.

Verbesserungsversuche scheiterten, und 2017 wurde das System aufgegeben. Man muss sich also immer fragen, ob die Trainingsdaten die erforderliche Grundgesamtheit und schutzbedürftige Subgruppen ohne Verzerrung widerspiegeln und ob die Qualität der Daten überall gleich gut ist. Aber selbst dann kann es im Betrieb passieren, dass sich die Umwelt ändert. Dann muss mit aktualisierten Daten nachtrainiert werden. Das bedeutet auch, dass man gut überlegen muss, ab wann eine KI in den produktiven Einsatz geht, wie viel Autonomie ihr zugestanden wird und wie häufig das System überprüft wird.

Mit Fairness ist das grundsätzliche Gebot verbunden, gleiche soziale Sachverhalte gleich zu behandeln (Abb. 10.43), es sei denn, ein abweichendes Vorgehen wäre sachlich gerechtfertigt. Dies bedeutet insbesondere, dass Individuen nicht aufgrund ihrer Zugehörigkeit zu einer bestimmten sozialen Gruppe, ihrer Religion oder ihres Geschlechts diskriminiert oder bessergestellt werden dürfen. Welche sozialen Gruppen geschützt sind und welche nicht, muss der Gesetzgeber oder der gesellschaftliche Konsens festlegen. KI-Anwendungen lernen aus historischen Daten. Diese sind nicht notwendigerweise vorurteilsfrei. Beinhaltet die Daten beispielsweise Benachteiligungen von Frauen, so wird die KI-Komponente diese Vorurteile übernehmen. Außerdem können in der Datengrundlage bestimmte Gruppen unterrepräsentiert sein. Man spricht dann von Bias, einer Verzerrung in den Daten. Bias kann ebenfalls zu Entscheidungen führen, die unfair sind.

Um diesem Sachverhalt zu begegnen, muss Fairness operationalisiert und messbar gemacht werden. Zunächst muss man diejenigen Gruppen identifizieren, die nicht benachteiligt werden sollen. Für alle identifizierten Gruppen sollen die Ergebnisse vergleichbar sein. Vergleichbar kann „gleiche Trefferwahrscheinlichkeit“ bedeuten, es gibt aber auch viele weitere Konzepte. Außerdem kann auch ohne Berücksichtigung von Gruppen auf individuelle Fairness gesetzt werden, also auf Gleichbehandlung gleichartiger Individuen. Welche Definition für den konkreten Anwendungsfall die richtige ist, muss kontextabhängig entschieden und aus Transparenzgründen dem Nutzer mitgeteilt werden.



**Abb. 10.43** Die von einem KI-System getroffenen Entscheidungen sollten in ihren Auswirkungen fair sein, d. h. keine Betroffenen ungerechtfertigt benachteiligen. Bildnachweis im Anhang

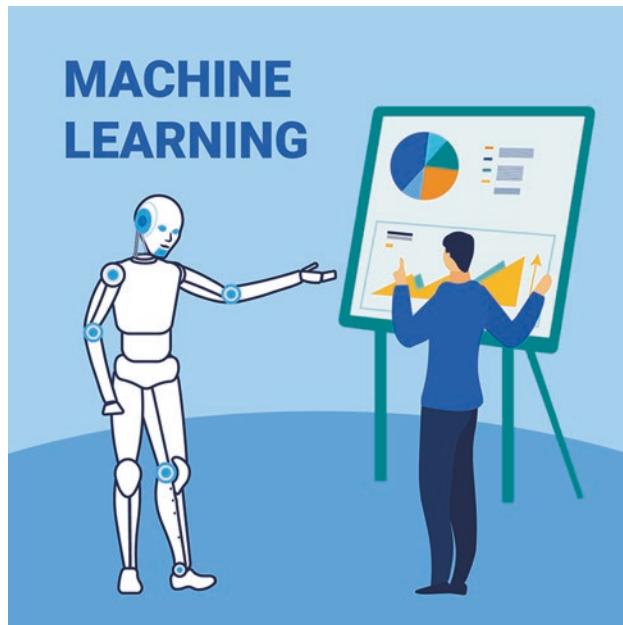
### 10.5.5 Sind Funktionsweise und Entscheidungen der KI nachvollziehbar?

Die Transparenz einer KI-Anwendung kann in vielen Einsatzgebieten für ihre Akzeptanz entscheidend sein. Dabei sind zwei Aspekte zu unterscheiden. Erstens müssen Informationen zum richtigen Umgang mit der KI-Anwendung verfügbar sein. Zweitens geht es um Anforderungen an die Interpretierbarkeit, Nachverfolgbarkeit und Reproduzierbarkeit von Ergebnissen, die Einsichten in die Funktionsweise der KI-Anwendung erfordern (Abb. 10.44).

Jedem Nutzer muss grundsätzlich klar sein, wann er mit einer KI-Anwendung kommuniziert. Dafür muss geklärt werden, was eine KI-Anwendung ist. Darüber hinaus müssen die Akteure angemessen mit dem Gebrauch der Anwendung vertraut gemacht werden. Dazu gehört ein Verständnis dafür, welchem Zweck die Anwendung dient, was sie leistet und welche potenziellen Risiken sie hat. Wird die KI-Anwendung in Situationen eingesetzt, für die sie nicht trainiert wurde, kann das negative Folgen haben. Das heißt, KI-Anwendungen müssen automatisch ihre Zuverlässigkeit überwachen (Abschn. 5.11).

Die zweite Art der Transparenz betrifft die inneren Prozesse der KI-Anwendung und speziell des ML-Modells. Nutzer möchten wissen, wie sicher ein Ergebnis ist, welche

**Abb. 10.44** Ein KI-System sollte das Zustandekommen seiner Ergebnisse so erklären können, dass die Grundzüge des Resultats nachvollziehbar sind. Bildnachweis im Anhang



Alternativen es gibt oder was für ein anderes Ergebnis gefehlt hat. Entwickler möchten genau nachvollziehen, wie ein Ergebnis zustande kam und im Schadensfall möchte man herausfinden, warum sich eine falsche Prognose ergab. Zwischen höherer Genauigkeit bzw. Robustheit einerseits und der Erklärbarkeit von Modellen andererseits gibt es ein Spannungsfeld, und KI-Ingenieure kennen das Dilemma: Black-Box-Modelle sind zwar in vielen Fällen genauer bzw. robuster als beispielsweise regelbasierte Modelle, jedoch sind sie nur bedingt interpretierbar. Allerdings kann Erklärbarkeit teilweise durch nachgeschaltete Verfahren, wie lokale Erklärungsmodelle oder eine Analyse des Eingabe/Ausgabe-Verhaltens von Modellen (Abschn. 6.10), erreicht werden. Die Interpretierbarkeit von Modellen ist ein aktives Forschungsfeld, und es werden viele Anstrengungen unternommen, die Lernprozesse von Black-Box-Modellen besser zu verstehen sowie ihre internen Prozesse zu visualisieren und die resultierenden Entscheidungen zu plausibilisieren.

### 10.5.6 Ist das KI-System sicher gegenüber Angriffen, Unfällen und Fehlern?

Sicherheit im Sinne von Schutz vor Angriffen (Security, → adversale Angriffe in 5.12.2) und Schutz vor Gefährdungen, die von der Anwendung selbst ausgehen (Safety), ist für KI-Anwendungen mindestens von ebenso großer Wichtigkeit wie für andere Informations- und technische Systeme. Beide Sicherheitskonzepte betreffen die gesamte Anwen-

dung, in die die KI-Komponente eingebettet ist. Die Gefährdungen können sich als Funktionsausfall oder starke Funktionsänderung der KI-Komponente sowie als unautorisierte Informationsabfluss äußern. Gefährdungen müssen vom System so früh wie möglich erkannt werden. Die KI-Komponente ist dann zu umgehen. Stattdessen kann die Kontrolle an den Nutzer übergeben werden oder das System wird von konventionell programmierten Modulen in einen sicheren Zustand überführt. Im selbstfahrenden Auto z. B. muss bei schlechter Sicht, Schneefall und starkem Verkehr der Autopilot deaktiviert werden, wenn unzureichendes Trainingsmaterial für diese Situationen zur Verfügung stand (Abschn. 8.5.3). Der Fahrer wird informiert und übernimmt selbst die Kontrolle.

### 10.5.7 Funktioniert die KI-Komponente zuverlässig und ist sie robust?

Die herkömmliche Prüfung der Sicherheit einer Anwendung muss durch spezielle Prüfungen der KI-Komponente ergänzt werden, und besonders des Modells, das aus den Daten maschinell gelernt wurde. Hier kann man sehr unterschiedliche Qualitätskriterien untersuchen: Die Güte oder Genauigkeit des Modells (Abschn. 3.5), die Einschätzung der ML-Modellunsicherheiten (Abschn. 5.11), die Robustheit gegenüber Rauschen oder schädlichen Eingaben (Abschn. 5.12, 6.11, 7.9, 8.6), aber auch der Ressourcenverbrauch und Antwortzeiten. Neuartige, für Menschen untypische und damit unerwartete Fehlermodi können zu potenziell kritischen, da nicht eingeübten Situationen führen, insbesondere bei direkter Mensch-Maschine-Interaktion.

Zur Verlässlichkeit gehört es auch, dass mögliche Daten- oder Modellmanipulation vermieden bzw. frühzeitig aufgedeckt werden. Falls solche Täuschungsversuche (Abb. 10.45) unternommen werden, muss dies erkannt und in schneller Weise auf eine fehlerfreie Ursprungsversion zurückgegriffen werden können.



**Abb. 10.45** Systeme der Künstlichen Intelligenz müssen robust sein gegenüber Angriffsversuchen und böswilligen Täuschungen. Bildnachweis im Anhang

Eine korrekte Implementierung der Trainingsroutinen und der Anwendung des fertig trainierten Modells ist eine weitere Anforderung an eine professionelle Software. Die dazu durchzuführenden Tests sollten im Bereich des maschinellen Lernens etabliert und auf die jeweilige Anwendung abgestimmt sein. Werden Modellschwächen aufgedeckt, ist mit geeigneten Korrekturmechanismen bis hin zum Einsatz eines Rückfallplans darauf zu reagieren. Ein überzeugendes Beispiel ist hier der Test von selbstfahrenden Autos durch ausgedehnte, realistische Simulationen bei Waymo (Abschn. 8.5.3). Auch im Produktivbetrieb ist die Verlässlichkeit der KI-Anwendung zu jedem Zeitpunkt zu gewährleisten. Dies impliziert, dass die Funktionsweise in angemessenen Abständen überprüft werden muss.

Wenn KI-Systeme richtig implementiert sind, dann tragen sie in hohem Maße dazu bei, eine Objektivität und Neutralität in die datengetriebene, unvoreingenommene Entscheidungsfindung mit einzubringen, da sie eine gleichbleibende Qualität liefern. So ist es nicht von der Hand zu weisen, dass der Mensch auch sehr häufig subjektive und vorurteilsbehafte Entscheidungen trifft. Beispielsweise wurde bei der Untersuchung von über 1000 Gerichtsurteilen in den USA festgestellt, dass die Wahrscheinlichkeit einer positiven Entscheidung zu Beginn des Arbeitstages oder nach einer Essenspause größer ist als später in der Reihenfolge der Fälle, und dieser Effekt war statistisch hochsignifikant. Aus Sicht des Angeklagten ist es also von großem Vorteil, zu Beginn der Sitzung (d. h. entweder zu Beginn des Tages oder unmittelbar nach der Pause) zu erscheinen (Danziger et al. 2011).

### 10.5.8 Schützt die KI die Privatsphäre und sonstige sensible Informationen?

KI-Anwendungen sind geeignet, in eine Vielzahl an Rechtspositionen einzugreifen (Abb. 10.46). Besonders häufig handelt es sich dabei um Eingriffe in die Privatsphäre bzw. das Recht auf informationelle Selbstbestimmung. So verarbeiten KI-Anwendungen oftmals sensible Informationen, personenbezogene oder persönliche Daten, etwa Stimmabnahmen, Fotos oder Videos. Daher ist sicherzustellen, dass die einschlägigen datenschutzrechtlichen Bestimmungen, wie etwa die Datenschutzgrundverordnung (DSGVO) und das Bundesdatenschutzgesetz (BDSG), eingehalten werden.

KI-Anwendungen können nicht nur ein Risiko für die Privatsphäre des Einzelnen darstellen. Aus ihren Modellen und den Nutzungsdaten könnten sich auch Geschäftsgeheimnisse entlocken lassen, die ethisch oder rechtlich schutzwürdig sind. Ein Beispiel sind Maschinendaten, die völlig unabhängig von der Frage, welche Person die Maschine bedient hat, Informationen über die Prozessauslastung oder Fehlerquoten beinhalten.

Ein KI-System wird durch Daten von Einzelfällen trainiert, und die geschätzten Parameter können als eine Art „Mittelwerte“ über die Einzelfälle betrachtet werden. Allein die Parameter werden bei einer Prognose verwendet. Allerdings ist es in gewissen Fällen möglich, aus diesen Parameterwerten wieder zurück auf die Einzelfälle zu schließen. Handelt es sich bei den Einzelfällen um Personen, so ist dann nach den Datenschutzbestimmungen die unbeschränkte Nutzung eines solchen Modells nicht erlaubt.



**Abb. 10.46** Gegen die Tendenz der Internetwirtschaft, alle erreichbaren Daten und Medien des Nutzers mithilfe von KI zu erfassen und zu monetarisieren, muss es einen wirksamen Schutz geben. Bildnachweis im Anhang

Differential Privacy (differentielle Privatsphäre) hat das Ziel, die Aussagekraft von neuronalen Netzen unter Minimierung der Wahrscheinlichkeit zu maximieren, dass man die zum Training verwendeten Einzelfälle identifizieren kann. Im Rahmen des maschinel- len Lernens wurden entsprechende Algorithmen entwickelt, die mittlerweile auch in TensorFlow, einer Toolbox für TNN, verfügbar sind (Galen 2019). Die Intuition für die Definition der Differential Privacy ist, dass die Privatsphäre einer Person durch eine statistische Veröffentlichung nicht gefährdet werden kann, wenn sich ihre Daten nicht in der Datenbank befinden. Aus diesem Grund ist es das Ziel, mit Differential Privacy jedem Einzelnen ungefähr den gleichen Datenschutz zu bieten, der sich aus der Entfernung seiner Daten aus der Trainingsmenge ergeben würde. Das heißt, das TNN sollte nicht übermäßig von den Daten einer einzelnen Person abhängen. Man kann so die Wahrscheinlichkeit der Gefährdung einer Person sehr stark herabsetzen und erhält für das trainierte Modell starke und robuste Garantien dieser Eigenschaft.

### 10.5.9 Die Herausforderungen für ein KI-Gütesiegel

Ein Gütesiegel für menschenzentrierte KI, das im gesellschaftlichen Konsens entstanden ist und Klarheit über die Eigenschaften einer Anwendung in den aufgelisteten Handlungs-

feldern verschafft, ist sicherlich mehr als wünschenswert. Dabei wird der Prüfer immer Ermessensspielräume haben. Umso wichtiger ist es, dass die Prüfer eine ausreichende Kompetenz vorweisen können. Hierzu sind Akkreditierungsstandards zu entwickeln. Außerdem kann ein Grundkatalog Mindestanforderungen nur allgemein formulieren. Sie müssen branchen- oder anwendungsbezogen ausdifferenziert werden. So wird ein Prüfkatalog für den medizinischen Kontext einen anderen Fokus legen müssen als für das autonome Fahren oder für industrielle Anwendungen.

Ein KI-Gütesiegel sollte als lebendes Prüfzertifikat verstanden werden, das sich parallel zur Technologie, unserem Wissen und unseren Werten weiterentwickelt. Wir sollten ständig überwachen, dass das geltende Recht sowie unsere Prinzipien und Werte respektiert werden und die eingesetzte Technologie robust und zuverlässig ist. Um dies sicherzustellen, ist ein disziplinübergreifender Austausch über Künstliche Intelligenz zwischen Informatik, Rechtswissenschaften, Philosophie und Politik notwendig.

Ein Problem, dem eine Zertifizierung nicht völlig begegnen kann, sind die Möglichkeiten des Dual Use. Eine KI-Anwendung kann sowohl für positive, wie auch für negative Zwecke eingesetzt werden. Dieser Risiken sollte sich jedes Unternehmen und jeder Entwickler bewusst sein und sie nach Möglichkeit reduzieren. Anwendungen sollten „by-design“ so konstruiert werden, dass sie in dem festgelegten Umfang prüfbar und nur schwer über den eigentlichen Zweck hinaus anwendbar sind. Die Maxime sollte lauten, Anwendungen zu entwickeln, die die Grundrechte des Menschen by-design bewahren und bei denen möglichst keine Menschen zu Schaden kommen. Unabhängig von einer konkreten Anwendung ist es hilfreich, im Unternehmen einen KI-Ethikkodex zu formulieren und bei der Zusammensetzung von KI-Teams auf Vielfalt und Heterogenität zu achten.

---

## 10.6 Zusammenfassung

Die durch Big Data und KI bevorstehenden Veränderungen sind fundamental und wahrscheinlich unumkehrbar. Sie haben Auswirkungen auf den Einzelnen und die Gesellschaft: das soziale Gefüge, den Wert und die Gestaltung der Arbeit, die Mensch-Maschine-Interaktion und die freie politische Willens- und Meinungsbildung.

An drei Einsatzgebieten wurden die Möglichkeiten und der Einfluss der Künstlichen Intelligenz aufgezeigt. Dabei ist beeindruckend, wie sehr sich die KI in unser Leben integrieren wird. Ob im eigenen Haushalt, im smarten Krankenhaus der Zukunft oder in einer der klassischen deutschen Domänen, der ingenieurwissenschaftlich geprägten verarbeitenden Industrie, wird KI in vielfältiger Weise zum Einsatz kommen.

Dabei sind Daten ein wichtiges Asset geworden, der Rohstoff der Digitalisierung. Insbesondere amerikanische Konzerne haben sich in den vergangenen Jahren enorme Datensmengen verschafft und treiben deren Monetarisierung ständig mit großem Erfolg voran. Mithilfe von Plattformen und Portalen schieben sie sich als Intermediäre zwischen Nutzer und traditionelle Anbieter, welche Daten erfassen und Leistungen vermitteln.

In der globalen Plattformökonomie ist es enorm wichtig, europäische Alternativen zu schaffen. Google wurde hier als Beispiel angeführt: Ursprünglich als Suchmaschine gestartet, hat das Unternehmen in zahlreiche Branchen expandiert. Hält der aktuelle Trend an, wird Deutschland zum reinen Nutzer oder Zulieferer amerikanischer und chinesischer Plattformen degradiert, zum Beispiel für Mobilitätsdienste, Sprachdienste und andere intelligente Fähigkeiten unserer Maschinen. Das könnte eine massive Einschränkung unserer staatlichen Souveränität bedeuten, da KI-Services und die damit einhergehenden zentralen Wertschöpfungen der Zukunft im Ausland gesteuert werden. Damit wir das Potential der KI nutzen und sie nach unseren Werten gestalten, müssen wir in Europa in allen Bereichen der KI-Anwendung eine aktive Rolle spielen. Insbesondere bei der industriellen Analytik bietet sich für Deutschland großes Potential. In diesem sich schnell verändernden Umfeld sollten wir Europäer eine Vorreiterrolle einnehmen und unsere moralischen und ethischen Vorstellungen in intelligenten Systemen umsetzen.

Prognosen zu den Auswirkungen auf den Arbeitsmarkt sind derzeit noch sehr schwierig. Es ist jedoch zu erkennen, dass neben der weiteren Automatisierung manueller Arbeit auch die Denkarbeit von gut ausgebildeten Fachkräften ein erhebliches Substituierungspotential durch KI besitzt. KI wird viele Routinetätigkeiten übernehmen und bei anspruchsvoller Tätigkeiten assistieren. Analytisches Denken, Kreativität, und die Fähigkeit, Probleme im Team zu lösen, sind Schlüsselkompetenzen der Zukunft, welche nur unzureichend von KI-Systemen erbracht werden. Daher müssen entsprechende Bildungsangebote modernisiert werden. Ganz massiv werden zukünftig Data Scientists gesucht – Gestalter der KI, die in der Lage sind, oftmals außerhalb oder an den Rändern der traditionellen Prozesse zu denken und Ideen umzusetzen, die bestehende Strukturen in Frage stellen.

Die Frage nach den Grenzen von KI-Systemen hinsichtlich ihrer Robustheit und ihrer Anfälligkeit gegen böswillige Angriffe haben wir in den vorherigen Kapiteln diskutiert. Ein KI-System hat keine moralischen oder ethischen Bedenken. Sein Verhalten folgt der vorgegebenen Verlust- oder Belohnungsfunktion und gelernten Mustern in Daten, die möglicherweise nicht repräsentativ sind, eine unzureichende Qualität aufweisen oder historische Bevorzugungen und Diskriminierungen enthalten. Beim Einsatzgebiet der Medizin zeigt sich ganz deutlich, dass schlussendlich der Mensch gefragt ist, Alternativen abzuwägen und Entscheidungen zu treffen (Human in the Loop).

Für den vertrauensvollen Einsatz von KI brauchen wir einen gesellschaftlichen Dialog über das Verhältnis von Mensch und Maschine sowie ein verlässliches ethisch-rechtliches Fundament. Mit einem Gütesiegel kann eine Qualitätsmarke „KI Made in Germany/Europa“ etabliert werden, die verantwortlich gestaltete, zuverlässig und sicher realisierte Anwendungen erkennbar macht und nachhaltig schützt. Ein Gütesiegel leistet einen Beitrag zur Akzeptanz von Künstlicher Intelligenz in der Gesellschaft.

## Literatur

- Andreessen, Marc.** 2011. Why Software Is Eating The World. *The Wall Street Journal*. 2011, 20.08.2011.
- Ärzteblatt.** 2018. GKV-Spitzenverband nimmt Exoskelett ins Hilfsmittelverzeichnis auf. *aerzteblatt.de*. [Online] 01. 02. 2018. [Zitat vom: 13. 01. 2020.] <https://www.aerzteblatt.de/nachrichten/89000/GKV-Spitzenverband-nimmt-Exoskelett-ins-Hilfsmittelverzeichnis-auf>.
- Bastian, M.** 2018. 600 Millionen US-Dollar Investition: Sensetime ist Chinas wertvollstes KI-Unternehmen. *mixed.de*. [Online] 09. 04. 2018. [Zitat vom: 28. 01. 2020.] <https://mixed.de/600-millionen-us-dollar-investition-sensetime-ist-chinas-wertvollstes-ki-unternehmen/>.
- Bate, Alex.** 2016. ALPHA vs. The Pro – Judgement Day. [Online] 13. 07. 2016. [Zitat vom: 14. 01. 2020.] <https://www.raspberrypi.org/blog/alpha-vs-pro-judgement-day/>.
- BearingPoint.** 2017. Studie Smarte Gesundheit 2017. [Online] 2017. [Zitat vom: 13. 01. 2020.] <https://www.bearingpoint.com/de-de/downloadformular/?item=8551&module=474592>.
- Bengio, Yoshua.** 2019. From System 1 Deep Learning to System 2 Deep Learning. *NeurIPS'2019 Keynote, December 11th, 2019, Vancouver BC*. 2019.
- Biermann, K.** 2019. Deutschland will zurückhauen. *Zeit Online*. [Online] 12. 07. 2019. [Zitat vom: 14. 01. 2020.] <https://www.zeit.de/digital/internet/2019-07/hackback-cyberwar-datensicherheit-digitaler-angriff-bundesregierung>.
- Biermann, Kai.** 2015. BND speichert 220 Millionen Telefondaten – jeden Tag. *Zeit Online*. 2015, 30.01.2015.
- BMBF.** 2018. Strategie Künstliche Intelligenz der Bundesregierung. *Bundesministerium für Bildung und Forschung*. [Online] 2018. [Zitat vom: 14. 01. 2020.] [https://www.bmbf.de/files/Nationale\\_KI-Strategie.pdf](https://www.bmbf.de/files/Nationale_KI-Strategie.pdf).
- Brinker, T. J., 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. 2019.
- Bryant, Meg.** 2019. How AI and machine learning are changing prosthetics. [Online] 2019. [Zitat vom: 25. 09. 2019.] <https://www.medtechdive.com/news/how-ai-and-machine-learning-are-changing-prosthetics/550788/>.
- Cadwalladr, Carole und Graham-Harrison, Emma.** 2018. How Cambridge Analytica turned Facebook ‘likes’ into a lucrative political tool. *The Guardian*. 2018, 17.03.2018.
- Castro, Daniel, McLaughlin, Michael und Chivot, Eline.** 2019. Who Is Winning the AI Race: China, the EU or the United States? *Center for Data Innovation*. Washington DC, 2019.
- Cremers, A, et al.** 2019. Vertrauenswürdiger Einsatz von Künstlicher Intelligenz. *Fraunhofer Institut für Intelligente Analyse- und Informationssysteme, Sankt Augustin*. 2019.
- Dachwitz, Ingo, Rudl, Tomas und Rebiger, Simon.** 2018. Was wir über den Skandal um Facebook und Cambridge Analytica wissen. *Netzpolitik.org*. [Online] 21. 03. 2018. [Zitat vom: 12. 04. 2020.] <https://netzpolitik.org/2018/cambridge-analytica-was-wir ueber-das-groesste-datenleck-in-der-geschichte-von-facebook-wissen>.
- Danziger, S., Levav, J. und Avnaim-Pesso, L.** 2011. Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences of the United States of America, PNAS April 26, 2011 108 (17) 6889–6892*: . 2011, <https://www.pnas.org/content/108/1>.
- Davenport, T. und Patil, D.J.** 2012. Data Scientist: The Sexiest Job of the 21st Century. *Harvard Business Review*. 2012, Oct. 2012.
- Deuber, Lea.** 2019. Überwacht, überall – Wer in der Volksrepublik eine neue Handynummer will, muss künftig vorher sein Gesicht scannen lassen. [Online] 18. 10. 2019. [Zitat vom: 14. 01. 2020.] <https://www.sueddeutsche.de/politik/china ueberwacht ueberall-1.4646259>.

- Diamond, A.** 2013. Executive functions. *Annu Rev Psychol.* 64: 135–168. 2013.
- Dorloff, Axel.** 2018. Mit Gesichtserkennung in Richtung Massenüberwachung. [Online] 2018. [https://www.deutschlandfunk.de/china-mit-gesichtserkennung-in-richtung-massenueberwachung.1773.de.html?dram:article\\_id=415748](https://www.deutschlandfunk.de/china-mit-gesichtserkennung-in-richtung-massenueberwachung.1773.de.html?dram:article_id=415748).
- DW.** 2019. Deutsche Wikipedia-Seite kurzzeitig gehackt. *Deutsche Welle*. [Online] 2019. [Zitat vom: 13. 01. 2020.] <https://www.dw.com/de/deutsche-wikipedia-seite-kurzzeitig-gehackt/a-50335577>.
- Dyrmann, M., Jørgensen, R. N. und 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. 2017.
- Erxleben, F., et al.** 2014. Introducing Wikidata to the linked data web. In *International Semantic Web Conference* (pp. 50–65). Springer, Cham. 2014.
- Europa.** 2017. Entschließung des Europäischen Parlaments vom 16. Februar 2017 mit Empfehlungen an die Kommission zu zivilrechtlichen Regelungen im Bereich Robotik (2015/2103(INL)). *Europäisches Parlament*. [Online] 16. 02. 2017. [Zitat vom: 14. 01. 2020.] [http://www.europarl.europa.eu/doceo/document/TA-8-2017-0051\\_DE.pdf](http://www.europarl.europa.eu/doceo/document/TA-8-2017-0051_DE.pdf).
- Expert-Group.** 2019. Ethics guidelines for trustworthy AI. *Report of High-Level Expert Group on AI*. [Online] 08. 04. 2019. [Zitat vom: 14. 01. 2020.] <https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai>.
- Fischer, M., et al.** 2019. DL2: Training and querying neural networks with logic. *Proceedings of the 36 th International Conference on Machine Learning*. 2019.
- Galen, A.** 2019. Tensorflow Privacy. [Online] 2019. [Zitat vom: 10. 09. 2019.] <https://github.com/tensorflow/privacy>.
- García-Ajofrín, Lola.** 2018. Paying for Fried Chicken with the Face: A Journey around the Biometric Landscape in China. [Online] 04. 12. 2018. [Zitat vom: 14. 01. 2020.] <https://outride.rs/en/paying-for-fried-chicken-with-the-face-a-journey-around-the-biometric-landscape-in-china/>.
- Görz, G., Schmid, U., Schneeberger, J. und Braun, T.** 2020. *Handbuch der Künstlichen Intelligenz*, 6. Auflage. s.l. : De Gruyter, 2020. 978-3-11-021808-4.
- Grävemeyer, Arne.** 2019. Künstlich intelligente Diagnose als zweite Meinung. ct. 2019, 18.10.2019.
- Gromann, D., Luis Espinosa, A. und Declereck, T.** 2019. Special issue on Semantic Deep Learning. *Semantic Web* 10 (2019) 815–822, DOI 10.3233/SW-190364. 2019.
- Guardian.** 2019. Paralysed man walks using mind-controlled exoskeleton. *Guardian*. [Online] 2019. [Zitat vom: 13. 01. 2020.] <https://www.theguardian.com/world/2019/oct/04/paralysed-man-walks-using-mind-controlled-exoskeleton>.
- Hamrick, J. B.** 2019. Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences*, 29, 8–16. 2019.
- Hecker, D., et al.** 2016. Big-Data-Geschäftsmodelle – die drei Seiten der Medaille, . In: *Wirtschaftsinformatik & Management* 8 (6), 2016, 20–30. 2016.
- Hecker, D., et al.** 2017b. Künstliche Intelligenz und die Potenziale des maschinellen Lernens für die Industrie. *Wirtschaftsinformatik & Management*. October 2017, Volume 9, Issue 5, pp 26–35, 2017.
- Hecker, Dirk, et al.** 2017a. Zukunftsmarkt Künstliche Intelligenz. Potenziale und Anwendungen. St. Augustin: Fraunhofer-Allianz Big Data, 2017, 60 S. 2017.
- Hossain, M. T., et al.** 2018. Distortion robust image classification with deep convolutional neural network based on discrete cosine transform. [Online] 2018. [Zitat vom: 30. 05. 2019.] arXiv 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. [Online] 2019. [Zitat vom: 13. 01. 2020.] [http://www.i-prognosis.eu/?page\\_id=59](http://www.i-prognosis.eu/?page_id=59).
- Ito, Joi.** 2018. Why Westerners Fear Robots and the Japanese Do Not. *Wired*. [Online] 30. 07. 2018. [Zitat vom: 13. 01. 2020.] <https://www.wired.com/story/ideas-joi-ito-robot-overlords/>.

- Jordan, M.** 2018. Artificial Intelligence — The Revolution Hasn't Happened Yet. [Online] medium. com 2018. <https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7>.
- Kahneman, Daniel.** 2011. *Thinking, Fast and Slow*. s.l. : Macmillan. ISBN 978-1-4299-6935-2. 2011.
- Kanellos, M.** 2016. 152,000 Smart Devices Every Minute. In 2025: IDC Outlines The Future of Smart Things. *Forbes*. 2016.
- KillerRobots.** 2018. Campaign to stop Killer Robots. [Online] 13. 04 2018. [Zitat vom: 14. 01. 2020.] <https://www.stopkillerrobots.org/>
- Kosinski, M., Stillwell, D. und Graepel, T.** 2013. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the national academy of sciences*, 110(15), 5802–5805. 2013.
- Kropp, P., Theuer, S. und Fritzsche, B.** 2018. Immer mehr Tätigkeiten werden durch Digitalisierung ersetzbar. *IAB* 2018, S SN 1861-14 35, S.32. 2018.
- Kruppe, T., et al.** 2019. Digitalisierung: Herausforderungen für die Aus- und Weiterbildung in Deutschland. *Instituts für Arbeitsmarkt-und Berufsforschung*. [Online] 2019. [Zitat vom: 28. 01. 2020.] <http://doku.iab.de/stellungnahme/2019/sn0119.pdf>.
- Lepies, J.** 2017. Künstliche Intelligenz in der Medizin: „Wir wollen Ärzte nicht arbeitslos machen“. *heise online*. [Online] 07. 09. 2017. [Zitat vom: 17. 01. 2020.] <https://www.heise.de/newsticker/meldung/Kuenstliche-Intelligenz-in-der-Medizin-Wir-wollen-Aerzte-nicht-arbeitslos-machen-3824121.html>.
- Leviathan, Y.** 2018. Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone. [Online] 2018. [Zitat vom: 28. 04. 2019.] <https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html>.
- Lewis, Nell.** 2019. Why robots will soon be picking soft fruits and salad. [Online] 04. 09. 2019. [Zitat vom: 25. 09. 2019.] <https://edition.cnn.com/2019/09/04/business/robot-farmers/index.html>.
- Manhaeve, R., et al.** 2018. Deepprobabilog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems* (pp. 3749–3759). 2018.
- Marcus, G.** 2018. Deep learning: A critical appraisal. [Online] 2018. [Zitat vom: 21. 05. 2019.] arXiv preprint arXiv:1801.00631.
- Marks, Anna.** 2019. How AI is radically changing our definition of human creativity. *Wired*. [Online] 15. 06. 2019. [Zitat vom: 26. 09. 2019.] <https://www.wired.co.uk/article/artificial-intelligence-creativity>.
- Metz, Cade.** 2016. Inside OpenAI, Elon Musk's Wild Plan to Set Artificial Intelligence Free. *Wired*. [Online] 27. 04. 2016. [Zitat vom: 14. 01. 2020.] <https://www.wired.com/2016/04/openai-elon-musk-sam-altman-plan-to-set-artificial-intelligence-free/>.
- Miners, Zach.** 2014. Google buys smart thermostat maker Nest for a cool \$3.2B. *Computerworld*. 2014, Jan 13, 2014.
- Mistreanu, S.** 2018. Life Inside China's Social Credit Laboratory. *Foreign Policy*. 2018, <https://foreignpolicy.com/2018/04/03/life-inside-chinas-social-credit-laboratory/>.
- Mitchell, Melanie.** 2018. Artificial Intelligence Hits the Barrier of Meaning. *New York Times*. 5. Nov. 2018.
- Mukherjee, Siddhartha.** 2017. A.I. Versus M.D. – What happens when diagnosis is automated? *The New Yorker*. 27.03.2017.
- Ng, Andrew.** 2016. What Artificial Intelligence Can and Can't Do Right Now. [Online] 2016. [Zitat vom: 25. 05. 2019.] <https://hbr.org/2016/11/what-artificial-intelligence-can-and-cant-do-right-now>.
- Nicklaus, Jack.** 2007. *Golf My Way: The Instructional Classic, Revised and Updated*. s.l. : Simon & Schuster, 2007.
- O'Neil, Cathy.** 2016. Weapons of Math Destruction. Crown Books, 2016.

- Osborn, L. E., et al.** 2018. Prosthesis with neuromorphic multilayered e-dermis perceives touch and pain. *Science Robotics*, 3(19), eaat3818. 2018.
- Pearl, J.** 2018. Theoretical impediments to machine learning with seven sparks from the causal revolution. [Online] 2018. [Zitat vom: 13. 09. 2019.] arXiv preprint arXiv:1801.04016.
- Petzinger, Jill.** 2018. Europe's newest unicorn is a process-mining startup founded by German students. *Quartz*. [Online] 26. 06. 2018. [Zitat vom: 13. 01. 2020.] <https://qz.com/1314817/europe-s-newest-tech-unicorn-celonis-is-a-data-mining-startup-founded-by-german-students/>.
- Radebaugh, C.** 2019. Introducing TensorFlow Privacy: Learning with Differential Privacy for Training Data. *medium*. [Online] 06. 03. 2019. [Zitat vom: 14. 01. 2020.] <https://medium.com/tensorflow/introducing-tensorflow-privacy-learning-with-differential-privacy-for-training-data-b143c5e801b6>.
- Richter, R. R. und Austin, T. M.** 2012. Using MeSH (medical subject headings) to enhance PubMed search strategies for evidence-based practice in physical therapy. *Physical therapy*, 92(1), 124–132. 2012.
- Scudellari, M.** 2019. Bionic ‘Feeling’ Leg Makes Walking Easier, Reduces Phantom Limb Pain. *IEEE Spectrum*. 09. Sept. 2019.
- Severson, Kim.** 2020. Thanks to A.I., Machines Get a Taste for the Right Kinds of Food. *New York Times*. 08.04.2020.
- Shoham, Y., et al.** 2018. The AI Index 2018 Annual Report. *AI Index Steering Committee, Human-Centered AI Initiative, Stanford University, Stanford, CA*. 2018.
- Slavin, E.** 2017. Pentagon unveils Perdix micro-drone swarm. *Stars and Stripes*. [Online] 10. 01. 2017. [Zitat vom: 14. 01. 2020.] <https://www.stripes.com/news/pentagon-unveils-perdix-micro-drone-swarm-1.448124>.
- Smith, Craig S.** 2020a. Dealing with Bias in Artificial Intelligence. *The New York Times*. 02.01.20.
- Smith, C.** 2020b. Want to Be Better at Sports? Listen to the Machines. *New York Times*. 2020, 08.04.2020.
- Tagesspiegel.** 2019. Amazon-Mitarbeiter hören Alexa-Aufzeichnungen ab. *Der Tagesspiegel*. 2019, Bde. <https://www.tagesspiegel.de/wirtschaft/auch-vertrauliche-daten-betroffen-amazon-mitarbeiter-hoeren-alexa-aufzeichnungen-ab/24209788.html>, 11.04.2019.
- TensorProb.** 2019. Tensorflow probability. [Online] 2019. [Zitat vom: 13. 09. 2019.] <https://www.tensorflow.org/probability/overview>.
- Tischbein, Valerie.** 2016. 98 Daten, die Facebook über dich weiß und nutzt, um Werbung auf dich zuzuschneiden. *Netzpolitik.org*. [Online] 22. 08. 2016. [Zitat vom: 11. 04. 2020.] <https://netzpolitik.org/2016/98-daten-die-facebook-ueber-dich-weiss-und-nutzt-um-werbung-auf-dich-zuzuschneiden/>.
- Valentino-deVries, Jennifer, et al.** 2018. Your Apps Know Where You Were Last Night, and They’re Not Keeping It Secret. *The New York Times*. 10.12.2018.
- Wang, H., et al.** 2018. A Captcha Design Based on Visual Reasoning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1967–1971). IEEE. 2018.
- Whittaker, Zack.** 2019. OpenAI built a text generator so good, it’s considered too dangerous to release. *techcrunch*. [Online] 17. 02. 2019. [Zitat vom: 14. 01. 2020.] <https://techcrunch.com/2019/02/17/openai-text-generator-dangerous/>.
- Wrobel, S. und Hecker, D.** 2019. Fraunhofer Big Data and Artificial Intelligence Alliance. In: Neugebauer R. (eds) *Digital Transformation*, Springer, Berlin. S. 253–264. 2019.

---

## A Anhang

---

## A.1 Glossar

Konzept (deutsch)	Konzept (englisch)	Bedeutung
5G	5G	Ist die fünfte Generation eines Mobilfunkstandards drahtloser Breitbandtechnologie für mobiles Internet und Mobilfunktelefonie. 5G zählt aktuell zu den leistungsfähigsten Mobilfunkstandards.
Aktivierungsfunktion	Activation function	Eine nichtlineare Funktion als Operator in einem Netz, die eine Nichtlinearität in das Netz einbringt. • Oft Bestandteil einer vollverbundenen Schicht (Fully Connected Layer).
Algorithmus	Algorithm	Eine Schritt-für-Schritt Anleitung zur Durchführung einer Aufgabe, in der Regel durch einen Computer. Computeralgorithmen können einfach sein (z. B. bestimme die Summe der beiden Zahlen durch folgende Schritte ...) oder komplex (z. B. bestimme das Objekt in diesem Foto durch folgende Schritte ...).
Autoencoder	Autoencoder	Ein Autoencoder ist ein künstliches neuronales Netz, das dazu genutzt wird, effiziente Kodierungen zu lernen. Es besteht aus mehreren Schichten. • Ziel ist es, die Eingabe durch die Ausgabe zu reproduzieren. • Ein- und Ausgabevektor sind gleich groß. • Die mittlere Schicht hat nur einen kurzen verdeckten Vektor. • Dadurch ist das Netz gezwungen, eine effektive Komprimierung der Eingaben in dem verdeckten Vektor zu erzeugen.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Backpropagation	Backpropagation	<p>Verfahren zu Berechnung des Gradienten <math>\partial L(w; x, y)/\partial w</math> einer Verlustfunktion bei neuronalen Netzen bei den gegebenen Trainingsdaten <math>x, y</math>. Wegen der einzelnen Schichten eines Netzes ist die Verlustfunktion eine geschachtelte Funktion: <math>L(w : x, y) = L(f_k(\dots f_1(x, w)\dots), y)</math></p> <p>Deren Ableitung wird mit der Kettenregel berechnet.</p> <ul style="list-style-type: none"> <li>• Der Verlust <math>L</math> bewertet die Abweichung zwischen prognostizierter und beobachteter Ausgabe <math>y</math>. Daher muss das TNN für eine gegebene Eingabe <math>x</math> zunächst die Ausgabe prognostizieren.</li> <li>• Im Effekt wird die Ableitung des Verlustes von der Ausgabe- zur Eingabeschicht „zurückpropagiert“.</li> <li>• In aktuellen Frameworks wird die Ableitung automatisch berechnet.</li> </ul>
Beam-Suche	Beam-Search	Ein heuristischer Suchalgorithmus. Er berücksichtigt die bisherigen $k$ besten Teilergebnisse. Bei der maschinellen Übersetzung wird er bei der Erzeugung des Ausgabesatzes verwendet.
Belohnung beim Bestärkungslernen	Reward	Für jeden Zeitpunkt $t$ eine reelle Zahl $r_t$ , z. B. ein Geldbetrag. Ziel des Bestärkungslernens ist es, die Summe der Belohnungen zu maximieren.
Benannte Entität	Named Entity	Benanntes Objekt (Name) im Text. Es werden täglich neue Namen für solche Entitäten (Personen, Orte, Produkte, usw.) gebildet. Daher ist deren Erkennung schwierig und nur aus dem Kontext möglich.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Bestärkungslernen	Reinforcement Learning	<p>Eine Lernaufgabe, bei dem das Lernsignal erst nach einer Reihe von Schritten verfügbar ist.</p> <p>Beispiel: Gewinn, Verlust oder Remis beim Schachspiel</p> <ul style="list-style-type: none"> <li>• In jedem Schritt muss der Agent (der Algorithmus) für einen Zustand <math>s_t</math> eine Aktion <math>a_t</math> auswählen. Diese Aktion beeinflusst den weiteren Verlauf.</li> <li>• Beim Schachspiel ist der Zustand die Position der Spielstein auf dem Brett und die Aktion ein Spielzug.</li> <li>• Hieraus ergibt sich ein neuer Zustand <math>s_{t+1}</math> und ein numerischer Reward (Belohnung) <math>r_{t+1}</math>. Beim Schachspiel gibt es die Belohnungen: Verloren (-1), Remis (0), gewonnen (+1), noch nicht entschieden (0).</li> <li>• Ziel ist es die Aktionen so auszuwählen, dass Summe aller Belohnungen maximiert wird.</li> <li>• Beim tiefen Bestärkungslernen wird ein neuronales Netz konstruiert, welches in jedem Zustand die optimale Aktion prognostiziert. Hierdurch können Lernaufgaben mit extrem vielen Zuständen gelöst werden.</li> <li>• Während des Trainings des Systems wird versucht, den Beitrag einzelner Aktionen in den unterschiedlichen Zuständen zur Erreichung des Ziels zu bestimmen.</li> </ul>
Bidirektionales LSTM	Bidirectional LSTM	<p>Bei der Prognose der Eigenschaft <math>y_t</math> mit einem rekurrenten neuronalen Netz (RNN) werden nur die vorherigen Worte berücksichtigt. Das bidirektionale RNN berücksichtigt vorherige und nachfolgende Worte:</p> <ul style="list-style-type: none"> <li>• Vorwärts-RNN berücksichtigt die vorherigen Worte und prognostiziert die verdeckten Vektoren <math>\tilde{h}_t</math></li> <li>• Rückwärts-RNN berücksichtigt die nachfolgenden Worte und prognostiziert die verdeckten Vektoren <math>\tilde{\tilde{h}}_t</math></li> <li>• Das logistische Regressionsmodell zur Prognose <math>y_t</math> erhält die verdeckten Vektoren beider Modelle <math>[\tilde{h}_t, \tilde{\tilde{h}}_t]</math> als Input.</li> </ul>
Bildverstehen	Computer vision	Computerprogramme, die eine inhaltliche Interpretation von Bildern oder Videos erlauben.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Black-Box-Modell	Black-box-model	Ein Modell des maschinellen Lernens oder ein TNN, welches Eingabetensoren in Ausgabentensoren transformiert. Der innere Ablauf der Transformation ist (nahezu) unbekannt und kann meist nicht von Menschen verstanden werden.
BLEU	BLEU	Performanzmaß zur Beurteilung einer Übersetzung. Es wird das Übersetzungsresultat mit Referenzübersetzungen verglichen. Dabei wird die Anzahl von gleichen Worten, 2-Grammen, ..., 4-Grammen berücksichtigt. • Höherer Wert → bessere Übersetzung
Chatbot	Chatbot	Ein Chatbot oder kurz Bot ist ein textbasiertes Dialogsystem, welches Nachrichten mit Internetnutzern austauscht.
CNN	CNN	siehe Convolutional Neural Network
Convolutional Neural Network, CNN	Convolutional Neural Network (CNN)	neuronales Netz, auch Faltungsnetz genannt, welches meist auf 2- oder 3-dimensionale Eingabentensoren angewendet wird (z. B. Pixelmatrix eines Bildes) und die Klasse eines Objektes im Bild prognostiziert. • Convolution-Schicht: Es wird eine kleine 2-dimensionale Matrix (Kernel) über die Eingabe bewegt, berechnet für jede Position einen Wert. • Pooling-Schicht: Summierung von benachbarten Werten Diese Schichten können mehrfach wiederholt werden. • Die letzte Schicht ist meist eine vollverbundene Schicht mit einer Softmax-Aktivierungsfunktion.
Daten	Data	Eine Menge von Beispieldaten (Beobachtungen). • Jedes Beispieldatum enthält Eingaben (und ggf. Ausgaben) für das maschinelle Lernverfahren, z. B. Bilder, Sprachsignale, Texte, Messwerte. Alle Arten von Daten werden zur Verarbeitung in Zahlen, meist Tensoren, konvertiert.
Digitalisierung	Digitization	Digitalisierung hat sich zum Ziel genommen, analoge Objekte und Abläufe in digitale Daten und Prozesse umzuwandeln. Die digitale Transformation geht einher mit dem Internet of Things, bei der sich Mensch, Maschine und Gerät vernetzen. Bis 2020 wird erwartet, dass sich 50 Milliarden Geräte miteinander vernetzt haben und Daten intelligent miteinander austauschen.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Dropout	Dropout	<p>Ein Ansatz zur Regularisierung.</p> <ul style="list-style-type: none"> <li>• Er geht aus von einem verdeckten Vektor <math>h</math>, innerhalb eines Modells.</li> <li>• In einer Dropout-Schicht wird ein bestimmter Prozentsatz (z. B. 50 %) der Komponenten von <math>h</math> zufällig ausgewählt und auf 0 gesetzt.</li> </ul>
Embedding, Wort-Embedding, Embeddingvektor	Word Embedding	<p>Vektor, welcher ein Wort einer Sprache repräsentiert. Die Einbettungen semantisch ähnliche Worte sollen einen geringen Vektor-Abstand haben.</p> <ul style="list-style-type: none"> <li>• Es können auch andere Objekte eingebettet werden (z. B. Sätze, Bilder).</li> </ul>
Embedding-Netz	Embedding Network	<p>Berechnet eine Einbettung für die Worte in einer Sequenz.</p> <ul style="list-style-type: none"> <li>• Die Einbettung eines Wortes ist der Input des Netzes.</li> <li>• Aus diesem Input werden die Worte in der Nachbarschaft des Wortes möglichst gut prognostiziert.</li> <li>• Dabei werden Parameter der Einbettung so optimiert, dass die Prognose möglichst gut ist.</li> </ul>
Ensemblemethoden, Ensemble von Modellen	Ensemble Methods	<p>Ensemblemethoden nutzen eine Menge gleichartiger Modelle, um eine Prognoseaufgabe zu lösen.</p> <p>Die Modelle weisen zufällige bzw. systematische Unterschiede auf.</p> <ul style="list-style-type: none"> <li>• Bootstrap oder Bagging ändern die Trainingsmenge zufällig.</li> <li>• Boosting konzentriert sich nach und nach auf die schwierigen Prognosefälle.</li> </ul> <p>Es lässt sich für beide Fälle zeigen, dass die Prognosevarianz geringer ist als bei Einzelmodellen. Die Methoden lassen sich für beliebige maschinelle Lernverfahren anwenden, d. h. auch für neuronale Netze.</p>
Episode beim Reinforcement-Lernen	Episode	Eine Sequenz von beobachteten Tripeln (Zustand, Belohnung, Aktion): $(s_1, r_1, a_1), (s_2, r_2, a_2), \dots, (s_n, r_n, a_n)$
Epoche	Epoch	Ein Durchgang durch sämtliche Trainingsdaten bei den Iterationen für die Optimierung.
Expertensystem	Expert System	Ein KI-System, welches Inhalte als logische Fakten und Regeln darstellt, welche meist manuell eingegeben werden. Man kann dann Eigenschaften von Objekten durch logische Schlussfolgerungen ableiten. War im vorigen Jahrhundert weit verbreitet.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Explodierender Gradient	Exploding Gradient	Nach $k$ Durchgängen durch eine Schicht geht ein Parameter $w$ als $w^k$ in den Gradienten ein. <ul style="list-style-type: none"> <li>• <math>w &gt; 1</math>: der Wert von <math>w^k</math> wird extrem groß und es ergibt sich ein explodierender Gradient</li> <li>• Optimierer konvergiert nicht</li> </ul>
Exploration	Exploration	Um die beste Strategie zu finden ist es notwendig, dass im Training die gesamte Zustands-Aktions-Menge evaluiert wird. Daher sollen neue Episoden nicht immer mit den optimalen Aktionen gemäß $\hat{Q}(s_t, a; w)$ bestimmt werden, sondern Aktionen auch manchmal zufällig gewählt werden
F-Wert	F-value	Ein Performanzmaß der Klassifikation bezüglich einer bestimmten Klasse $c$ : $F = \frac{2 * \text{Präzision}_c * \text{Recall}_c}{\text{Präzision}_c + \text{Recall}_c}$ Harmonisches Mittel von $\text{Präzision}_c$ und $\text{Recall}_c$ . Wert zwischen 0.0 (schlecht) und 1.0 (gut)
Genauigkeit	Accuracy	Ein Performanzmaß der Klassifikation: Anzahl der korrekt klassifizierten Testbeispiele dividiert durch Anzahl aller Testbeispiele. Wert zwischen 0.0 (schlecht) und 1.0 (gut)
Generalisierung	Generalization	Sinnvolle und adäquate Übertragung der Zusammenhänge in den vorliegenden Daten auf neue Daten.
Generatives adversariales Netz (GAN)	generative adversarial Networks	Generative adversarial Networks bestehen aus zwei neuronalen Netzen, die gegeneinander arbeiten. Das Generator-Netz erstellt Objekte, z. B. Bilder. Das Diskriminator-Netz erhält in zufälliger Reihenfolge echte Bilder oder erzeugte Bilder. Es muss entscheiden, ob es sich um echte oder erzeugte Bilder handelt. <ul style="list-style-type: none"> <li>• Typischerweise bildet das Generator-Netz einen Vektor aus zufälligen Werten auf den gewünschten Ergebnisraum ab. Das Ziel des Generators ist es, Bilder zu erzeugen, die vom Generator nicht abgelehnt werden.</li> <li>• Der Diskriminator wird hingegen darauf trainiert, die Bilder des Generators von den Bildern aus der Trainingsmenge zu unterscheiden.</li> </ul> Im Ergebnis kann der Generator Bilder erzeugen, die der Diskriminator nicht mehr von den echten Bildern der Trainingsmenge unterscheiden kann.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Gerichteter Graph	Directed Graph	<p>Neuronale Netze haben oft nicht nur aufeinanderfolgende Schichten, sondern bilden einen gerichteten Graphen von Operatoren.</p> <ul style="list-style-type: none"> <li>Die Knoten des Graphen (Operatoren) modellieren einzelne Berechnungen.</li> <li>Die gerichteten Kanten des Graphen entsprechen Daten, welche von einem Operator zu einem anderen fließen. Diese Kanten können parallel sein.</li> </ul> <p>Meist handelt es sich um gerichtete azyklische Graphen.</p> <p>Diese Modellstrukturen sind in TensorFlow darstellbar.</p>
G-Kraft	G-Force	G-Kräfte sind Belastungen, die aufgrund starker Änderung von Größe und/oder Richtung der Geschwindigkeit auf den menschlichen Körper oder ein Fahrzeug einwirken. Insbesondere auf Kampfpiloten und Astronauten wirken hohe G-Kräfte im Luftkampf oder beim Raketenstart ein. Hohe G-Kräfte können vom Menschen nur kurz ausgehalten werden.
Gradient von $L(w)$	Gradient of $L(w)$	<p>Bei gegebener Datenmenge <math>x, y</math>: Für eine Funktion <math>L(w)</math> mit mehrdimensionaler Eingabe <math>w</math> und einen Vektor <math>w_0</math> die Richtung (Vektor)</p> $\frac{\partial L(w)}{\partial w}(w_0)$ <p>in der die Funktion <math>L(w)</math> im Punkt <math>w_0</math> den steilsten Anstieg hat.</p>
Gradientenabstiegs-optimierung	Gradient Descent Optimization	<p>Bei gegebener Datenmenge <math>x, y</math>: Eine Methode zur Parameteroptimierung für eine Funktion <math>L(w)</math>:</p> <ul style="list-style-type: none"> <li>Zuerst wird der Wert des Parametervektors <math>w</math> initialisiert (z. B. zufällig).</li> <li>Dann wird der Gradient <math>\partial L(w)/\partial w</math> für den aktuellen Parameter <math>w</math> berechnet. Es ergibt sich ein Vektor <math>\partial L(w)/\partial w</math>, der gleich lang ist wie <math>w</math>. Dieser Gradient wird mit der Lernrate <math>\lambda</math> (kleine positive Zahl) multipliziert und von dem aktuellen Parameter <math>w</math> abgezogen.</li> </ul> $w := w - \lambda \partial L(w)/\partial w$ <p>Diese Schritt wird wiederholt bis sich keine Änderungen mehr im Wert von <math>L(w)</math> ergeben. <math>\lambda</math> ist ein Hyperparameter.</p>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Grafikprozessor, GPU	Graphical Processing Unit, GPU	<p>Ein auf die Berechnung von Grafiken spezialisierter und optimierter Prozessor.</p> <ul style="list-style-type: none"> <li>• Enthält oft mehrere Tausend Rechenwerke und hat eine sehr hohe Rechenkapazität.</li> <li>• Die aktuellen Frameworks für neuronale Netze (z. B. TensorFlow) können den Grafikprozessor für parallele Berechnungen nutzen.</li> </ul>
HAL 9000	HAL 9000	<p>Der britische Schriftsteller und Physiker Arthur C. Clarke entwickelte in seinem Buch „2001: Odyssee im Weltraum“ den fiktiven Computer Hal 9000. An Bord des Raumschiff Discovery steuert und überwacht der Supercomputer Hal (Heuristically programmed Algorithmic Computer) die Prozesse und den Zweck der Mission. Neben der Raumschiffsteuerung beeindruckt Hal auch durch Schachspielen, Sprache erkennen und natürlichsprachiges Antworten.</p>
Hyperparameter	Hyperparameter	<p>Ein Hyperparameter ist ein Parameter, dessen Wert gesetzt wird, bevor der Lernprozess beginnt, z. B. die Anzahl der Layer in einem neuronalen Netz.</p> <ul style="list-style-type: none"> <li>• Im Gegensatz zu normalen Parametern werden die Werte der Hyperparameter nicht durch das Training bestimmt.</li> </ul>
Hyperparameter-Optimierung	Hyperparameter Optimization	<p>Verfahren zur Auswahl der Werte von Hyperparametern.</p> <ul style="list-style-type: none"> <li>• Ein einfacher Ansatz trainiert mehrere Modelle auf den Trainingsdaten mit unterschiedlichen Werten der Hyperparameter und vergleicht die Modellperformance auf einer zusätzlichen Validationsmenge von annotierten Beispielen.</li> </ul>
Informationsextraktion	Information Extraction	<p>Die Extraktion von Informationen aus Text, insbesondere:</p> <ul style="list-style-type: none"> <li>• Die Extraktion von Entitäten (Objekten mit Namen) und die Bestimmung des Typs der Entitäten (Named Entity Recognition).</li> <li>• Die Extraktion von Relationen zwischen den Entitäten (Relation Extraction).</li> </ul>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Internet der Dinge (IoT)	Internet of Things (IoT)	Internet of Things (IoT), oder das Internet der Dinge, bezeichnet eine weitere Entwicklung der Digitalisierung. Im Internet der Dinge vernetzen sich Produkte und Geräte sowohl untereinander als auch direkt mit dem Internet selbst. Alltagsgegenstände werden z. B. mit Prozessoren oder Sensoren ausgerüstet und geben im Laufe ihrer Konnektivität zahlreiche Informationen über den Gebrauch, die Umgebung und das Nutz(er)verhalten wieder. Das IoT schafft somit Potentiale für Leistungs- und Serviceoptimierungen der vernetzten Geräte.
Keras	Keras	Keras ist eine in Python geschriebene Bibliothek für neuronale Netze. <ul style="list-style-type: none"><li>• Enthält zahlreiche Bausteine von neuronalen Netzen, welche auf einfache Weise verwendet werden können.</li><li>• Fungiert als Steuermodul für TensorFlow.</li><li>• Die Berechnungen werden automatisch auf der GPU durchgeführt.</li></ul>
Kernel	Kernel	Bei einer Convolution-Schicht wird ein kleines Netz über einen großen Tensor (z. B. Matrix) von Eingaben bewegt. Erzeugt für jede Position einen Ausgabewert.
Kettenregel	Chain Rule	Regel, nach der man die Ableitung geschachtelter Funktionen $f(g(x))$ berechnen kann. Dies wird bei der automatischen Berechnung von Ableitungen verwendet.
Klassifikation	Classification	Eine Art des überwachten Lernens, bei der die Zielvariable nur endlich viele diskrete Werte annehmen kann.
Künstliche Intelligenz (KI)	Artificial Intelligence	Der Begriff Künstliche Intelligenz, abgekürzt KI, steht für Computersysteme, die menschliche Intelligenz nachahmen.
Künstliches neuronales Netz (KNN)	artificial neural network, ANN	siehe neuronales Netz.
Lernrate	Learning Rate	Kleine positive reelle Zahl $\lambda$ . Der Gradient wird mit der Lernrate $\lambda$ multipliziert und von dem aktuellen Parameter $w$ subtrahiert.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Lineare Transformation	Linear Transformation	<p>Multipliziert einen Eingabevektor <math>x</math> mit einer Matrix <math>W</math> und addiert einen Vektor <math>b</math>, genannt Bias</p> $u = W * x + b$ <p>also</p> $u_i = w_{i,1}x_1 + \dots + w_{i,n}x_n + b_i$ <ul style="list-style-type: none"> <li>• Überführt gerade Linien in gerade Linien.</li> <li>• <math>W</math> bestimmt die Drehung und Streckung der Transformation und <math>b</math> eine globale Verschiebung.</li> <li>• Grundbaustein der meisten Modelle.</li> </ul>
Logistische Regression, logistisches Regressionsmodell	logistic Regression	<p>Das logistische Regressionsmodell berechnet die Wahrscheinlichkeit von Klassen für einen Input vektor <math>x</math> durch</p> $p(y x) = \text{softmax}(Ax + b)$ <p>Wählt man die Klasse mit der höchsten Wahrscheinlichkeit, so erhält man den zugehörigen Klassifikator.</p> <ul style="list-style-type: none"> <li>• Bestandteil vieler tiefer neuronaler Netze.</li> </ul>
Long Short Term Memory, LSTM	Long Short Term Memory, LSTM	<p>Eingaben: Eingabe <math>x_t</math>, Memory Vektor <math>c_{t-1}</math> und verdeckter Vektor <math>h_{t-1}</math></p> <p>Hat drei Komponenten:</p> <ul style="list-style-type: none"> <li>• Speicher-Gate: addiert modifizierte Inputs <math>[x_t, h_{t-1}]</math> zum Memory Vektor, gesteuert durch transformierte Eingaben <math>[x_t, h_{t-1}]</math>.</li> <li>• Vergessen-Gate: reduziert einzelne Memory Vektor Komponenten, gesteuert durch transformierten Eingaben <math>[x_t, h_{t-1}]</math>.</li> <li>• Output-Gate: liest Memory Vektor Komponenten aus gesteuert durch transformierte Eingaben <math>[x_t, h_{t-1}]</math>. Die ausgelesenen Komponenten werden transformiert und bilden den neuen verdeckten Vektor <math>h_t</math>.</li> </ul> <p>Ausgaben: Memory Vektor <math>c_t</math> und verdeckter Vektor <math>h_t</math></p> <p>Hat 5 vollverbundene Schichten mit trainierbaren Parametern.</p>
Losgröße 1	Lot size 1	Die Losgröße 1 beschreibt in der Fertigung die Möglichkeit, individualisierte Produkte zum Preis von Massenprodukten herzustellen. Die Herstellungsprozesse sind so effizient und hochautomatisiert, dass es nunmehr keinen bedeutenden Kostenunterschied macht, ob man ein individuelles Kundenprodukt oder ein Serienprodukt produziert.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Maschinelle Übersetzung	Machine Translation	Automatische Übersetzung von Text aus einer Sprache in eine andere.
Maschinelles Lernen	Machine Learning	Extrahiert mithilfe eines trainierbaren Modells selbsttätig Informationen aus Daten, die für den Benutzer relevant sind.
Matrix	Matrix	Ein rechteckiges Arrangement von reellen Zahlen, z. B. $\begin{pmatrix} 1.7 & \dots & -9.5 \\ \vdots & \ddots & \vdots \\ 2.8 & \dots & 3.8 \end{pmatrix}$
Maximum-Likelihood-Prinzip	Maximum Likelihood Principle	Verwendet ein Modell, welches von einem Parameter $w$ abhängt und die Wahrscheinlichkeit (Likelihood) der Trainingsdaten $lik(w)$ bestimmt, wenn diese von dem Modell erzeugt worden wären <ul style="list-style-type: none"> <li>• Maximum-Likelihood-Prinzip: ändere den Parametervektor <math>w</math> so, dass <math>lik(w)</math> maximal wird.</li> </ul>
Mehrschichtiges Feedforward-Netz	Multilayer Neural Network	Ein neuronales Netz, welches aus mehreren hintereinandergeschalteten Schichten (Layern) besteht. <ul style="list-style-type: none"> <li>• Jede Schicht bildet eine Funktion <math>x^{(i+1)} = f_i(x^{(i)})</math>, welche einen Tensor <math>x^{(i)}</math> mit der Funktion <math>f_i</math> in einen anderen Tensor <math>x^{(i+1)}</math> transformiert.</li> <li>• Das Netz führt diese Funktionen nacheinander aus.  <math>x^{(i)} = f_{i-1}(\dots f_1(x^{(1)})\dots)</math></li> <li>• Es kann zur Klassifikation (mit Softmax im letzten Layer) oder zur Regression (mit linearem letzten Layer) verwendet werden.</li> </ul>
Mehrschichtiges RNN	Multilayer RNN	RNN mit mehreren Schichten: <ul style="list-style-type: none"> <li>• Input der höheren Schichten sind die ausgegebenen verdeckten Vektoren der darunterliegenden Schicht.</li> <li>• Aus dem verdeckten Vektor der höchsten Schicht wird das nächste Element (Wort) durch das logistische Modell prognostiziert.</li> </ul> <p>Alternativ können die verdeckten Vektoren des letzten Prognosezeitpunktes weiter verwendet werden.</p>
Minibatch	Minibatch	Eine kleine Teilmenge (Batch) der beobachteten Trainingsbeispiele. <ul style="list-style-type: none"> <li>• Gradienten werden meist für alle Trainingsbeispiele eines Minibatches aufaddiert.</li> </ul>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Modell	Model	Eine Funktion, die einer Eingabe (z. B. Bild, Tensor) eine Ausgabe (z. B. Klassenlabel) zuordnet. Ein Modell hat i.a. einen Vektor $w$ von ‚freien‘ Parametern, dessen Wert anhand der Daten optimiert wird.
Modellanwendung	Model Application	Anwendung eines trainierten Modells zur Prognose/Analyse von Daten, die nicht beim Training verwendet wurden.
Neuronales Netz	Neural Network	Ein aus mehreren Schichten bestehendes Computerprogramm, welches eine Eingabe (z. B. Bilder) durch einfache Berechnungen in eine Ausgabe (z. B. Objektklasse) transformiert. Die nächste Schicht erhält die Ausgabe der vorigen Schicht als Eingabe. Die Verbindung der Schichten kann auch netzartig aussehen. Ein neuronales Netz hat einen Vektor von unbekannten Parametern, welcher durch Optimierungsverfahren verändert wird. Ziel ist dabei, aus den beobachteten Eingaben die zugehörigen Ausgaben möglichst gut zu prognostizieren.
N-Gramm	Ngram	Sequenzen von der festen Länge $n$ <ul style="list-style-type: none"> <li>• Buchstaben N-Gramm: Abschnitte von <math>n</math> aufeinanderfolgenden Buchstaben aus einem Text.</li> <li>• Wort N-Gramm: Abschnitte von <math>n</math> aufeinanderfolgenden Worten aus einem Text.</li> </ul>
Nichtlineare Funktion	Nonlinear Function	Eine Funktion $g(x)$ , die keine lineare Funktion $f(x) = Ax + b$ ist.
Operator	Operator	Bestandteil in einem neuronalen Netz, welcher Berechnungen durchführt. <ul style="list-style-type: none"> <li>• Hat ein oder mehrere Tensoren als Eingabe und ein oder mehrere Tensoren als Ausgabe.</li> <li>• Historisch als Schicht (Layer) bezeichnet.</li> </ul>
Parameter	Parameter	Ein neuronales Netz (oder anderes Modell des maschinellen Lernens) hat einen Vektor von unbekannten Parametern, welcher durch Optimierungsverfahren verändert wird. Ziel ist dabei, aus den beobachteten Eingaben die zugehörigen Ausgaben möglichst gut zu prognostizieren.

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Perplexität	Perplexity	Maßzahl für die Güte des Wortmodells: die inverse Wahrscheinlichkeit der Testmenge $w_1, \dots, w_N$ , nomalisiert durch die Anzahl der Worte der Testmenge also $p(w_1, \dots, w_N)^{-1/N}$ • Kleiner ist besser.
Pooling-Schicht	Pooling Layer	Eine Komponente in einem Convolutional Neural Network, welche eine Aggregation (z. B. Maximum) benachbarter Elemente in einem Tensor berechnet. Der Ausgabetensor enthält also weniger Elemente.
Präzision	Precision	Ein Performanzmaß der Klassifikation bezüglich einer bestimmten Klasse $c$ : <i>Precision</i> : Anzahl der korrekt zugeordneten Testbeispiele der Klasse $c$ , dividiert durch Anzahl aller der Klasse $c$ zugeordneten Testbeispiele. Wert zwischen 0.0 (schlecht) und 1.0 (gut)
Q-Funktion beim Reinforcement-Lernen	Q-Function	Ordnet jedem Zustand-Aktionspaar $(s, a)$ einen Wert zu: <ul style="list-style-type: none"> <li>die maximale diskontierte zukünftige Belohnung, die für <math>(s, a)</math> mit einer optimalen Strategie <math>\pi^* : a \rightarrow s</math> erreichbar ist.</li> <li><math>\gamma &lt; 1</math> ist der Diskontfaktor, der steuert, wie die zukünftigen Belohnungen abgewertet werden.</li> </ul> $Q(s_t, a_t) = \max_{\pi^*} (r_{t+1} + \gamma^1 r_{t+2} + \dots + \gamma^{n-1} r_{t+n})$
Recall	Recall	Ein Performanzmaß der Klassifikation bezüglich einer bestimmten Klasse $c$ : <i>Recall</i> : Anzahl der korrekt zugeordneten Testbeispiele der Klasse $c$ , dividiert durch die Anzahl aller tatsächlichen Testbeispiele der Klasse $c$ . Wert zwischen 0.0 (schlecht) und 1.0 (gut)
Regression	Regression	Eine Art des überwachten Lernens, bei der die Zielvariable kontinuierlich ist. Sie kann ein Skalar oder ein Tensor sein.
Regularisierung	Regularization	Maßnahme, um eine Überanpassung zu vermeiden. <ul style="list-style-type: none"> <li>Reduktion der Modellkomplexität (weniger Layer, weniger Parameter)</li> <li>Zusätzlicher Term der Verlustfunktion, der die Werte der Parameter in Richtung 0.0 „zieht“.</li> <li>Dropout</li> <li>Minibatch-Normalisierung usw.</li> </ul>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Rekurrentes neuronales Netz, RNN	Recurrent Neural Network, RNN	<p>Modell zur Prognose des nächsten Elementes (Wort) <math>w_t</math> einer Sequenz <math>w_1, \dots, w_n</math>:</p> <ul style="list-style-type: none"> <li>• Eingabe: voriges Element <math>w_{t-1}</math> und voriger verdeckter Vektor <math>h_{t-1}</math></li> <li>• Ausgabe: nächster verdeckter Vektor <math>h_t</math> und Wahrscheinlichkeiten für alle möglichen des nächsten Elemente <math>w_t</math>.</li> <li>• Verlustfunktion: gleich wie beim logistischen Regressionsmodellen, wird summiert über die gesamte Sequenz.</li> </ul> <p>Kann Sequenzen variabler Länge als Netz mit fester Ein-/Ausgabedimension verarbeiten.</p>
Sequence-to-Sequence Modell	Sequence-to-Sequence Model	<p>Ein Modell zur Übersetzung einer Eingabesequenz in eine entsprechende Ausgabesequenz. Eine wichtige Anwendung ist die Übersetzung eines Satzes aus einer Sprache in eine andere.</p> <ul style="list-style-type: none"> <li>• Die Eingabesequenz wird durch ein Encoder-RNN in einen verdeckten Vektor kodiert.</li> <li>• Der verdeckte Vektor wird durch ein Decoder-RNN in die Ausgabesequenz übertragen. Das logistische Regressionsmodell berechnet die Wahrscheinlichkeit der Ausgabeworte.</li> </ul> <p>Trainingskriterium: Erzeuge die Worte der Ausgabesequenz mit möglichst hoher Wahrscheinlichkeit.</p>
Skalar	Scalar	Eine reelle Zahl, z. B. $1.7 \in \mathbb{R}$
Softmaxfunktion	Softmax Function	<p>Eine Funktion, die einen beliebigen Vektor <math>u = (u_1, \dots, u_k)</math> in einen Wahrscheinlichkeitsvektor transformiert.</p> $s = \exp(u_1) + \dots + \exp(u_k)$ $\text{softmax}(u) = \left( \frac{\exp(u_1)}{s}, \dots, \frac{\exp(u_k)}{s} \right)$
Sprachmodell	Language Model	Prognostiziert in einem Text das laufende Word aus allen vorhergehenden Wörtern. Dabei wird die Wahrscheinlichkeit des nächsten Wortes durch ein Modell abgeschätzt.
Stochastische Strategie	Stochastic Policy	<p>Eine nicht-deterministische Strategie, welche in einem Zustand <math>s</math> eine Aktion <math>a</math> zufällig gemäß einer Wahrscheinlichkeitsverteilung <math>\pi(a s)</math> über den Aktionen auswählt.</p> <ul style="list-style-type: none"> <li>• generiert bei gegebenem Zustand <math>s</math>, zufällig eine Aktion <math>a_t \sim \pi(a s_t)</math>.</li> <li>• Oft besser als eine deterministische Strategie: z. B. beim Spiel Stein-Papier-Schere.</li> </ul>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Stochastischer Gradientenabstieg	Stochastic Gradient Descent	<p>Variante der Gradientenabstiegsoptimierung, bei der der Gradient nur für einen Minibatch von Daten (und nicht für alle Trainingsdaten) berechnet wird.</p> <ul style="list-style-type: none"> <li>• Geringer Rechenaufwand.</li> <li>• Gradient schwankt zufällig um den korrekten Wert, daher kann das Verfahren lokale Minima überwinden.</li> </ul> <p>Das Verfahren konvergiert meist wesentlich schneller, als wenn man den Gradienten für die gesamte Trainingsmenge berechnet.</p> <p>Hyperparameter:</p> <ul style="list-style-type: none"> <li>• Die Größe des Minibatches</li> <li>• Lernrate</li> </ul>
Strategie	Policy	<p>Eine Funktion <math>\pi</math>, die beim Bestärkungslernen jedem Zustand <math>s_t</math> eine Aktion <math>a_t</math> zuordnet</p> $\pi : s_t \rightarrow a_t$ <ul style="list-style-type: none"> <li>• Die Strategie kann auch zufällig sein, d.h. eine Aktion gemäß einer Wahrscheinlichkeitsverteilung auswählen. Dies erfordert spezielle Methoden: z. B. Strategiemodell (Policy Model)</li> </ul>
Strategiemodell	Strategiegradientenmodell	<p>Ein Modell zum Bestärkungslernen, welches eine Wahrscheinlichkeitsverteilung <math>\hat{\pi}(a_t   s_t; w)</math> über den Aktionen definiert.</p> <ul style="list-style-type: none"> <li>• Ist eine stochastische Strategie.</li> <li>• Wird durch ein neuronales Netz mit Parameter <math>w</math> realisiert.</li> </ul>
Tensor	Tensor	<p>Ein 1, 2, 3- oder höherdimensionales Arrangement von reellen Zahlen, z. B.</p> $(a_{i,j,k})_{i=1,\dots,r;j=1,\dots,s;k=1,\dots,t} \in \Re^{r \times s \times t}$ <p>auch multidimensionaler Array genannt.</p>
Testbeispiel	Test example, Instance	<p>Eine Beobachtung (Beispiel) aus der Testmenge. Die Testbeispiele sind häufig statistisch unabhängig und sollten der gleichen Verteilung wie die Trainingsbeispiele folgen.</p>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Testmenge	Test set	<p>Eine Menge von beobachteten und gegebenenfalls manuell mit Labels (Annotationen) versehenen beobachteten Beispiele, die <b>nicht</b> für das Training, sondern nur zur Bestimmung der Performanz eines Modells verwendet wird.</p> <ul style="list-style-type: none"> <li>• Hierbei darf die Testmenge nur einmal am Ende der Modellerstellung verwendet werden und die Ergebnisse dürfen nicht zur Verbesserung des Modells genutzt werden.</li> </ul> <p>Hat die Testmenge die gleiche Verteilung wie die zukünftigen Anwendungsdaten, so ist die Performanz auf den Testdaten übertragbar auf die Anwendungsdaten.</p>
Tiefes Lernen	Deep Learning	<p>Der Vorgang mit dem ein tiefes neuronales Netz sich durch schrittweise Optimierung an die Trainingsdaten anpasst. Hierbei werden interne Repräsentationen für die verdeckten Vektoren bestimmt, welche die Eingaben in jeder Schicht des TNN etwas abstrakter kodieren. Damit kann in der letzten Schicht die gewünschte Ausgabe leicht berechnet werden.</p>
Tiefes neuronales Netz, TNN	Deep Neural Network	<p>Ein mehrschichtiges neuronales Netz mit vielen Schichten.</p> <ul style="list-style-type: none"> <li>• Verallgemeinerung: Netz mit beliebig verbundenen Operatoren (Verbindungsgraph). Die Operatoren berechnen aus einem oder mehreren Eingabe-Tensoren einen Ausgabetensor.</li> </ul>
Tiefes Q-Netz	Deep Q-Network	<p>Beim Bestärkungslernen muss die Q-Funktion komplexe Zusammenhänge abbilden:  <math>\Rightarrow</math> approximiere Q-Funktion durch ein tiefes neuronales Netz <math>\hat{Q}(s_t, a_t; w)</math>.</p>
TNN	DNN	Tiefes neuronales Netz
Toolkit zum tiefen Lernen	Deep Learning Toolkit	<p>System zur Formulierung und Ausführung von Deep Learning Modellen: CNTK, Pytorch, TensorFlow, ...</p> <ul style="list-style-type: none"> <li>• Operatoren und Verbindungen werden in einfacher Weise spezifiziert.</li> <li>• Gradienten werden automatisch bestimmt.</li> <li>• Ausführung auf CPUs und GPUs.</li> </ul>
Training	Training	<p>Ändert den Parametervektor <math>w</math> des Modells so ab, dass für jede Eingabe der Unterschied zwischen den vorhergesagten Ausgaben (z. B. Klassenlabeln) und beobachteten Ausgaben möglichst klein wird.</p>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Trainingsbeispiel	Training example, Instance	Eine Beobachtung (Beispiel) aus der Trainingsmenge. Die Trainingsbeispiele sind häufig statistisch unabhängig.
Trainingsmenge	Training set	Eine Menge von beobachteten und gegebenenfalls manuell mit Labels (Annotationen) versehenen Trainingsbeispielen, die <b>nur</b> für das Training verwendet wird. Typischerweise der größte Teil der Daten (60–90 %).
Transferlernen	Transfer Learning	<p>Hierbei wird ein Modell zunächst für eine Lernaufgabe mit sehr vielen Trainingsdaten trainiert.</p> <ul style="list-style-type: none"> <li>Anschließend wird das trainierte Modell für eine andere, aber ähnliche, Lernaufgabe kurz nachtrainiert. Diese Trainingsmenge ist meist sehr klein.</li> <li>Das Modell kann dann oft das Gelernte von der Ausgangslernaufgabe auf die neue Lernaufgabe übertragen und diese befriedigend lösen, obwohl nur wenige Trainingsdaten zur Verfügung standen.</li> </ul>
Überanpassung	Overfitting	<p>Ein Modell kann zu stark an die vorhandenen Daten angepasst werden und damit nicht-systematische, zufällige Einflüsse in den Daten reproduzieren.</p> <ul style="list-style-type: none"> <li>Ein solches Modell hat im Allgemeinen höhere Prognosefehler auf neuen Daten und kann daher nur schlecht generalisieren.</li> <li>Als Gegenmaßnahme muss eine Regularisierung verwendet werden.</li> </ul>
Überwachtes Lernen	Supervised learning	<p>Überwachtes Lernen:</p> <ul style="list-style-type: none"> <li>Es gibt eine ausgewiesene Zielvariable <math>y</math>, deren Werte anhand der Eingabeveriablen <math>x</math> vorherzusagen sind.</li> <li>In der Trainingsmenge liegen Werte für die Eingabeveriable und die Zielvariable vor.</li> <li>Gelernt wird der Zusammenhang zwischen Eingabeveriable und Zielvariable.</li> </ul> <p>Bei der Anwendung des Modells liegen nur die Werte für die Eingabeveriable vor.</p>
Unüberwachtes Lernen	Unsupervised learning	<p>Die Daten sind nicht gelabelt.</p> <ul style="list-style-type: none"> <li>Es gibt keine Zielvariable, nur Eingabeveriable. Gelernt wird der statistische Zusammenhang zwischen den Variablen.</li> <li>Selbstüberwachtes Lernen: Es kann auch der Zusammenhang zwischen Komponenten eines Eingabevektors modelliert werden.</li> </ul>

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Validationsmenge	Validation set	Eine separate Menge von beobachteten und gegebenenfalls manuell mit Labels (Annotationen) versehenen Trainingsbeispiele, die nur für die Messung der Modellgüte während des Trainings und das sukzessive Auswählen von Hyperparametern (z. B. Anzahl der Layer) verwendet wird.
Vektor	Vector	Eine Spalte von Zahlen, z. B. $\begin{pmatrix} 2.7 \\ \vdots \\ -8.9 \end{pmatrix} \in \mathbb{R}^r$
Verarbeitung natürlicher Sprache	Natural language processing	Prozess, bei dem der Computer versucht, gesprochene oder geschriebene Sprache zu „verstehen“. Dabei muss er sowohl den Satzbau (Syntax) als auch die Bedeutung (Semantik) der Worte erfassen. Dies wird heute meist mit maschinellen Lernverfahren und tiefen neuronalen Netzen durchgeführt.
Verdeckter Vektor	Hidden vector	Interner Vektor in einem mehrschichtigen neuronalen Netz. Für diesen Vektor liegen keine Beobachtungswerte vor, sondern er wird während der Modellprognose berechnet.
Verlustfunktion	Loss function	<p>Reellwertige Funktion <math>L(\hat{y}, y)</math>, die für die gegebene Daten <math>x, y</math> den Unterschied zwischen prognostizierten Ausgaben <math>\hat{y}</math> und beobachteten Ausgaben <math>y</math> durch eine reelle Zahl misst.</p> <ul style="list-style-type: none"> <li>• Kleinerer Wert von <math>L(\hat{y}, y) \iff</math> geringerer Unterschied zwischen prognostizierten und beobachteten Ausgaben.</li> <li>• Verlustfunktion kann auch für die komplette Trainingsmenge definiert werden.</li> </ul>
Verschwindender Gradient	Vanishing Gradient	<p>Nach <math>k</math> Durchgängen durch eine Schicht geht ein Parameter <math>w</math> als <math>w^k</math> in den Gradienten ein.</p> <ul style="list-style-type: none"> <li>• <math>w &lt; 1</math>: der Wert von <math>w^k</math> wird extrem klein und es ergibt sich ein verschwindender Gradient</li> <li>• Optimierer berücksichtigt bei kleinem <math>w</math> keine weitreichenden Zusammenhänge.</li> </ul>
Vollverbundene Schicht	Fully Connected Layer	<p>Eine Schicht eines neuronalen Netzes, welche eine lineare Transformation und anschließend eine nicht lineare Aktivierungsfunktion <math>g</math> ausführt:</p> $y = g(W * x + b)$

Konzept (deutsch)	Konzept (englisch)	Bedeutung
Wahrscheinlichkeitsvektor	Probability vector	Vektor aus reellen Zahlen: <ul style="list-style-type: none"><li>• jedes Element ist <math>\geq 0</math>.</li><li>• Die Summe aller Elemente ist 1.</li></ul> Ein Wahrscheinlichkeitsvektor der Länge $k$ enthält in der $i$ -ten Position die Wahrscheinlichkeit (Plausibilität) der $i$ -ten von $k$ Alternativen.
WLAN (wireless local area network)	WLAN, WiFi	WLAN ist die Abkürzung für wireless local area network, ein drahtloses lokales Netzwerk. Ein WLAN ermöglicht die Datenübertragung via Funkwellen, z. B. für das kabellose Surfen im Internet, oder die Vernetzung von internetfähigen Endgeräten.

---

## A.2 Verzeichnis der Abbildungen und deren Quellen

In diesem Anhang wurden die Bildnachweise für die im Buch verwendeten Fotos und Abbildungen zusammengestellt. Alle hier nicht eigens nachgewiesenen Abbildungen stammen von den Autoren und wurden größtenteils von Svenja Niehus überarbeitet. Das Copyright der selbsterstellten Abbildungen liegt bei den Autoren und Fraunhofer IAIS. Ausgenommen sind Abbildungen, die Teile von anderen Abbildungen mit Creative Commons Lizenz enthalten. Für diese Abbildungen gelten die entsprechenden Creative Commons Lizenzen der Teilbilder. Links zu den Creative Commons Bildlizenzen, die in den folgenden Bildnachweisen aufgeführt werden:

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/>

Abbildungs-Nr.	Bildnachweis
Titelbild	<a href="https://stock.adobe.com/de/images/white-cyborg-finger-about-to-touch-human-finger-3d-rendering/204325271">https://stock.adobe.com/de/images/white-cyborg-finger-about-to-touch-human-finger-3d-rendering/204325271</a>
Foto G. Paß	Aufnahme Fotoatelier Herff, Bonn. <a href="http://www.atelier-herff.de/">http://www.atelier-herff.de/</a>
Foto D. Hecker	Aufnahme Fraunhofer IAIS, 2019
Abb. 1.1	<p><b>Läufer:</b> <a href="https://www.istockphoto.com/de/foto/die-nahaufnahme-der-f%C3%BCnf-mannen-des-mannes-running-und-training-auf-laufband-gm924938754-253831773">https://www.istockphoto.com/de/foto/die-nahaufnahme-der-f%C3%BCnf-mannen-des-mannes-running-und-training-auf-laufband-gm924938754-253831773</a> Ausschnitt</p> <p><b>Treppenhaus:</b> Nilfanion: View down the spiral staircase in Haldon Belvedere, a folly on Haldon Hill, near Exeter, Devon. <a href="https://commons.wikimedia.org/wiki/File:Spiral_staircase_in_Haldon_Belvedere.jpg">https://commons.wikimedia.org/wiki/File:Spiral_staircase_in_Haldon_Belvedere.jpg</a> CC BY-SA 3.0 Ausschnitt</p> <p><b>Frau und Kind:</b> Adobe Stock: <a href="https://stock.adobe.com/de/images/mother-and-daughter-in-their-home-they-are-looking-at-coloring-book/167651778">https://stock.adobe.com/de/images/mother-and-daughter-in-their-home-they-are-looking-at-coloring-book/167651778</a> Ausschnitt</p> <p><b>Gitarrist:</b> <a href="https://en.wikipedia.org/wiki/Ben_Carey">https://en.wikipedia.org/wiki/Ben_Carey</a> Author: Justin M. Martinez, U.S. Marine Corps gemeinfrei Ausschnitt</p> <p><b>Mathematik:</b> <a href="https://de.wikipedia.org/wiki/Edward_Frenkel#/media/File:Edward_Frenkel.jpg">https://de.wikipedia.org/wiki/Edward_Frenkel#/media/File:Edward_Frenkel.jpg</a> Autor: 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 Ausschnitt</p> <p><b>Zwischenmenschlich:</b> <a href="https://de.wikipedia.org/wiki/Empathie#/media/File:There%27s_no_crying_in_baseball_(4549295140)_2.jpg">https://de.wikipedia.org/wiki/Empathie#/media/File:There%27s_no_crying_in_baseball_(4549295140)_2.jpg</a> Jax House – Diese Datei ist ein Ausschnitt aus einer anderen Datei: Jax House: There's no crying in baseball! (4549295140).jpg CC BY-SA 2.0 Ausschnitt</p> <p><b>selbststreflektiv:</b> Maria Rantanen self reflection <a href="https://www.flickr.com/photos/idhren/4176198989">https://www.flickr.com/photos/idhren/4176198989</a> CC BY-SA 2.0 Ausschnitt</p> <p><b>Gewitter:</b> Bidgee: Cloud to ground lightning strike during a dry thunderstorm near Wagga Wagga, Australia. <a href="https://en.wikipedia.org/wiki/Dry_thunderstorm#/media/File:Cloud_to_ground_lightning_strikes_south-west_of_Wagga_Wagga.jpg">https://en.wikipedia.org/wiki/Dry_thunderstorm#/media/File:Cloud_to_ground_lightning_strikes_south-west_of_Wagga_Wagga.jpg</a> CC BY-SA 3.0 au Ausschnitt.</p>
Abb. 1.3	Foto Katze: <a href="https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873">https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873</a> Ausschnitt.
Abb. 1.4	<p>Foto <b>Affe:</b> Male chimpanzee Von EBFoto <a href="https://stock.adobe.com/de/images/male-chimpanzee/103531598">https://stock.adobe.com/de/images/male-chimpanzee/103531598</a></p> <p>Foto <b>Katze:</b> <a href="https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873">https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873</a> Ausschnitt.Ausschnitt.</p> <p>Foto <b>Frau:</b> Cheerful young african woman smiling Von mimagephotos <a href="https://stock.adobe.com/de/images/cheerful-young-african-woman-smiling/56513248">https://stock.adobe.com/de/images/cheerful-young-african-woman-smiling/56513248</a> Ausschnitt</p>
Abb. 1.5	Foto Adobe Stock: <a href="https://stock.adobe.com/de/images/mother-and-daughter-in-their-home-they-are-looking-at-coloring-book/167651778">https://stock.adobe.com/de/images/mother-and-daughter-in-their-home-they-are-looking-at-coloring-book/167651778</a>
Abb. 1.6	Albert <b>Einstein</b> 1921. <a href="https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png">https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png</a> . Lizenz: Public Domain. <b>Bilder der Merkmale:</b> (Lee et al. 2011).
Abb. 1.7	Albert Einstein 1921. <a href="https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png">https://en.wikipedia.org/wiki/Albert_Einstein#/media/File:Albert_Einstein_(Nobel).png</a> . Lizenz: Public Domain.

Abbildungs-Nr.	Bildnachweis
Abb. 1.9	<a href="https://www.istockphoto.com/de/foto/transparente-fahrzeug-gm154411679-16663939">https://www.istockphoto.com/de/foto/transparente-fahrzeug-gm154411679-16663939</a>
Abb. 1.10	Auto: Oliver Kurnis Sportwagen 'Porsche 911', Modell SC 1986 <a href="https://de.wikipedia.org/wiki/Automobil#/media/File:1986_Porsche_911_SC.jpg">https://de.wikipedia.org/wiki/Automobil#/media/File:1986_Porsche_911_SC.jpg</a> CC BY 3.0 Bilder der Merkmale: (Lee et al. 2011).
Abb. 1.11	Eigene Grafik außer: Foto von <b>Turing</b> : Turing als 16-Jähriger. Gemeinfrei <a href="https://de.wikipedia.org/wiki/Alan_Turing#/media/File:Alan_Turing_Aged_16.jpg">https://de.wikipedia.org/wiki/Alan_Turing#/media/File:Alan_Turing_Aged_16.jpg</a> Expertensystem Dendral: (Lindsay et al. 1993) Foto <b>McCarthy</b> : Professor „Uncle John“ McCarthy discusses LISP, von null0 from Singapore, CC BY 2.0 <a href="https://commons.wikimedia.org/wiki/File:John_McCarthy_(computer_scientist)_Stanford_2006_(272020300).jpg?uselang=de">https://commons.wikimedia.org/wiki/File:John_McCarthy_(computer_scientist)_Stanford_2006_(272020300).jpg?uselang=de</a> Foto <b>Katze</b> /Krakenimages.com: <a href="https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873">https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873</a> Ausschnitt.
Abb. 1.12	<b>OCR</b> : <a href="https://stock.adobe.com/de/images/businessman-taking-photo-of-bill-with-mobile-phone/216276055">https://stock.adobe.com/de/images/businessman-taking-photo-of-bill-with-mobile-phone/216276055</a> <b>Schach</b> : <a href="https://stock.adobe.com/de/117711879">https://stock.adobe.com/de/117711879</a> Lokalisierung <b>Gesicht</b> : <a href="https://stock.adobe.com/de/300883540">https://stock.adobe.com/de/300883540</a> <b>Aktienkursprognose</b> : Adobe Stock: <a href="https://stock.adobe.com/de/images/stock-market-data/83882611">https://stock.adobe.com/de/images/stock-market-data/83882611</a> <b>Handschrifterkennung</b> : Eigene Grafik
Abb. 2.1	(Krizhevsky et al. 2012) mit eigener Grafik
Abb. 2.2	obere Reihe und unten links gemeinfrei. <a href="https://commons.wikimedia.org/wiki/File:Firm_red_skin_lump.jpg">https://commons.wikimedia.org/wiki/File:Firm_red_skin_lump.jpg</a> <a href="https://commons.wikimedia.org/wiki/File:Crusty_skin_lump.jpg">https://commons.wikimedia.org/wiki/File:Crusty_skin_lump.jpg</a> <a href="https://commons.wikimedia.org/w/index.php?title=Special:Search&amp;limit=50&amp;offset=0&amp;profile=default&amp;search=skin+cancer&amp;advancedSearch-current={{}}&amp;ns0=1&amp;ns6=1&amp;ns12=1&amp;ns14=1&amp;ns100=1&amp;ns106=1#/media/File:Melanoma_with_color_differences.jpg">https://commons.wikimedia.org/w/index.php?title=Special:Search&amp;limit=50&amp;offset=0&amp;profile=default&amp;search=skin+cancer&amp;advancedSearch-current={{}}&amp;ns0=1&amp;ns6=1&amp;ns12=1&amp;ns14=1&amp;ns100=1&amp;ns106=1#/media/File:Melanoma_with_color_differences.jpg</a> <a href="https://en.wikipedia.org/wiki/Nevus#/media/File:Spitz_nevus.jpg">https://en.wikipedia.org/wiki/Nevus#/media/File:Spitz_nevus.jpg</a> Unten Mitte Jmarchn Becker's nevus, right shoulder <a href="https://en.wikipedia.org/wiki/Nevus#/media/File:Becker's_nevus_of_shoulder.JPG">https://en.wikipedia.org/wiki/Nevus#/media/File:Becker's_nevus_of_shoulder.JPG</a> CC BY-SA-3.0. Unten rechts Mathew Bellemare <a href="https://commons.wikimedia.org/wiki/File:51_105_Leg_Nevus_(149029661).jpeg">https://commons.wikimedia.org/wiki/File:51_105_Leg_Nevus_(149029661).jpeg</a> CC BY-SA 3.0
Abb. 2.3	<a href="https://deepmind.com/blog/article/alphafold">https://deepmind.com/blog/article/alphafold</a>
Abb. 2.5	<a href="https://www.istockphoto.com/de/foto/3d-rendering-piktogramm-spracherkennungssystem-von-blauem-grund-gm683287514-125386755">https://www.istockphoto.com/de/foto/3d-rendering-piktogramm-spracherkennungssystem-von-blauem-grund-gm683287514-125386755</a>
Abb. 2.7	Photos: © WTO/ Cuika Foto. World Trade Organization Heads of Delegation meeting, 11 December <a href="https://de.wikipedia.org/wiki/Simultandolmetschen#/media/File:Heads_of_Delegation_meeting,_11_December_(38130168095).jpg">https://de.wikipedia.org/wiki/Simultandolmetschen#/media/File:Heads_of_Delegation_meeting,_11_December_(38130168095).jpg</a> CC BY-SA-2.0 Ausschnitt
Abb. 2.8	By Rosemaryetoufee „Watson Jeopardy demo“ <a href="https://kids.kiddle.co/Image:Watson_Jeopardy_demo.jpg">https://kids.kiddle.co/Image:Watson_Jeopardy_demo.jpg</a> CC BY-SA 4.0 Ausschnitt

Abbildungs-Nr.	Bildnachweis
Abb. 2.9	Myrabella <a href="https://de.wikipedia.org/wiki/Schlacht_bei_Hastings#/media/Datei:Bayeux_Tapestry_scene55_William_Hastings_battlefield.jpg">https://de.wikipedia.org/wiki/Schlacht_bei_Hastings#/media/Datei:Bayeux_Tapestry_scene55_William_Hastings_battlefield.jpg</a> gemeinfrei
Abb. 2.12	IBM Research IBM Project Debater with Dan Zafrir <a href="https://www.flickr.com/photos/ibm_research_zurich/45818968004">https://www.flickr.com/photos/ibm_research_zurich/45818968004</a> CC BY-ND 2.0
Abb. 2.13	Jürg Vollmer Schachweltmeister Garri Kasparov aus Baku/Aserbeidschan 26.08.2009. CC BY-SA 2.0. <a href="https://www.flickr.com/photos/maiakinfo/3859735342/">https://www.flickr.com/photos/maiakinfo/3859735342/</a>
Abb. 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. <a href="https://www.flickr.com/photos/erikbenson/25717574115/">https://www.flickr.com/photos/erikbenson/25717574115/</a>
Abb. 2.15	<a href="https://stock.adobe.com/de/images/people-play-poker-at-the-table-in-the-casino/294179973">https://stock.adobe.com/de/images/people-play-poker-at-the-table-in-the-casino/294179973</a>
Abb. 2.16	Eigene Snapshots vom public domain Stella Emulator
Abb. 2.17	Ausschnitt aus <a href="https://www.theverge.com/2018/7/4/17533898/deepmind-ai-agent-video-game-quake-iii-capture-the-flag">https://www.theverge.com/2018/7/4/17533898/deepmind-ai-agent-video-game-quake-iii-capture-the-flag</a>
Abb. 2.18	<a href="https://www.theverge.com/2018/8/6/17655086/dota2-openai-bots-professional-gaming-ai">https://www.theverge.com/2018/8/6/17655086/dota2-openai-bots-professional-gaming-ai</a>
Abb. 2.19	<a href="https://commons.wikimedia.org/wiki/File:BeerBottlePass.JPG">https://commons.wikimedia.org/wiki/File:BeerBottlePass.JPG</a> Gemeinfrei
Abb. 2.20	Anthony Kendall This is VW's entry in the Darpa Grand Challenge. <a href="https://www.flickr.com/photos/80546870@N00/87625236">https://www.flickr.com/photos/80546870@N00/87625236</a> CC BY-SA 2.0
Abb. 2.21	Adobe Stock/Andrei <a href="https://www.istockphoto.com/de/foto/waymo-selbstfahrendes-auto-f%C3%BCr-chrt-tests-auf-einer-str%C3%A9e-in-der-n%C3%A4he-von-googles-b%C3%BCro-gm1162994767-319188671">https://www.istockphoto.com/de/foto/waymo-selbstfahrendes-auto-f%C3%BCr-chrt-tests-auf-einer-str%C3%A9e-in-der-n%C3%A4he-von-googles-b%C3%BCro-gm1162994767-319188671</a>
Abb. 2.22	<a href="https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview">https://www.theverge.com/2018/5/9/17307156/google-waymo-driverless-cars-deep-learning-neural-net-interview</a>
Abb. 2.23	<a href="https://de.wikipedia.org/wiki/Portrait_of_Edmond_de_Blamy#/media/Datei:Edmond_de_Blamy.png">https://de.wikipedia.org/wiki/Portrait_of_Edmond_de_Blamy#/media/Datei:Edmond_de_Blamy.png</a> gemeinfrei
Abb. 2.24	(Karras 2019) <a href="https://thispersondoesnotexist.com/public_domain">https://thispersondoesnotexist.com/public_domain</a> . See <a href="https://news.ycombinator.com/item?id=19144280">https://news.ycombinator.com/item?id=19144280</a>
Abb. 3.1	<a href="https://www.istockphoto.com/de/foto/eine-gruppe-von-studenten-auf-die-anatomie-klasse-in-hoch-schule-gm519589014-90619453">https://www.istockphoto.com/de/foto/eine-gruppe-von-studenten-auf-die-anatomie-klasse-in-hoch-schule-gm519589014-90619453</a>
Abb. 3.2	„Defying Gravity“ by Andrew Newill is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/157c3159-14ef-40ff-bdc1-c394ea5006ac">https://ccsearch.creativecommons.org/photos/157c3159-14ef-40ff-bdc1-c394ea5006ac</a>
Abb. 3.3	<a href="https://stock.adobe.com/de/search?serie_id=206970868&amp;asset_id=206970923">https://stock.adobe.com/de/search?serie_id=206970868&amp;asset_id=206970923</a>
Abb. 3.4	Eigene Grafik Haus: <a href="https://www.pexels.com/photo/home-real-estate-106399/">https://www.pexels.com/photo/home-real-estate-106399/</a> free to use.
Abb. 3.7	Selbst produzierte Grafik aus MNIST-Daten
Abb. 3.8	<a href="https://www.researchgate.net/figure/Selected-difficult-examples-from-the-MNIST-database_fig2_220914543">https://www.researchgate.net/figure/Selected-difficult-examples-from-the-MNIST-database_fig2_220914543</a>
Abb. 3.10	Selbst produzierte Grafik aus MNIST-Daten
Abb. 3.22	Edna Winti Above it all CC BY 2.0 <a href="https://www.flickr.com/photos/ednawinti/45068960405/">https://www.flickr.com/photos/ednawinti/45068960405/</a>
Abb. 4.3	alles eigene Grafiken außer: Perceptron: The Mark I Perceptron, an early pattern recognition system, at the Cornell Aeronautical Laboratory. Fair Use license. <a href="https://en.wikipedia.org/wiki/Perceptron#/media/File:Mark_I_perceptron.jpeg">https://en.wikipedia.org/wiki/Perceptron#/media/File:Mark_I_perceptron.jpeg</a> Krizhevsky: Nvidia Tesla 2075 Raysonho @ Open Grid Scheduler/Grid Engine – Eigenes Werk CC0 1.0 <a href="https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG">https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG</a>

Abbildungs-Nr.	Bildnachweis
Abb. 4.5	By Sethwoodworth Marvin Minsky was visiting the OLPC offices and picked up a Firefox wrist band. <a href="https://en.wikipedia.org/wiki/Marvin_Minsky#/media/File:Marvin_Minsky_at OLPCb.jpg">https://en.wikipedia.org/wiki/Marvin_Minsky#/media/File:Marvin_Minsky_at OLPCb.jpg</a> CC BY 3.0
Abb. 4.10	Eigene Grafik. Haus: <a href="https://stock.adobe.com/de/290172089">https://stock.adobe.com/de/290172089</a>
Abb. 4.13	Eviatar Bach: „Geoffrey Hinton giving a lecture about deep neural networks at the University of British Columbia“ <a href="https://en.wikipedia.org/wiki/Geoffrey_Hinton#/media/File:Geoffrey_Hinton_at_UBC.jpg">https://en.wikipedia.org/wiki/Geoffrey_Hinton#/media/File:Geoffrey_Hinton_at_UBC.jpg</a> Ausschnitt. CC BY-SA 3.0
Abb. 4.14	<a href="https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG">https://de.wikipedia.org/wiki/Nvidia_Tesla#/media/Datei:NvidiaTesla2075.JPG</a> CC0 gemeinfrei
Abb. 4.21	Gedruckte Buchstaben aus dem notMNIST-Datensatz. <a href="https://www.kaggle.com/jwjohnson314/notmnist CC0">https://www.kaggle.com/jwjohnson314/notmnist CC0</a> Public Domain
Abb. 4.27	Eigene Grafik. Schachbild <a href="https://www.istockphoto.com/de/foto/opening-game-of-chess-gm586376700-100670363">https://www.istockphoto.com/de/foto/opening-game-of-chess-gm586376700-100670363</a>
Abb. 4.31	AnotherSamWilson Bayesian optimization of a function (black) with Gaussian processes (purple). Three acquisition functions (blue) are shown at the bottom. <a href="https://en.wikipedia.org/wiki/Bayesian_optimization#/media/File:GpParBayesAnimationSmall.gif">https://en.wikipedia.org/wiki/Bayesian_optimization#/media/File:GpParBayesAnimationSmall.gif</a> CC BY-SA 4.0
Abb. 4.33	(Noy et al. 2020)
Abb. 4.34	<a href="https://de.wikipedia.org/wiki/Neuronales_Netz#/media/Datei:Neurons_big1.jpg">https://de.wikipedia.org/wiki/Neuronales_Netz#/media/Datei:Neurons_big1.jpg</a> Gemeinfrei
Abb. 5.1	Eigene Umrandungen Fotos: <a href="https://www.istockphoto.com/de/foto/bengal-kitten-in-blumenwiese-gm905117504-249578201">https://www.istockphoto.com/de/foto/bengal-kitten-in-blumenwiese-gm905117504-249578201</a> <a href="https://stock.adobe.com/de/images/domestic-animals-chicken-dog-and-cat-eating-together-as-best-friend/221661292">https://stock.adobe.com/de/images/domestic-animals-chicken-dog-and-cat-eating-together-as-best-friend/221661292</a>
Abb. 5.2	Gehirn: _DJ_ human brain on white background CC BY-SA 2.0 <a href="https://www.flickr.com/photos/flamephoenix1991/8376271918/">https://www.flickr.com/photos/flamephoenix1991/8376271918/</a> Auge: Menschliches Auge <a href="https://www.istockphoto.com/de/foto/menschliches-auge-gm91170533-6467713">https://www.istockphoto.com/de/foto/menschliches-auge-gm91170533-6467713</a>
Abb. 5.3	Eigene Grafik: Katze: <a href="https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873">https://stock.adobe.com/de/images/beautiful-short-hair-cat-lying-on-the-bed-at-home/247439873?asset_id=247439873</a>
Abb. 5.4	Kirsch: <a href="https://en.wikipedia.org/wiki/Russell_A._Kirsch#/media/File:NBSFirstScanImage.jpg">https://en.wikipedia.org/wiki/Russell_A._Kirsch#/media/File:NBSFirstScanImage.jpg</a> public domain Viola&Jones: <a href="https://de.wikipedia.org/wiki/Viola-Jones-Methode#/media/Datei:Face_detection.jpg">https://de.wikipedia.org/wiki/Viola-Jones-Methode#/media/Datei:Face_detection.jpg</a> 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 Leopard: <a href="https://ccsearch.creativecommons.org/photos/e828b6d0-a257-406a-9e0a-6e99ccaa41ab">https://ccsearch.creativecommons.org/photos/e828b6d0-a257-406a-9e0a-6e99ccaa41ab</a> „EA0A7014 Kopie“ by harald 65 is licensed under CC BY 2.0

Abbildungs-Nr.	Bildnachweis
Abb. 5.5	Katze1: „20170708_Apotheke_004“ by Tauralbus is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/de285ab1-1f4b-4207-9b82-d21041f21a44">https://ccsearch.creativecommons.org/photos/de285ab1-1f4b-4207-9b82-d21041f21a44</a> , Katze2: „20170708_Apotheke_001“ by Tauralbus is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/d4678450-efd4-439f-9bcb-68a30e796699">https://ccsearch.creativecommons.org/photos/d4678450-efd4-439f-9bcb-68a30e796699</a> , Katze3: „Miezi“ by Tobi NDH is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/da7bef85-344f-466d-8b48-ab1c2e0d6c73">https://ccsearch.creativecommons.org/photos/da7bef85-344f-466d-8b48-ab1c2e0d6c73</a> , Katze4: „DSCF4280a – Kopie“ by wooz is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/8758d5c6-73ca-465f-b11d-557e8c9e3016">https://ccsearch.creativecommons.org/photos/8758d5c6-73ca-465f-b11d-557e8c9e3016</a> , Katze5: „Katzenerinnerungsfoto 1/3“ by avatar-1 is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/498cd7c8-31bc-4e6b-b06d-88d96ffe23a">https://ccsearch.creativecommons.org/photos/498cd7c8-31bc-4e6b-b06d-88d96ffe23a</a> , Katze6: Photo of a cat wrapped in a blanket Von Pavlo <a href="https://stock.adobe.com/de/images/photo-of-a-cat-wrapped-in-a-blanket/189043951">https://stock.adobe.com/de/images/photo-of-a-cat-wrapped-in-a-blanket/189043951</a> Brücke1: „chain bridge Budapest“ by Namor Trebat is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/94c9f51a-66d0-4131-95a5-01d395b0c329">https://ccsearch.creativecommons.org/photos/94c9f51a-66d0-4131-95a5-01d395b0c329</a> , Brücke2: „Nocni-most-03“ by WonderfulSnaps.com is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/5831d1f8-2a46-43a4-8f9e-2a83f387ba95">https://ccsearch.creativecommons.org/photos/5831d1f8-2a46-43a4-8f9e-2a83f387ba95</a> , Brücke3: „Bridge“ by Jun_nor is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/3bbdfb98-e54c-4108-8317-26defc777f76">https://ccsearch.creativecommons.org/photos/3bbdfb98-e54c-4108-8317-26defc777f76</a> , Brücke4: „London Bridge“ by Sean MacEntee is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/3164cb8e-6a1b-4caa-8868-1ed3070e30fd">https://ccsearch.creativecommons.org/photos/3164cb8e-6a1b-4caa-8868-1ed3070e30fd</a> , Brücke5: „Tower Bridge“ by longplay is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/3f8616c4-1c7e-4b51-b128-c59f0d33ad95">https://ccsearch.creativecommons.org/photos/3f8616c4-1c7e-4b51-b128-c59f0d33ad95</a> , Brücke6: „The Sydney Harbour Bridge“ by jonisanowl is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/412d3ce5-a7c9-4eec-9b07-ff8157ea6e4a">https://ccsearch.creativecommons.org/photos/412d3ce5-a7c9-4eec-9b07-ff8157ea6e4a</a>
Abb. 5.7	<a href="https://de.wikipedia.org/wiki/Laplace-Filter#/media/Datei:Laplace_beispiel.png">https://de.wikipedia.org/wiki/Laplace-Filter#/media/Datei:Laplace_beispiel.png</a> gemeinfrei
Abb. 5.14	<a href="https://en.wikipedia.org/wiki/Yann_LeCun#/media/File:Yann_LeCun_-_2018_(cropped).jpg">https://en.wikipedia.org/wiki/Yann_LeCun#/media/File:Yann_LeCun_-_2018_(cropped).jpg</a> Ausschnitt. „Conference of French computer scientist Yann LeCun, director of Facebook AI Research, at the École Polytechnique“ Jérémie Barande CC BY-SA 2.0
Abb. 5.15	(Wu et al. 2018)
Abb. 5.18	(He 2016)
Abb. 5.17	Eigene Grafik (Li et al. 2018).
Abb. 5.20	Eigene Grafik. frau: <a href="https://stock.adobe.com/de/images/cheerful-young-african-woman-smiling/56513248">https://stock.adobe.com/de/images/cheerful-young-african-woman-smiling/56513248</a>
Abb. 5.21	(Huang et al. 2017) eigene Grafik. Pferd: <a href="https://stock.adobe.com/de/images/pferd-weisser-hintergrund/113638638">https://stock.adobe.com/de/images/pferd-weisser-hintergrund/113638638</a>
Abb. 5.22	(Touvron et al. 2019) mit Übersetzungen
Abb. 5.23	(Zeiler und Fergus 2014)
Abb. 5.24	(Zeiler und Fergus 2014)
Abb. 5.25	(Krizhevsky et al. 2012)

Abbildungs-Nr.	Bildnachweis
Abb. 5.26	(Van Horn et al. 2018)
Abb. 5.27	(Esteva et al. 2017).
Abb. 5.28	(Grover 2018)
Abb. 5.29	(Redmon und Farhadi 2017)
Abb. 5.30	(Chen 2018 S. 841)
Abb. 5.32	(Ronneberger et al. 2015), editiert
Abb. 5.34	(Zhu et al. 2019)
Abb. 5.35	(Eslami et al. 2018)
Abb. 5.36	(Phillips et al. 2018)
Abb. 5.37	(Wang et al. 2019)
Abb. 5.38	Eigene Grafik <b>Hund1:</b> „Buffy 3“ by Ethernurn is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/a15e6ae8-64d9-4f81-a612-eb8a97f67e08">https://ccsearch.creativecommons.org/photos/a15e6ae8-64d9-4f81-a612-eb8a97f67e08</a> Ausschnitt <b>Hund2:</b> „2011-10-23 um 15-46-41 – Hund macht Pause“ by torstenbehrrens is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/febe6ff8-2d75-46d8-873f-a17345e2c1f2">https://ccsearch.creativecommons.org/photos/febe6ff8-2d75-46d8-873f-a17345e2c1f2</a> Ausschnitt <b>Hund3:</b> „_MG_0274“ by jenefeldt is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/9888d873-d1f4-43ad-9e39-fac9a356f9cf">https://ccsearch.creativecommons.org/photos/9888d873-d1f4-43ad-9e39-fac9a356f9cf</a> <b>Katze3:</b> Bertilvidet~commonswiki Van Cat, male kitten CC BY 2.5 <a href="https://tr.wikipedia.org/wiki/Dosya:Turkish_Van_Cat.jpg">https://tr.wikipedia.org/wiki/Dosya:Turkish_Van_Cat.jpg</a>
Abb. 5.39	Eigene Grafik: Foto <b>Katze1:</b> „Just Looking“ by Lisa Zins is licensed under CC BY 2.0. <a href="https://ccsearch.creativecommons.org/photos/4eb69cd1-bb0b-42f9-bfdf-7c5c844d1ffa">https://ccsearch.creativecommons.org/photos/4eb69cd1-bb0b-42f9-bfdf-7c5c844d1ffa</a> Ausschnitt <b>Katze2:</b> „IMG_8622“ by dietzwolfi is licensed under CC BY 2.0 cc-icon <a href="https://ccsearch.creativecommons.org/photos/f329a21b-2b3c-4c13-ab80-312ff0509f97">https://ccsearch.creativecommons.org/photos/f329a21b-2b3c-4c13-ab80-312ff0509f97</a> Ausschnitt, <b>Katze3:</b> „Miezekatze“ by chris-sy is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/d859641d-24a4-447e-b571-ab4b23c7796c">https://ccsearch.creativecommons.org/photos/d859641d-24a4-447e-b571-ab4b23c7796c</a> <b>Hund1:</b> „Buffy 3“ by Ethernurn is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/a15e6ae8-64d9-4f81-a612-eb8a97f67e08">https://ccsearch.creativecommons.org/photos/a15e6ae8-64d9-4f81-a612-eb8a97f67e08</a> Ausschnitt <b>Hund2:</b> „2011-10-23 um 15-46-41 – Hund macht Pause“ by torstenbehrrens is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/febe6ff8-2d75-46d8-873f-a17345e2c1f2">https://ccsearch.creativecommons.org/photos/febe6ff8-2d75-46d8-873f-a17345e2c1f2</a> Ausschnitt
Abb. 5.42	(Teye et al. 2018)
Abb. 5.43	(Teye et al. 2018)
Abb. 5.44	(Hendrycks und Dietterich 2019)
Abb. 5.45	(Hendrycks und Dietterich 2019)
Abb. 5.46	(Eykholt et al. 2018)
Abb. 5.47	(Eykholt et al. 2018)
Abb. 5.48	(Szegedy et al. 2014)
Abb. 6.1	Bovee und Thill „Business Technology“ <a href="https://www.flickr.com/photos/52883612@N05/39475182975">https://www.flickr.com/photos/52883612@N05/39475182975</a> Lizenz: CC BY 2.0
Abb. 6.4	Eigene Grafik. <a href="http://www.trivial.io/word2vec-on-databricks/">http://www.trivial.io/word2vec-on-databricks/</a>

Abbildungs-Nr.	Bildnachweis
Abb. 6.5	<p>Text  Evan-Amos An orange, whole and split <a href="https://en.wikipedia.org/wiki/Orange_(fruit)#/media/File:Orange-Whole-&amp;-Split.jpg">https://en.wikipedia.org/wiki/Orange_(fruit)#/media/File:Orange-Whole-&amp;-Split.jpg</a> CC BY-SA 3.0  Almu – Naranja de gran tamaño cosechada en el Bajo Andarax, Almería.  <a href="https://es.wikipedia.org/wiki/Naranja#/media/Archivo:Naranja_Almeria.JPG">https://es.wikipedia.org/wiki/Naranja#/media/Archivo:Naranja_Almeria.JPG</a>  CC BY-SA 4.0  Evan-Amos Lemon external surface and cross-section <a href="https://en.wikipedia.org/wiki/Lemon#/media/File:Lemon-Whole-Split.jpg">https://en.wikipedia.org/wiki/Lemon#/media/File:Lemon-Whole-Split.jpg</a> public domain</p>
Abb. 6.8	Jérémie Barande, <a href="https://fr.wikipedia.org/wiki/Yoshua_Bengio#/media/Fichier:Yoshua_Bengio_-_2017.jpg">https://fr.wikipedia.org/wiki/Yoshua_Bengio#/media/Fichier:Yoshua_Bengio_-_2017.jpg</a> CC BY-SA-2.0
Abb. 6.18	<p>Noten: <a href="https://www.cantus.org/notenarchiv/files/d872-4.pdf">https://www.cantus.org/notenarchiv/files/d872-4.pdf</a>, gemeinfrei  DNA: <a href="https://stock.adobe.com/de/images/digital-illustration-of-a-dna-model-3d-rendering/140436292">https://stock.adobe.com/de/images/digital-illustration-of-a-dna-model-3d-rendering/140436292</a>  Ton: <a href="https://stock.adobe.com/de/images/blue-audio-waveform-background/85619893">https://stock.adobe.com/de/images/blue-audio-waveform-background/85619893</a></p>
Abb. 6.24	ITU/R.Farrell <a href="https://commons.wikimedia.org/wiki/File:J%C3%BCrgen_Schmidhuber.jpg">https://commons.wikimedia.org/wiki/File:J%C3%BCrgen_Schmidhuber.jpg</a> CC BY 2.0
Abb. 6.31	Text. Quelle: <a href="https://de.wikipedia.org/wiki/Joseph_von_Fraunhofer">https://de.wikipedia.org/wiki/Joseph_von_Fraunhofer</a>
Abb. 6.33	Stock Market Data Von James Thew <a href="https://stock.adobe.com/de/images/stock-market-data/83882611">https://stock.adobe.com/de/images/stock-market-data/83882611</a>
Abb. 6.34	(Strobelt et al. 2016)
Abb. 6.38	<i>Text</i> (Sutskever et al. 2014)
Abb. 6.39	<i>Tabelle</i> : (Sutskever et al. 2014)
Abb. 6.40	Eigene Grafik Quelle: (Sutskever et al. 2014)
Abb. 6.42	Eigene Grafik: <a href="https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/">https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-3/</a>
Abb. 6.43	Tabelle: Quelle: (Bahdanau et al. 2015)
Abb. 6.44	Quelle: (Bahdanau et al. 2015),
Abb. 6.45	Quelle: <a href="http://wmt.ufal.cz/">http://wmt.ufal.cz/</a>
Abb. 6.46	(Britz et al. 2017)
Abb. 6.47	Eigene Grafik: Quelle: <a href="https://ricardokleinklein.github.io/2017/11/16/Attention-is-all-you-need.html">https://ricardokleinklein.github.io/2017/11/16/Attention-is-all-you-need.html</a> Quelle: (Vaswani et al. 2017)
Abb. 6.50	Eigene Grafik: (Vaswani et al. 2017)
Abb. 6.52	Text: (Sennrich et al. 2016)
Abb. 6.54	Eigene Grafik. Quelle: <a href="https://ricardokleinklein.github.io/2017/11/16/Attention">https://ricardokleinklein.github.io/2017/11/16/Attention</a>
Abb. 6.55	(Popel und Bojar 2018)
Abb. 6.56	Businessman standing and holding speaker to speak in the auditorium. Quelle <a href="https://stock.adobe.com/de/285635048">https://stock.adobe.com/de/285635048</a>
Abb. 6.57	Eigene Grafik: Quelle: (Jia et al. 2019)
Abb. 6.64	Tabelle (MultiNLI 2017)
Abb. 6.65	Tabelle (Devlin et al. 2018)
Abb. 6.66	Quelle: <a href="https://en.wikipedia.org/wiki/Genghis_Khan">https://en.wikipedia.org/wiki/Genghis_Khan</a> gemeinfrei
Abb. 6.67	<i>California National Guard: California's Soldiers and Airmen help extinguish raging wildfire. 24.08.2012. CC BY 2.0.</i> <a href="https://www.flickr.com/photos/caguard/7880896072">https://www.flickr.com/photos/caguard/7880896072</a>
Abb. 6.69	<a href="https://en.wikipedia.org/wiki/Robin_Spielberg#/media/File:Robin_Spielberg.jpg">https://en.wikipedia.org/wiki/Robin_Spielberg#/media/File:Robin_Spielberg.jpg</a> Photograph of Robin Spielberg Larry Kosson CC BY-SA 3.0

Abbildungs-Nr.	Bildnachweis
Abb. 6.71	(Kudugunta et al. 2019)
Abb. 6.72	<i>Eigene grafik:</i> (Shazeer et al. 2017).
Abb. 6.73	<a href="https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html">https://ai.googleblog.com/2019/10/exploring-massively-multilingual.html</a>
Abb. 6.74	Eigene Grafik (Nayak 2019)
Abb. 6.75	<a href="http://cocodataset.org/#explore?id=195696">http://cocodataset.org/#explore?id=195696</a> , <a href="http://cocodataset.org/#termsofuse">http://cocodataset.org/#termsofuse</a>
Abb. 6.76	<i>Eigene Grafik: Bild: DJHeini Some friends playing frisbee on the beach.</i> <i>IMG_0531</i> <a href="https://www.flickr.com/photos/74837817@N00/4880595155CC">https://www.flickr.com/photos/74837817@N00/4880595155CC</a> BY-SA 2.0
Abb. 6.77	(Sharma et al. 2018)
Abb. 6.80	<i>Eigene Grafik:</i> (Ribeiro et al. 2016)
Abb. 6.81	(Lapuschkin et al. 2019)
Abb. 6.82	Tabelle: (Karpukhin et al. 2019)
Abb. 6.84	Eigene Grafik: <a href="https://gluebenchmark.com/leaderboard">https://gluebenchmark.com/leaderboard</a>
Abb. 7.2	Rburtonresearch This is a picture of a wave file captioned with pressure and time as the ,X‘ and ,Y‘ coordinates. <a href="https://en.wikipedia.org/wiki/Sound#/media/File:The_Elements_of_Sound_jpg.jpg">https://en.wikipedia.org/wiki/Sound#/media/File:The_Elements_of_Sound_jpg.jpg</a> übersetzt CC BY-SA 4.0
Abb. 7.3	<a href="https://de.wikipedia.org/wiki/Fourier-Analysis#/media/Datei:Fourier_transform_time_and_frequency_domains_(small).gif">https://de.wikipedia.org/wiki/Fourier-Analysis#/media/Datei:Fourier_transform_time_and_frequency_domains_(small).gif</a> gemeinfrei
Abb. 7.6	Quelle: <a href="https://archive.org/details/SwitchboardCorpusSample">https://archive.org/details/SwitchboardCorpusSample</a>
Abb. 7.7	Dragon: <a href="https://de.wikipedia.org/wiki/Dragon_NaturallySpeaking#/media/Datei:Dragon_Naturally_Speaking_Logo.png">https://de.wikipedia.org/wiki/Dragon_NaturallySpeaking#/media/Datei:Dragon_Naturally_Speaking_Logo.png</a> gemeinfrei Amazon Echo: Michael Sheehan The black Amazon Echo Dot (second generation) sitting idle on a wood surface. <a href="https://www.flickr.com/photos/hightechdad/33309161691/">https://www.flickr.com/photos/hightechdad/33309161691/</a> CC BY 2.0 Ibm shoebox: <a href="https://www.wired.com/2013/11/tech-time-warp-of-the-week-ibms-speech-savvy-computer-1986/">https://www.wired.com/2013/11/tech-time-warp-of-the-week-ibms-speech-savvy-computer-1986/</a> Tangora: <a href="https://www.wired.com/2013/11/tech-time-warp-of-the-week-ibms-speech-savvy-computer-1986/">https://www.wired.com/2013/11/tech-time-warp-of-the-week-ibms-speech-savvy-computer-1986/</a> <a href="https://stackoverflow.com/questions/35169706/audio-feature-for-deep-learning/39624704">https://stackoverflow.com/questions/35169706/audio-feature-for-deep-learning/39624704</a> Tangora and Shoebox Courtuosity of IBM.
Abb. 7.9	Quelle: (Chan et al. 2016)
Abb. 7.12	Eigene Grafik. Quelle: (Li et al. 2019)
Abb. 7.15	Quelle: <a href="https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html">https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html</a>
Abb. 7.16	Eigene Grafik: (Chung et al. 2017)
Abb. 7.17	Quelle: <a href="http://en.wikipedia.org/wiki/File:Signal_Sampling.png">http://en.wikipedia.org/wiki/File:Signal_Sampling.png</a> gemeinfrei
Abb. 7.21	Quelle: „Happy woman talking on the phone“ by Rawpixel Ltd <a href="https://www.flickr.com/photos/147875007@N03/30849323977">https://www.flickr.com/photos/147875007@N03/30849323977</a> was cropped and is licensed under CC BY 2.0
Abb. 7.22	Quellen: (links) Maurizio Pesce „Android Assistant on the Google Pixel XL smartphone“ <a href="https://www.flickr.com/photos/pestoverde/29526761674/">https://www.flickr.com/photos/pestoverde/29526761674/</a> CC BY 2.0. (Mitte): Frmorrison at English Wikipedia „This is the Amazon Echo appliance that is basically a talking speaker“ <a href="https://en.wikipedia.org/wiki/Virtual_assistant#/media/File:Amazon_Echo.jpg">https://en.wikipedia.org/wiki/Virtual_assistant#/media/File:Amazon_Echo.jpg</a> CC BY-SA 3.0. (rechts) „Ausblick auf iOS 7“ by FalkoMD is licensed under CC BY-SA 2.0 <a href="https://www.flickr.com/photos/95190793@N08/9339721523">https://www.flickr.com/photos/95190793@N08/9339721523</a>

Abbildungs-Nr.	Bildnachweis
Abb. 7.25	Tabelle. Quelle: <a href="https://web.stanford.edu/~jurafsky/ws97/manual.august1.html">https://web.stanford.edu/~jurafsky/ws97/manual.august1.html</a>
Abb. 7.28	Quelle: <a href="https://arxiv.org/pdf/1609.06782.pdf">https://arxiv.org/pdf/1609.06782.pdf</a>
Abb. 7.29	Quelle: (Monfort et al. 2019)
Abb. 7.30	Bild aus Video: <a href="https://www.youtube.com/watch?v=x9AhZLDkbyc">https://www.youtube.com/watch?v=x9AhZLDkbyc</a>
Abb. 7.31	<i>Eigene Grafik: Bilder aus Video Henry Stober The Big 5CC BY-SA 3.0</i> <a href="https://vimeo.com/239953264">https://vimeo.com/239953264</a>
Abb. 7.32	<i>Eigene Grafik. Bilder aus Video Quelle: Henry Stober The Big 5CC BY-SA 3.0</i> <a href="https://vimeo.com/239953264">https://vimeo.com/239953264</a>
Abb. 7.33	(Simonyan und Zisserman 2014)
Abb. 7.34	<i>Eigene Grafik</i> (Zhou et al. 2018)
Abb. 7.35	(Zhou et al. 2018)
Abb. 7.36	(Sun et al. 2019)
Abb. 7.37	(Sun et al. 2019)
Abb. 7.38	(Schönherr et al. 2019).
Abb. 8.1	<a href="https://www.istockphoto.com/de/foto/junge-w%C3%A4rmte-seine-h%C3%A4nde-in-der-n%C3%A4he-von-lagerfeuer-im-wald-gm670835208-122726195">https://www.istockphoto.com/de/foto/junge-w%C3%A4rmte-seine-h%C3%A4nde-in-der-n%C3%A4he-von-lagerfeuer-im-wald-gm670835208-122726195</a>
Abb. 8.2	Schach: <a href="https://commons.wikimedia.org/wiki/File:Chess_pieces_bokeh.jpg">https://commons.wikimedia.org/wiki/File:Chess_pieces_bokeh.jpg</a> Chess pieces with colorful bokeh Mukumbura CC BY-SA 2.0. Seaquest: Eigene Snapshots vom public domain Stella Emulator
Abb. 8.3	Quelle: <a href="https://www.istockphoto.com/de/foto/frau-mit-smartphone-in-autonome-auto-selbstfahrer-fahrzeug-autopilot-kfz-technik-gm829192848-134881835">https://www.istockphoto.com/de/foto/frau-mit-smartphone-in-autonome-auto-selbstfahrer-fahrzeug-autopilot-kfz-technik-gm829192848-134881835</a> Andyhill8 – Andy Hill <a href="https://en.wikipedia.org/wiki/Stockbroker#/media/File:Stockbroker.jpg">https://en.wikipedia.org/wiki/Stockbroker#/media/File:Stockbroker.jpg</a> public domain
Abb. 8.4	Eigene Grafik Börse: Stock Market Data Von James Thew. <a href="https://t3.ftcdn.net/jpg/00/83/88/26/240_F_83882611_0W8qxnDlILQGfLDsXONvlwMD3NSMPmzF.jpg">https://t3.ftcdn.net/jpg/00/83/88/26/240_F_83882611_0W8qxnDlILQGfLDsXONvlwMD3NSMPmzF.jpg</a> Tastatur: Adobe Stock/hansgeel <a href="https://t3.ftcdn.net/jpg/01/57/68/92/240_F_157689265_R2Y24NgjCi7xx7iBLNg9aecbpkiJLtgC.jpg">https://t3.ftcdn.net/jpg/01/57/68/92/240_F_157689265_R2Y24NgjCi7xx7iBLNg9aecbpkiJLtgC.jpg</a>
Abb. 8.5	Snarc: <a href="http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/">http://cyberneticzoo.com/mazesolvers/1951-maze-solver-minsky-edmonds-american/</a> Shakey: SRI International Shakey in 1972 CC BY-SA 3.0 <a href="https://en.wikipedia.org/wiki/Shakey_the_robot#/media/File:SRI_Shakey_with_callouts.jpg">https://en.wikipedia.org/wiki/Shakey_the_robot#/media/File:SRI_Shakey_with_callouts.jpg</a> Dickmann: <a href="https://www.lifehacker.com.au/2016/02/creator-of-the-worlds-first-self-driving-cars-ernst-dickmanns/">https://www.lifehacker.com.au/2016/02/creator-of-the-worlds-first-self-driving-cars-ernst-dickmanns/</a> Backgammon: Bdevel, took this picture on Sept. 19 2005 and grant its use under the GNU Free Documentation License. <a href="https://de.wikipedia.org/wiki/Datei:Backgammon_board.jpg">https://de.wikipedia.org/wiki/Datei:Backgammon_board.jpg</a> Go: <a href="https://stock.adobe.com/de/editorial/chinese-go-player-ke-jie-reacts-during-his-first-match-with-google-s-artificial-intelligence-program-alphago-at-the-future-of-go-summit-in-wuzhen/155279016?prev_url=detail">https://stock.adobe.com/de/editorial/chinese-go-player-ke-jie-reacts-during-his-first-match-with-google-s-artificial-intelligence-program-alphago-at-the-future-of-go-summit-in-wuzhen/155279016?prev_url=detail</a> Darpa: Anthony Kendall This is VW's entry in the Darpa Grand Challenge. <a href="https://www.flickr.com/photos/80546870@N00/87625236">https://www.flickr.com/photos/80546870@N00/87625236</a> CC BY-SA 2.0

Abbildungs-Nr.	Bildnachweis
Abb. 8.11	Pong, Breakout: Eigene Snapshots vom public domain stella Simulator
Abb. 8.12	(Osband et al. 2016)
Abb. 8.13	Eigene Snapshots vom public domain stella Simulator
Abb. 8.16	(Mnih et al. 2015) Ausschnitt
Abb. 8.17	Quelle: eigener screenshot vom public domain stella emulator
Abb. 8.18	james bamber 4th UK Rock Paper Scissors Championships <a href="https://commons.wikimedia.org/wiki/File:Rps2010_2.jpg">https://commons.wikimedia.org/wiki/File:Rps2010_2.jpg</a> gemeinfrei
Abb. 8.20	(Levine 2018)
Abb. 8.21	Richard S. Sutton 2016 Steve Jurvetson from Menlo Park, USA – Deep Thinkers on Deep Learning <a href="https://de.wikipedia.org/wiki/Richard_S._Sutton#/media/Datei:Richard_Sutton,_October_27,_2016.jpg">https://de.wikipedia.org/wiki/Richard_S._Sutton#/media/Datei:Richard_Sutton,_October_27,_2016.jpg</a> CC BY 2.0
Abb. 8.22	Quelle: <a href="https://de.wikipedia.org/wiki/Datei:Path_to_Angels_Landing.jpg">https://de.wikipedia.org/wiki/Datei:Path_to_Angels_Landing.jpg</a> Roman Fuchs, Path to Angels Landing – Zion National Park. CC BY 3.0
Abb. 8.23	Quelle: (Schulman et al. 2015)
Abb. 8.24	(Schulman et al. 2015)
Abb. 8.26	Quelle: <a href="https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister-1904-140710.html">https://www.golem.de/news/dota-2-open-ai-besiegt-die-weltmeister-1904-140710.html</a>
Abb. 8.27	<a href="https://en.wikipedia.org/wiki/Sebastian_Thrun#/media/File:Sebastian_Thrun_World_Economic_Forum_2013.jpg">https://en.wikipedia.org/wiki/Sebastian_Thrun#/media/File:Sebastian_Thrun_World_Economic_Forum_2013.jpg</a> 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
Abb. 8.28	(Fridman 2019)
Abb. 8.29	(Fridman 2019)
Abb. 8.30	(Fridman 2019)
Abb. 8.31	(Fridman 2019)
Abb. 8.32	<i>Eigene Grafik</i> (Giacaglia 2019)
Abb. 8.33	Eigene Grafik: Quellen: <a href="https://www.machinedesign.com/motion-control/saved-sensor-vehicle-awareness-self-driving-age">https://www.machinedesign.com/motion-control/saved-sensor-vehicle-awareness-self-driving-age</a> , <a href="https://commons.wikimedia.org/wiki/File:C3top_svg.svg">https://commons.wikimedia.org/wiki/File:C3top_svg.svg</a>
Abb. 8.34	<a href="https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto-%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungen-gm938038454-256530428">https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto-%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungen-gm938038454-256530428</a> <a href="https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto-%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungen-gm938038442-256530421">https://www.istockphoto.com/de/foto/verschneite-gef%C3%A4hrlich-auto-%C3%BCberholen-auf-der-autobahn-mit-whiteout-bedingungen-gm938038442-256530421</a>
Abb. 8.35	(Anguelov 2019)
Abb. 8.36	(Anguelov 2019)
Abb. 8.37	<i>Quelle:</i> (Fridman 2019)
Abb. 8.38	(Pinto et al. 2017)
Abb. 8.39	Quelle: (Chen et al. 2019)

Abbildungs-Nr.	Bildnachweis
Abb. 9.1	Eigene Grafik <a href="https://commons.wikimedia.org/wiki/Category:Paintings_by_Vincent_van_Gogh_by_title?uselang=de">https://commons.wikimedia.org/wiki/Category:Paintings_by_Vincent_van_Gogh_by_title?uselang=de</a> gemeinfrei Bild1: „Vincent van Gogh (1853–1890), Irises, May 1890“ by Tulip Hysteria/ Go to albums is licensed under CC PDM 1.0 <a href="https://ccsearch.creativecommons.org/photos/c71e11ef-3a38-4201-8b62-347c7a5fef05">https://ccsearch.creativecommons.org/photos/c71e11ef-3a38-4201-8b62-347c7a5fef05</a> , Bild2: „Self Portrait 1887“ by Thomas Henneman is licensed under CC PDM 1.0 <a href="https://ccsearch.creativecommons.org/photos/2b32ce17-15b6-430d-8e88-f9edfff8aaa7">https://ccsearch.creativecommons.org/photos/2b32ce17-15b6-430d-8e88-f9edfff8aaa7</a> , Bild3: „Vincent van Gogh“ by KUUNSTKUULTUR is licensed under CC PDM 1.0 <a href="https://ccsearch.creativecommons.org/photos/440302b1-5fca-4e44-b3b5-c0593cfcc217">https://ccsearch.creativecommons.org/photos/440302b1-5fca-4e44-b3b5-c0593cfcc217</a> , Bild4: „Roses (1890) by Vincent Van Gogh. Original from the MET Museum. Digitally enhanced by rawpixel.“ by Free Public Domain Illustrations by rawpixel is licensed under CC BY 2.0 <a href="https://ccsearch.creativecommons.org/photos/30ee54fa-e85e-4259-a067-660fde8a93c1">https://ccsearch.creativecommons.org/photos/30ee54fa-e85e-4259-a067-660fde8a93c1</a> , Bild5: „Absinthe_Van Gogh Museum“ by E V E is licensed under CC BY-SA 2.0 <a href="https://ccsearch.creativecommons.org/photos/07ede5fa-7d80-408b-baa4-3b8c7ea6c423">https://ccsearch.creativecommons.org/photos/07ede5fa-7d80-408b-baa4-3b8c7ea6c423</a> Bild6: Portrait of Félix Rey, January 1889, Pushkin Museum; public domain <a href="https://en.wikipedia.org/wiki/Vincent_van_Gogh#/media/File:Felix_Rey_portrait_&amp;_sketch.jpg">https://en.wikipedia.org/wiki/Vincent_van_Gogh#/media/File:Felix_Rey_portrait_&amp;_sketch.jpg</a>
Abb. 9.3	Quelle: (Karras et al. 2018)
Abb. 9.4	(Karras et al. 2018)
Abb. 9.5	(Karras et al. 2018).
Abb. 9.6	(Brock et al. 2019)
Abb. 9.7	(Karras et al. 2018)
Abb. 9.8	Quelle: (Radford et al. 2016)
Abb. 9.9	Quelle: (Isola et al. 2017)
Abb. 9.10	(Isola et al. 2017)
Abb. 9.11	(Isola et al. 2017)
Abb. 9.12	(Zhu et al. 2017)
Abb. 9.13	(Wolf 2018)
Abb. 9.14	Quelle: (Zhu et al. 2017)
Abb. 9.15	(Elgammal et al. 2017)
Abb. 9.16	(Elgammal et al. 2017)
Abb. 9.17	(Zhang et al. 2017).
Abb. 9.18	(Zhang et al. 2017)
Abb. 9.19	(Vardaan 2019). Einzelbilder aus einem Video
Abb. 9.22	Eigene Grafik (Fan et al. 2018)
Abb. 9.27	Quelle: (Vig 2019)
Abb. 9.28	(Dathathri et al. 2020)
Abb. 9.29	Text Quelle: (Payne 2019)
Abb. 9.30	Eigene Grafik, Quelle: (Payne 2019)
Abb. 9.31	Quelle: (Payne 2019)
Abb. 9.33	Quelle: (Huang et al. 2019)
Abb. 9.35	Steve Troughton Ex_Machina 02n Public Domain Mark 1.0 <a href="https://www.flickr.com/photos/stevetroughton/16477163443/">https://www.flickr.com/photos/stevetroughton/16477163443/</a>
Abb. 9.37	(Shum et al. 2018)

Abbildungs-Nr.	Bildnachweis
Abb. 9.38	Eigene Grafik nach (Shum et al. 2018)
Abb. 9.41	<a href="https://www.flickr.com/photos/websummit/22762564236">https://www.flickr.com/photos/websummit/22762564236</a> 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.
Abb. 9.42	(Guo et al. 2018)
Abb. 9.43	Quelle: rechts: <a href="https://thispersondoesnotexist.com/">https://thispersondoesnotexist.com/</a> , links: (Elgammal, et al. 2017)
Abb. 9.44	Quelle: <a href="https://towardsdatascience.com/openai-gpt-2-writes-alternate-endings-for-game-of-thrones-c9be75cd2425">https://towardsdatascience.com/openai-gpt-2-writes-alternate-endings-for-game-of-thrones-c9be75cd2425</a>
Abb. 9.45	<a href="https://en.wikipedia.org/wiki/Ben_Carey">https://en.wikipedia.org/wiki/Ben_Carey</a> Justin M. Martinez, U.S. Marine Corps, public domain
Abb. 10.1	Eigene Grafik, verwendet: The famous red eye of HAL 9000 by Cryteria – <a href="https://de.wikipedia.org/wiki/HAL_9000#/media/Datei:HAL9000.svg">https://de.wikipedia.org/wiki/HAL_9000#/media/Datei:HAL9000.svg</a> CC BY 3.0
Abb. 10.2	Bild 2008: Raised hands in honor of a musical show on stage Von 9parusnikov <a href="https://stock.adobe.com/de/266551332">https://stock.adobe.com/de/266551332</a> , Bild 2012: People taking photographs with touch smart phone during a music entertainment public concert Von verve <a href="https://stock.adobe.com/de/290346853">https://stock.adobe.com/de/290346853</a> Bild 2016: Group of fans taking selfie with smart phone while watching match in stadium. <a href="https://www.istockphoto.com/de/foto/zuschauergruppe-jubelt-im-stadion-gm1131336410-299526968">https://www.istockphoto.com/de/foto/zuschauergruppe-jubelt-im-stadion-gm1131336410-299526968</a>
Abb. 10.4	Copyrightfrei 50 Jahre nach Entstehung 1922. <a href="https://www.imdb.com/title/tt0013099/mediaviewer/rm320567040?ref_=ttmi_mi_all_sf_12">https://www.imdb.com/title/tt0013099/mediaviewer/rm320567040?ref_=ttmi_mi_all_sf_12</a>
Abb. 10.5	Lung CA seen on CXR James Heilman, MD CC BY-SA 3.0 <a href="https://en.wikipedia.org/wiki/Lung_cancer#/media/File:LungCACXR.PNG">https://en.wikipedia.org/wiki/Lung_cancer#/media/File:LungCACXR.PNG</a>
Abb. 10.6	Magnetresonanzaufnahme eines Knies Autor: Test21 CC BY-SA 3.0 <a href="https://de.wikipedia.org/wiki/Magnetresonanztomographie#/media/Datei:Knie_mr.jpg">https://de.wikipedia.org/wiki/Magnetresonanztomographie#/media/Datei:Knie_mr.jpg</a>
Abb. 10.7	Doctor making a vaccination Von guerrieroale <a href="https://stock.adobe.com/de/images/doctor-making-a-vaccination/150155835">https://stock.adobe.com/de/images/doctor-making-a-vaccination/150155835</a>
Abb. 10.8	<a href="https://stock.adobe.com/de/images/young-disable-man-in-the-robotic-exoskeleton-sitting-in-wheelchair-in-the-rehabilitation-clinic/227997068">https://stock.adobe.com/de/images/young-disable-man-in-the-robotic-exoskeleton-sitting-in-wheelchair-in-the-rehabilitation-clinic/227997068</a>
Abb. 10.10	Autor: 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 <a href="https://en.wikipedia.org/wiki/Smart_manufacturing#/media/File:BMW_Leipzig_MEDIA_050719_Download_Karosseriebau_max.jpg">https://en.wikipedia.org/wiki/Smart_manufacturing#/media/File:BMW_Leipzig_MEDIA_050719_Download_Karosseriebau_max.jpg</a>
Abb. 10.12	(Dyrmann et al. 2017)
Abb. 10.13	public domain <a href="https://commons.wikimedia.org/wiki/File:FEMA_-_34156_-_Preliminary_Damage_Assessment_Team_workin_in_Tennessee.jpg">https://commons.wikimedia.org/wiki/File:FEMA_-_34156_-_Preliminary_Damage_Assessment_Team_workin_in_Tennessee.jpg</a>
Abb. 10.14	Grendelkhan A Waymo self-driving car on the road in Mountain View, making a left turn. (Side view.) <a href="https://it.wikipedia.org/wiki/File:Waymo_self-driving_car_side_view.gk.jpg">https://it.wikipedia.org/wiki/File:Waymo_self-driving_car_side_view.gk.jpg</a> CC BY-SA 4.0
Abb. 10.15	(Osborn 2018)
Abb. 10.16	David Berkowitz Nest thermostat – Consumer Electronics Show – CES 2013 – Las Vegas CC BY 2.0 <a href="https://www.flickr.com/photos/davidberkowitz/8381790944/">https://www.flickr.com/photos/davidberkowitz/8381790944/</a>
Abb. 10.18	(Wang et al. 2018)

Abbildungs-Nr.	Bildnachweis
Abb. 10.19	(Kanellos 2016).
Abb. 10.20	Sean MacEntee: Amazon UK. CC BY 2.0 <a href="https://www.flickr.com/photos/smemon/6500597907/">https://www.flickr.com/photos/smemon/6500597907/</a>
Abb. 10.21	Eigene Grafik, (Hosseini und Schmidt 2018)
Abb. 10.24	(Shoham et al. 2018)
Abb. 10.25	Eigene Grafik, Quelle (Kropp et al. 2018)
Abb. 10.26	<a href="https://en.wikipedia.org/wiki/DJ_Patil#/media/File:DJPatil.jpg">https://en.wikipedia.org/wiki/DJ_Patil#/media/File:DJPatil.jpg</a> public domain
Abb. 10.28	Datei-Nr.: 283969663 Competition between humans and robots in tug of war concept. Von Elnur <a href="https://stock.adobe.com/de/images/competition-between-humans-and-robots-in-tug-of-war-concept/283969663">https://stock.adobe.com/de/images/competition-between-humans-and-robots-in-tug-of-war-concept/283969663</a>
Abb. 10.29	Stock adobe.com: 215544030. Data Security For Doctor Providing Telemedicine <a href="https://t3.ftcdn.net/jpg/02/15/54/40/240_F_215544030_XNKDaFvP5m92mFohOuRqmWNIdz6zbBxy.jpg">https://t3.ftcdn.net/jpg/02/15/54/40/240_F_215544030_XNKDaFvP5m92mFohOuRqmWNIdz6zbBxy.jpg</a>
Abb. 10.30	Roncheng's „civilized families“ are displayed on public noticeboards like these. (Simina Mistreanu) <a href="https://foreignpolicy.com/2018/04/03/life-inside-chinas-social-credit-laboratory/">https://foreignpolicy.com/2018/04/03/life-inside-chinas-social-credit-laboratory/</a>
Abb. 10.31	Dirk Ingo Franke Six surveillance cameras overlooking a gas station next to the Autobahn A9 3.Juni 2013. CC BY 3.0 <a href="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">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</a>
Abb. 10.32	<a href="https://en.wikipedia.org/wiki/Perdix_(drone)">https://en.wikipedia.org/wiki/Perdix_(drone)</a> #/media/File:Perdix-drone.jpg public domain
Abb. 10.33	Adobe Stock Datei 102117631. <a href="https://stock.adobe.com/de/search/images?load_type=search&amp;native_visual_search=&amp;similar_content_id=&amp;is_recent_search=&amp;search_type=usertyped&amp;k=artificial+general+intelligence&amp;asset_id=102117631">https://stock.adobe.com/de/search/images?load_type=search&amp;native_visual_search=&amp;similar_content_id=&amp;is_recent_search=&amp;search_type=usertyped&amp;k=artificial+general+intelligence&amp;asset_id=102117631</a>
Abb. 10.34	<a href="https://stock.adobe.com/de/search?load_type=search&amp;native_visual_search=&amp;similar_content_id=&amp;is_recent_search=&amp;search_type=usertyped&amp;k=artificial+intelligence&amp;asset_id=258774440">https://stock.adobe.com/de/search?load_type=search&amp;native_visual_search=&amp;similar_content_id=&amp;is_recent_search=&amp;search_type=usertyped&amp;k=artificial+intelligence&amp;asset_id=258774440</a>
Abb. 10.35	<i>Eigene Darstellung. Quelle:</i> <a href="http://biomine.cs.helsinki.fi/">http://biomine.cs.helsinki.fi/</a>
Abb. 10.36	Eigene Darstellung. Quelle: <a href="https://commons.wikimedia.org/wiki/File:Human-brain.SVG">https://commons.wikimedia.org/wiki/File:Human-brain.SVG</a>
Abb. 10.37	John F. Kennedy (1963), Foto: Cecil W. Stoughton gemeinfrei. <a href="https://de.wikipedia.org/wiki/John_F._Kennedy#/media/Datei:John_F._Kennedy,_White_House_color_photo_portrait.jpg">https://de.wikipedia.org/wiki/John_F._Kennedy#/media/Datei:John_F._Kennedy,_White_House_color_photo_portrait.jpg</a>
Abb. 10.38	Pfeile selbst eingezeichnet. <a href="https://stock.adobe.com/de/images/male-golf-player-on-professional-golf-course-golfer-with-golf-club-taking-a-shot/242337141">https://stock.adobe.com/de/images/male-golf-player-on-professional-golf-course-golfer-with-golf-club-taking-a-shot/242337141</a>
Abb. 10.39	Cathy O'Neil bei der Vorstellung ihres Buches bei Google Cambridge 2016 <a href="https://de.wikipedia.org/wiki/Cathy_O%27E2%80%99Neil#/media/Datei:Cathy_O'Neil_at_Google_Cambridge.jpg">https://de.wikipedia.org/wiki/Cathy_O%27E2%80%99Neil#/media/Datei:Cathy_O'Neil_at_Google_Cambridge.jpg</a> Von GRuban – Eigenes Werk, CC BY-SA 4.0
Abb. 10.41	<i>Erzeugt aus dem Text von (Cremers et al. 2019) mit Hilfe von</i> <a href="https://www.wordclouds.com/">https://www.wordclouds.com/</a>
Abb. 10.42	<i>Eigene Darstellung Quelle</i> (Cremers et al. 2019)

Abbildungs-Nr.	Bildnachweis
Abb. 10.43	<i>Humanoider Roboter als Justitia mit einem Schwert und einen Waage</i> Von Alexander Limbach. 229515942 <a href="https://t4.ftcdn.net/jpg/02/29/51/59/240_F_229515942_s71SdDf0w2s9rFUVX78nU0R36XHnYz7G.jpg">https://t4.ftcdn.net/jpg/02/29/51/59/240_F_229515942_s71SdDf0w2s9rFUVX78nU0R36XHnYz7G.jpg</a>
Abb. 10.44	Stock-Illustration-ID:1145748016 <a href="https://www.istockphoto.com/de/vektor/maschinentechnik-flachbanner-gm1145748016-308484501">https://www.istockphoto.com/de/vektor/maschinentechnik-flachbanner-gm1145748016-308484501</a>
Abb. 10.45	Richard Patterson: Hacker that looks to have been caught in the act. CC BY 2.0. <a href="https://www.flickr.com/photos/136770128@N07/40492737110/">https://www.flickr.com/photos/136770128@N07/40492737110/</a>
Abb. 10.46	<a href="https://www.istockphoto.com/de/foto/computer-netzwerk-sicherheit-cyber-sicherheit-gro%C3%9Fe-daten-schutz-verschl%C3%BCsselung-gm1130248375-298854279">https://www.istockphoto.com/de/foto/computer-netzwerk-sicherheit-cyber-sicherheit-gro%C3%9Fe-daten-schutz-verschl%C3%BCsselung-gm1130248375-298854279</a>

---

## A.3 Literatur zum Anhang

- Anguelov, Drago. 2019.** Taming the Long Tail of Autonomous Driving Challenges. [Online] 2019. [Zitat vom: 05. 06 2019.] <https://www.youtube.com/watch?v=Q0nGo2-y0xY>.
- Bahdanau, D., Cho, K. und Bengio, Y. 2015.** Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. *ICLR*. 2015.
- Britz, D., et al. 2017.** Massive Exploration of Neural Machine Translation Architectures. *EMNLP*. 2017.
- Brock, A., Donahue, J. und Simonyan, K. 2019.** Large scale gan training for high fidelity natural image synthesis. *ICLR*. 2019.
- Chan, W., et al. 2016.** Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 49. 2016).
- Chen, T., et al. 2019.** Adversarial attack and defense in reinforcement learning-from AI security view. *Cybersecurity*, 2(1), 11. 2019.
- Chung, J. S., et al. 2017.** Lip reading sentences in the wild. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3444–3453). IEEE. 2017.
- Cremers, A., et al. 2019.** Vertrauenswürdiger Einsatz von Künstlicher Intelligenz. *Fraunhofer Institut für Intelligente Analyse- und Informationssysteme, Sankt Augustin*. 2019.
- Dathathri, S., et al. 2020.** Plug and play language models: a simple approach to controlled text generation. [Online] 2020. arXiv preprint arXiv:1912.02164.
- Devlin, J., et al. 2018.** Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint*. [Online] 2018. arXiv:1810.04805.
- Dyrmann, M., Jørgensen, R. N. und 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. 2017.
- Elgammal, A., et al. 2017.** Can: Creative adversarial networks, generating “art” by learning about styles and deviating from style norms. [Online] 2017. [Zitat vom: 10. 01 2020.] arXiv preprint arXiv:1706.07068.
- Eslami, S. A., et al. 2018.** Neural scene representation and rendering. *Science*, 360(6394), 1204–1210. 2018.

- 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. 2017.
- Eykholz, K., et al. 2018.** Robust physical-world attacks on deep learning visual classification. *Proc. ICPR pp. 1625–1634*. 2018.
- Fan, A., Lewis, M. und Dauphin, Y. 2018.** Hierarchical neural story generation. [Online] 2018. arXiv preprint arXiv:1805.04833.
- Fridman, Lex. 2019.** Self driving cars: state of the art 2019. [Online] 2019. [Zitat vom: 03. 06 2019.] <https://www.youtube.com/watch?v=sRxaMDDMWQQ>.
- Giacaglia, G. 2019.** Self-Driving Cars. [Online] 2019. [Zitat vom: 05. 06 2019.] <https://medium.com/@giacaglia/self-driving-cars-f921d75f46c7>.
- Grover, P. 2018.** Evolution of Object Detection and Localization Algorithms. [Online] 2018. <https://towardsdatascience.com/evolution-of-object-detection-and-localization-algorithms-e241021d8bad>.
- Guo, X., et al. 2018.** Group-Level Emotion Recognition using Hybrid Deep Models based on Faces, Scenes, Skeletons and Visual Attentions. *Proceedings of the 2018 on International Conference on Multimodal Interaction (pp. 635–639)*. ACM. 2018.
- He, K. 2016.** Deep Residual Networks – Deep Learning Gets Way Deeper ICML 2016 Tutorial. *ICML 2016*. 2016.
- Hendrycks, D. und Dietterich, T. 2019.** Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *ICLR 2019*. 2019.
- Hosseini, H. und Schmidt, H. 2018.** Wert der Plattform-Ökonomie steigt im ersten Halbjahr um 1 Billion \$. [Online] 25. 06 2018. [Zitat vom: 14. 01 2020.] <https://www.platformeconomy.com/blog/wert-der-plattform-okonomie-steigt-im-ersten-halbjahr-um-1-billion>.
- Huang, C.-Z., Simon, I. und Dinculescu, M. 2019.** Music Transformer: Generating Music with Long-Term Structure. [Online] 2019. [Zitat vom: 19. 05 2019.] <https://magenta.tensorflow.org/music-transformer>.
- Huang, G., et al. 2017.** Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700–4708)*. 2017.
- Isola, P., et al. 2017.** Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1125–1134)*. 2017.
- Jia, Y., et al. 2019.** Direct speech-to-speech translation with a sequence-to-sequence model. [Online] 12. 04 2019. [Zitat vom: 06. 06 2019.] <https://arxiv.org/abs/1904.06037>.
- Kanellos, M. 2016.** 152,000 Smart Devices Every Minute In 2025: IDC Outlines The Future of Smart Things. *Forbes*. 2016.
- Karpukhin, V., et al. 2019.** Training on Synthetic Noise Improves Robustness to Natural Noise in Machine Translation. [Online] 2019. [Zitat vom: 15. 06 2019.] arXiv preprint arXiv:1902.01509.
- Karras. 2019.** This Person does not exist. [Online] 2019. [Zitat vom: 19. 05 2019.] <https://thispersondoesnotexist.com/>.

- Karras, T., et al. 2018.** Progressive growing of gans for improved quality, stability, and variation. *ICLR 2018*. 2018.
- Krizhevsky, A., Sutskever, I. und Hinton, G. E. 2012.** Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105). 2012.
- Kropp, P., Theuer, S. und Fritzsche, B. 2018.** Immer mehr Tätigkeiten werden durch Digitalisierung ersetzbar. *IAB 2018, S SN 1861-14 35, S.32*. 2018.
- Kudugunta, S. R., et al. 2019.** Investigating Multilingual NMT Representations at Scale. [Online] 2019. [Zitat vom: 21. 10 2019.] arXiv preprint arXiv:1909.02197.
- Lapuschkin, S., et al. 2019.** Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1), 1–8. 2019.
- Lee, H., et al. 2011.** Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks. *Communications of the ACM*. 2011.
- Levine, S. 2018.** Policy Gradients. *CS 294–112: Deep Reinforcement Learning*. [Online] 2018. [Zitat vom: 06. 05 2019.] <http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-5.pdf>.
- Li, Fei-Fei, Johnson, Justin und Yeung, Serena. 2018.** *CNNs for Visual Recognition. Lecture 2: Image Classification pipeline*. s.l. : Stanord University, 2018.
- Li, J., et al. 2019.** Jasper: An End-to-End Convolutional Neural Acoustic Model. [Online] 2019. [Zitat vom: 23. 04 2019.] arXiv preprint arXiv:1904.03288.
- Liang, J., et al. 2018.** Gpu-accelerated robotic simulation for distributed reinforcement learning. [Online] 2018. [Zitat vom: 27. 05 2019.] arXiv preprint arXiv:1810.05762.
- Lindsay, RK, Buchanan, BG und Feigenbaum, EA. 1993.** DENDRAL: a case study of the first expert system for scientific hypothesis formation. *Artificial intelligence*, 1993. 1993.
- Mnih, V., et al. 2015.** Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529. 2015.
- Monfort, M., et al. 2019.** Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*. 2019.
- MultiNLI. 2017.** [Online] 2017. [Zitat vom: 04.04.2019] <https://www.nyu.edu/projects/bowman/multinli/>.
- Nayak, Pandu. 2019.** Understanding searches better than ever before. [Online] 2019. [Zitat vom: 25.10.2019] <https://blog.google/products/search/search-language-understanding-bert>.
- Noy, A., et al. 2020.** ASAP: Architecture search, anneal and prune. In International Conference on Artificial Intelligence and Statistics (pp. 493–503). PMLR.
- Osband, I., et al. 2016.** Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems* (pp. 4026–4034). 2016.
- Osborn, L. E., et al. 2018.** Prosthesis with neuromorphic multilayered e-dermis perceives touch and pain. *Science Robotics*, 3(19), eaat3818. 2018.
- Payne, C. 2019.** MuseNet. [Online] 2019. [Zitat vom: 18.05.2019.] <https://openai.com/blog/musenet/>.

- Phillips, P. J., et al. 2018.** Face recognition accuracy of forensic examiners, superrecognitioners, and face recognition algorithms. *Proceedings of the National Academy of Sciences*, 115(24), 6171–6176. 2018.
- Pinto, L., et al. 2017.** Robust Adversarial Reinforcement Learning. *Proceedings of the 34th International Conference of Machine Learning, PMLR 70*. 2017.
- Popel, M. und Bojar, O. 2018.** *Training Tips for the Transformer Model*. Prag : Karls-Universität, 2018.
- Radford, A., Metz, L. und Chintala, S. 2016.** Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR 2016*. 2016.
- Redmon, J. und Farhadi, A. 2017.** YOLO9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263–7271)*. 2017.
- Ribeiro, M. T., Singh, S. und Guestrin, C. 2016.** Model-agnostic interpretability of machine learning. [Online] 2016. [Zitat vom: 24.08.2019] arXiv preprint arXiv:1606.05386.
- Ronneberger, O., Fischer, P. und Brox, T. 2015.** U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention (pp. 234–241)*. Springer, Cham. 2015.
- Schönherr, L., et al. 2019.** Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding. *Network and Distributed System Security Symposium, NDSS 2019*, . 2019.
- Schulman, J., et al. 2015.** High-dimensional continuous control using generalized advantage estimation. [Online] 2015. [Zitat vom: 08. 05 2019.] arXiv preprint arXiv:1506.02438.
- Sennrich, R., Haddow, B. und 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) (Vol. 1, pp. 1715–1725)*. 2016.
- Sharma, P., et al. 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 (pp. 2556–2565)*. 2018.
- Shazeer, N., et al. 2017.** Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. [Online] 2017. [Zitat vom: 21.10.2019] arXiv preprint arXiv:1701.06538.
- Shoham, Y., et al. 2018.** The AI Index 2018 Annual Report. *AI Index Steering Committee, Human-Centered AI Initiative, Stanford University, Stanford, CA*, . 2018.
- Shum, H. Y., He, X. D. und Li, D. 2018.** From Eliza to XiaoIce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 10–26. 2018.
- Simonyan, K. und Zisserman, A. 2014.** Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems (pp. 568–576)*. 2014.
- Strobelt, H., et al. 2016.** Visual analysis of hidden state dynamics in recurrent neural networks. [Online] 2016. arXiv preprint arXiv:1606.07461.

- Sun, C., et al. 2019.** Videobert: A joint model for video and language representation learning. [Online] 2019. arXiv preprint arXiv:1904.01766.
- Sutskever, I., Vinyals, O. und Le, Q. V. 2014.** Sequence to sequence learning with neural networks. *Advances in neural information processing systems* (pp. 3104–3112). 2014.
- Szegedy, C., et al. 2014.** Intriguing properties of neural networks. [Online] 2014. [Zitat vom: 30. 05 2019.] arXiv preprint arXiv:1312.6199.
- Teye, M., Azizpour, H. und Smith, K. 2018.** Bayesian uncertainty estimation for batch normalized deep networks. [Online] 2018. [Zitat vom: 09.06.2019] arXiv preprint arXiv:1802.06455.
- Touvron, H., et al. 2019.** Fixing the train-test resolution discrepancy. [Online] 19. 07 2019. [Zitat vom: 06. 10 2019.] <https://arxiv.org/pdf/1906.06423v2.pdf>.
- Van Horn, G., et al. 2018.** The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8769–8778). 2018.
- Vardaan. 2019.** AI to Steal Fashion Model Jobs? New AI Able to Generate Entire Bodies of People Who Don't Exist. [Online] 2019. [Zitat vom: 02. 09 2019.] <https://www.indianweb2.com/2019/05/05/ai-to-steal-fashion-model-jobs-new-ai-able-to-generate-entire-bodies-of-people-who-dont-exist/>.
- Vaswani, Ashish, et al. 2017.** Attention is All you Need. *NIPS 2017: 6000–6010*. 2017.
- Vig, J. 2019.** OpenAI GPT-2: Understanding Language Generation through Visualization. [Online] 2019. [Zitat vom: 17.05.2019] <https://towardsdatascience.com/openai-gpt-2-understanding-language-generation-through-visualization-8252f683b2f8>.
- Wang, C., Li, M. und Smola, A. J. 2019.** Language Models with Transformers. [Online] 2019. [Zitat vom: 27. 05 2019.] arXiv preprint arXiv:1904.09408.
- Wang, H., et al. 2018.** A Captcha Design Based on Visual Reasoning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1967–1971). IEEE. 2018.
- Wolf, S. 2018.** CycleGAN: Learning to Translate Images (Without Paired Training Data). [Online] 2018. <https://towardsdatascience.com/cyclegan-learning-to-translate-images-without-paired-training-data-5b4e93862c8d>.
- Wu, L. Y., et al. 2018.** Starspace: Embed all the things! *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- Zeiler, M. D. und Fergus, R. 2014.** Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer, Cham. 2014.
- Zhang, H., et al. 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.). 2017.
- Zhou, L., et al. 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). 2018.

- Zhu, J. Y., et al. 2017.** Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *In Proceedings of the IEEE International Conference on Computer Vision (pp. 2223–2232)*. 2017.
- Zhu, Y., et al. 2019.** Improving Semantic Segmentation via Video Propagation and Label Relaxation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 8856–8865)*. 2019.

---

## A.4 Stichwortverzeichnis

### A

abgeschnittene lineare Funktion 83  
Abkühlen 113  
Ableitung  
    einer Funktion 68  
    partielle 69  
ActivityNet Videodaten 282  
Actor-Critic 313  
adversariale Attacke 160, 242  
adversariales Beispiel 160, 431  
AGI = Artificial General Intelligence 39. . *Siehe auch auch* allgemeine Künstliche Intelligenz  
AKI 39. . *Siehe auch* allgemeine Künstliche Intelligenz  
Aktion beim Bestärkungslernen 295  
Aktivierungsfunktion 83, 445  
Alexa Sprachassistent 25, 249, 272, 375  
AlexNet Bildklassifikation 16, 132  
Algorithmus 50, 445  
Alibaba chinesische Internetfirma 402  
allgemeine Künstliche Intelligenz 39, 418  
AlphaGo Go-Agent 28, 314  
ALPHA Luftkampfbot 416  
AlphaZero Spieleagent 28  
Amazon 401  
Analogie 178  
Annealing 113  
Anreicherung der Trainingsdaten 137, 264  
a-posteriori-Wahrscheinlichkeit 155  
Apple 401  
Arbeitsmarkt, Auswirkungen der KI 404  
Architektursuche 111  
Aristo System 24, 230

Artificial General Intelligence 39. . *Siehe auch allgemeine Künstliche Intelligenz*  
ASAP-Methode zur Hyperparameter-Optimierung 113  
Assistent, intelligenter persönlicher 25, 270  
Atari  
    2600 30, 305  
    Breakout 305  
    Montezuma's Revenge 308  
    Pong 305  
    Roadrunner 305  
    Seaquest 305  
Attention 204  
Attentiongewichte 203  
Attention, Multi-Head- 211  
Aufmerksamkeit 423  
Auftragsassistent 270  
Ausgabegatter 190  
Autoencoder 106, 445  
Autoencoder-Netze 106  
AutoKeras 112  
automatische Bildbeschreibung 234  
autonomes Waffensystem 417  
Auto, selbstfahrendes 317

### B

Backpropagation 85, 90, 446  
Bagging 155, 449. . *Siehe auch* Bootstrap  
Baidu chinesische Internetfirma 402  
BAT, chinesische Internetfirmen 402  
Batch-Normalisierung 102  
Bayessche neuronale Netze 155  
Beamforming 286

- Beam-Suche 200, 446  
 Beantworten von Fragen 22. . Siehe auch  
     Fragen beantworten  
 Belohnung beim Bestärkungslernen 48,  
     295, 446  
 Benannte Entität 195, 446  
 Bengio, Yoshua 174  
 Bestärkungslernen 28, 47, 293, 447  
 Bestärkungslernen-Netz 105, 301, 311  
 Bestrafung großer Parameterwerte 100  
 Bewusstsein 39, 423  
 Bias 59  
 Bidirektionales LSTM 195, 447  
 Bilderkennung 4, 15, 119  
 Bildsegmentierung 145  
 Bildung und KI 406  
 Bildverstehen 119, 447  
 biologische Landwirtschaft und KI 390  
 Bionik und KI 393  
 black box-Attacke 162  
 Black-Box-Modell 237, 448  
 Blender Chatbot 367  
 BLEU-Wert der Übersetzungsqualität 201, 448  
 BookCorpus, Bücher und deren Verfilmung 224  
 Boosting 449  
 Bootstrap 155, 449  
 BOS Symbol für Satzanfang 199  
 Bot 30  
 Bounding-Box 119, 144  
 Breakout Atari-Spiel 305  
 Business Expert 409  
 Bypass 212  
 Byte Pair Encoding 216
- C**  
 Cambridge Analytica 415  
 Captcha Bildannotation 395  
 Capture the Flag Videospiel 30  
 Carcraft-Simulationsumgebung 322  
 Chatbot 270, 448  
     Alexa 272  
     Eliza 361  
     Gunrock 272  
     Tay 271  
     XiaoIce 359  
 CIFAR-10 Bilddaten 113, 137  
 Cityscapes-Datensatz für Bildsegmentierung 148  
 Cloud 93
- CNN 105, 123, 448. . Siehe auch Convolutional Neural Network  
 CNTK Toolkit 94, 460  
 Compute-Cluster 93  
 computergeneriertes Gemälde 36, 331, 342  
 ConceptNet semantisches Netz 273  
 Conceptual Captions-Bilddaten 235  
 CoNLL-2003 Datensatz zur Namenserkennung  
     195, 226  
 Convolutional Neural Network 105, 123, 448  
 Convolutional Neural Network, gedecktes 266  
 Convolution-Schicht 123  
 Convolution-Schicht, eindimensionale 260  
 CRISP-DM 109. . Siehe auch Cross-Industry Standard Process for Data Mining  
 Cross-Industry Standard Process for Data Mining 109  
 Crowdworker 15  
 Cyberangriffe 418  
 Cyborg Texterzeugung 347  
 CycleGAN 339  
 Cyc Wissensdatenbank 228, 421
- D**  
 DARPA Wettbewerb für selbstfahrende Autos 33  
 Data Analyst 409  
 Data Augmentation 137. . Siehe auch Anreicherung der Trainingsdaten  
 Data Manager 409  
 Data Scientist 407  
 Daten 448  
     immaterieller Vermögenswert 395  
     Wertschöpfung aus 394  
 Datenmonopol 396  
 Datenschutzgrundverordnung 415  
 Datenstrategie 395  
 3D-Convolution-Schicht 279  
 Debater Dialogsystem 25  
 Decoder 212  
     Decoder-RNN 199, 458  
     Deep Blue 27  
     Denken, langsames 423  
     Denken, schnelles 422  
     Diagnose von Krankheitsbildern 380  
     Dialogakt 273  
     Dialogsystem Debater 25  
     differential Privacy 438

- Digitaler Zwilling 388  
Digitalisierung 398, 448  
Dimensionen der Intelligenz 1  
Diskriminatornetz 332  
Dota2 Videospiel 32, 316  
Drohnen 417  
Drohnenschwarm 417  
Dropout 101, 449  
3D-Struktur von Proteinen 18  
duales Lernen 218  
Dual Use KI-System 439  
Duplex Sprachgenerierung 269, 426
- E**  
ebenenweise Relevanz-Propagierung 240  
Economy of Scale 399  
Economy of Speed 398  
Eigennamenerkennung 195, 226  
eindimensionale Convolution-Schicht 260  
ELIZA Chatbot 270, 361  
Embedding 172, 449. . *Siehe auch*  
    Embeddingvektor  
Embedding-Netz 173, 449  
Embeddingvektor 172, 449  
Encoder 212  
Encoder-RNN 199, 458  
Ensemblemethode 155, 449  
Ensemble von Modellen 155, 202, 449  
Entscheidungsbaum 238  
EOS Symbol für Satzende 199  
Episode 295, 449  
Epoche 73, 93, 449  
Erklärbarkeit, KI-System 237, 435  
Erklärungsmodell 238  
Erklärungsmodell, globales 238  
Erklärungsmodell, lokales 239  
Exoskelett 384  
Expertensystem 12, 449  
explodierender Gradient 187, 450  
Exploration beim Bestärkungslernen 296,  
    304, 450  
exponentielles Wachstum 88
- F**  
Fabrik der Zukunft 386  
Facebook 401  
Fairness, KI-System 433
- Faltungsnetz 123. . *Siehe auch* Convolutional Neural Network  
Feedforward-Netz, mehrschichtiges 87,  
    104, 455  
Fehlerinjektion, adaptive 314  
Filteralgorithmen für verbotene Inhalte 392  
Filterblasen 392  
Finanzbranche und KI 391  
FLOPS, Gleitkommaoperationen 134  
Flow Machines Musikerzeugung 355  
Fouriertransformation 251  
Frage-Antwort-Assistent 270  
Fragen beantworten 22, 226  
F-Wert Klassifikationsgüte 76, 450
- G**  
GAFA, US-Internetfirmen 401  
GAM 238. . *Siehe auch* verallgemeinertes additives Modell  
GAN 332. . *Siehe auch* generatives Adversariales Netz  
Gatter 189  
gebogene Trennfläche 82  
gedehntes Convolutional Neural Network 266  
Gemälde, computergeneriert 36  
Genauigkeit der Klassifikation 75, 450  
Generalisierung 56, 75, 450  
generalized additive model 238. . *Siehe auch* verallgemeinertes additives Modell  
generatives adversariales Netz 106, 332, 450  
Generatornetz 332  
gerade Trennfläche 80  
Gerichteter Graph 94, 451  
Geschichte, computergeneriert 38, 347, 351  
Gesichtserkennung 150, 413  
Gewinn, diskontierter zukünftiger 299  
G-Kraft beim Fliegen 417, 451  
globales Erklärungsmodell 238  
5G Mobilfunkstandard 445  
Go-Agent AlphaGo 28  
Google 401, 402  
Google Duplex Sprachgenerierung 26, 269, 426  
Google Home Sprachassistent 25, 249  
GoogLeNet. 133  
Go Spiel 28, 314  
GPT-2 Texterzeugung 349  
GPU 93. . *Siehe auch* Graphical Processing Unit

Gradient 67, 69, 451  
 negativer 69  
 Gradientenabstiegsoptimierung 67, 451  
 Gradientenabstieg, stochastischer 74  
 Gradienten-Clipping 188  
 Grafikprozessor 93, 452  
 Graphical Processing Unit 93  
 Gunrock Chatbot 272  
 gym Simulation von Videospielen 305

**H**

Haftungsfragen und KI 427  
 HAL 9000 KI-System 375, 452  
 halbüberwachtes Lernen 143  
 Hautkrebs 382  
 Hinton, Geoffrey 93  
 hybride KI 422  
 Hyperparameter 72, 96, 452  
 Hyperparameter-Optimierung 112, 113, 137, 282, 452  
 Hyperparameter-Optimierung, ASAP-Methode 113

**I**

ImageNet Bilddaten 15, 130, 143  
 Industrie 4.0 386  
 Informationsextraktion 195, 452  
 Informations-Flaschenhals 106  
 intelligenter Lautsprecher 271. . Siehe auch Smart Speaker  
 intelligenter persönlicher Assistent 270  
 Intelligenz, Dimensionen der - 1  
 Interaktion, statistische 80  
 Internet der Dinge 380, 396, 453  
 Internet of Things 453. . Siehe auch Internet der Dinge  
 Internet-of-Things-Plattform 388  
 Internet und KI 392  
 IoT Internet der Dinge 380, 453

**K**

Kahneman, Daniel 422  
 Keras 96, 453  
 Kernel, Convolution 124, 453  
 Kernel, Schrittweite 125  
 Kettenregel 90, 446, 453  
 KI 1. . Siehe auch Künstliche Intelligenz

KI-Gütesiegel 439  
 Klasse 56  
 Klassifikation 46, 453  
 Klassifikator 56  
 Kleinstquadrat-Verlustfunktion 64  
 Kommunikation und KI 390  
 Konnektionismus 12, 86  
 Kontextvektor 203  
 Kontrafaktisches Schließen 423  
 Konversationsassistent 270  
 Koreferenz 273  
 Koreferenzresolution 273  
 Kosinus-Distanz 177  
 Krankenpflege, KI-System 383  
 kreative Künstliche Intelligenz 393  
 Kreuzattention 204, 208  
 Künstliche Intelligenz 1, 453  
     as a service 400  
 Künstliche Intelligenz, allgemeine 39  
 Künstliche Intelligenz, hybride 422  
 Künstliche Intelligenz, schwache 418  
 Künstliche Intelligenz, starke 418  
 Künstliche Intelligenz, symbolische 11, 23, 230  
 künstliches neuronales Netz 79  
 Kunst und KI 393

**L**

L1-Regularisierung 100  
 L2-Regularisierung 100  
 Labeled Faces in the Wild Gesichtsdaten 150  
 Landwirtschaft und KI 390  
 LeCun, Yann 130  
 Lernen 4  
     assoziativ 423  
     Bestärkungs- 47  
     maschinelles 45  
     selbstüberwachtes 46  
     Situationsänderung 423  
     überwachtes 45  
     unüberwachtes 46  
 Lernen, halbüberwacht 143  
 Lernen, maschinelles 7  
 Lernen, tiefes 7  
 Lernen, Transfer- 141  
 Lernen von Repräsentationen 244  
 Lernrate 72, 453  
 LibriSpeech Sprachdatensatz 253, 259, 263, 264  
 Lidar 318

Likelihood Wahrscheinlichkeit der Trainingsmenge 63  
 LIME-Verfahren 239  
 lineares Modell 81, 238  
 lineare Transformation 59, 454  
 Lippenlesen 415  
 Listen-Attend-Spell Spracherkennung 256  
 Loebner-Preis 276  
 Löschgatter 190  
 logistisches Regressionsmodell 62, 454  
 Log-Likelihood Verlustfunktion 64  
 lokales Erklärungsmodell 239  
 Lokalisation von Objekten in Bildern 119, 144  
 Long Short Term Memory 189, 454  
 LSTM 189. . Siehe auch Long Short Term Memory  
 LSTM-Zelle 191

**M**

MARCO Datensatzes für Webseiten-Ranking 234  
 Markov-Chain-Monte-Carlo-Verfahren 155  
 maschinelles Lernen 7, 45, 455  
 maschinelle Übersetzung 20, 198, 455  
 maskiertes Sprachmodell 222  
 Matrix 51, 455  
     rechteckiges Schema von Zahlen 52  
 Maximum-Entropie Verlustfunktion 64  
 Maximum-Likelihood-Prinzip 63, 455  
 Max-Pooling 127  
 Max-Unpooling 146  
 MCMC 155. . Siehe auch Markov-Chain-Monte-Carlo  
 Medien und KI 390  
 medizinische Bilddaten 17  
 medizinische Bilderkennung 17, 141  
 Meena Chatbot 368  
 mehrschichtiges Feedforward-Netz 87, 104, 455  
 mehrschichtiges RNN 193, 455  
 Mel Frequency Cepstral Coefficients 253  
 MFCC 253. . Siehe auch Mel Frequency Cepstral Coefficients  
 Mikrotargeting 416  
 Militär und KI 416  
 Minibatch 72, 455  
 Minibatch-Gradientenabstieg 74  
 Minsky, Marvin 84  
 mittlerer reziproker Rang 234

Mixture-of-Experts 231  
 MNIST Datensatz zur Ziffernklassifikation 50  
 Modell 9, 49, 62, 456  
 Modellanwendung 74, 456  
 Modellunsicherheit 153  
 Model Predictive Control 321  
 Moments-in-Time Videodaten 277  
 Montezuma's Revenge Atari-Spiel 308  
 MRR@ 10 Kriterium für Websuche 234  
 MS-COCO Bild-Datensatz 235  
 Multi-Head-Attention 211  
 MuseNet Musikerzeugung 356

**N**

Named Entity Recognition 195. . Siehe auch Eigennamenerkennung  
 Natural Language Processing 167  
 Neocognitron Bilderkennung 121  
 NER = Named Entity Recognition 195. . Siehe auch Eigennamenerkennung  
 Nerveninterface 393  
 Nest Smart-Home Firma 394  
 neuronales Netz 6, 456  
 neuronales Netz, künstliches 79  
 neuronales Netz, tiefes 6, 88, 460  
 N-Gramm 181, 456  
 Nichtlineare Funktion 82, 456  
 NLP = Natural Language Processing 167. . Siehe auch Verarbeitung natürlicher Sprache  
 Noise Contrastive Estimation 176  
 notMNIST-Datensatz  
     Buchstabenklassifikation 98

**O**

Objektklassifikation in Bildern 4, 15, 119, 123  
 Objektlokalisation in Bildern 119  
 Objektsegmentierung 119, 145  
 Operator 8, 456  
 Opinionmining 226  
 Optimierung 53  
 Optimierungsschritt 55

**P**

Parameter 7, 54, 456  
 Parameter-Sharing 126

- Parametervektor 54  
     globaler 59
- Parkinson 381
- Patil, Dhanurjay 408
- Penn-Treebank-Textdaten 194, 351
- Perplexität 194, 457
- persönlicher intelligenter Assistent 25
- personalisierte Empfehlungen und KI 392
- Pflegeroboter 384
- Phrase 198
- Pixel 3
- Platform Economy 398
- Plattformökonomie 398
- Pluribus Pokeragent 29
- Poker Spiel 29
- policy gradient 312. . *Siehe auch*  
     Strategiegradient
- Pong Atari-Spiel 305
- Pooling-Schicht 127, 457
- Portraitfoto 36
- Positions-Kodierung 208
- Präzision Klassifikationsgüte 76, 457
- Predictive Maintenance – vorausschauende  
     Wartung 386
- Preisfestsetzung, dynamische 392
- Printmedien und KI 390
- Privacy Preserving Machine Learning 413
- Prognose eines Modells 53, 74
- Prosodie Sprachmelodie 265
- Prothesen, intelligente 384
- Proximal Policy Optimization 313
- Prozessmodell 388
- Punktwert 56
- PyTorch Toolkit 94, 460
- Q**
- Q-Funktion zum Bestärkungslernen 299, 457
- Q-Netz zum Bestärkungslernen, tiefes  
     301, 460
- quadratischer Abstand 104
- Quake III Arena Capture the Flag 30
- Qualitätssicherung in der Produktion 387
- Question Answering 22. . *Siehe auch* Fragen  
     beantworten
- R**
- Radarsensor 318
- Radiologie, Interpretation von Befunden 382
- Ranking-Algorithmus 234
- Recall Klassifikationsgüte 76, 457
- Regelmenge 238
- Regression 46, 457
- Regressionsmodell 104  
     logistisches 62
- Regressionsproblem 104
- Regularisierung 100, 457
- Regularisierung, Batch-Normalisierung 102
- Regularisierung, Dropout 101
- Regularisierung durch stochastischen  
     Gradientenabstieg 103
- Regularisierung, L1- 100
- Regularisierung, L2- 100
- Reinforcement Learning 28. . *Siehe auch*  
     Bestärkungslernen
- rekurrentes neuronales Netz 105, 183, 458
- Relation 170
- Repräsentationen, Lernen von 244
- Residualer Block 133
- residuale Verbindung 133, 212
- ResNeXt-101 Bildklassifikator 137
- rezeptives Feld 124
- RNN 105. . *Siehe auch* rekurrentes  
     neuronales Netz
- RNN, Decoder 458
- RNN, Encoder 458
- Roadrunner Atari-Spiel 305
- Routenplanung 319
- Rückübersetzung 218
- S**
- SAP 401
- Schach 27
- Schachagent AlphaZero 28
- Schachagent Deep Blue 27
- Schicht eines neuronalen Netzes 6, 87
- Schon 421
- Schrittweite eines Kernels 125
- schwache Künstliche Intelligenz 39, 418
- Seaquest Atari-Spiel 305
- Segmentierung, Bild- 145
- Segmentierung, Objekt- 119, 145
- Selbstattention 208
- selbstfahrendes Auto 33
- semisupervised learning 143. . *Siehe auch*  
     halbüberwachtes Lernen
- Sensoren für selbstfahrende Autos 318
- Sensoren in der Produktion 387

- Sentimentanalyse 226, 365  
Sequence-to-Sequence Modell 199, 458  
Sequenz 105, 183  
Serious Games 382  
Sicherheit, KI-System 435  
Sigmoid-Funktion 83  
Simulationsmodell für das Lernen 424  
Simultanübersetzung 21, 219  
Singularität, technologische 418  
Siri Sprachassistent 25, 249, 375  
Skalar, einzelne Zahl 52, 458  
Skalarprodukt 58  
Smart City 396  
Smart Home 377, 378  
Smart Speaker 271  
Social Credit in China 414  
Softmaxfunktion 61, 458  
Softwareagent 30  
Software-Architekt 410  
Speichergatter 190  
Spielkonsole Atari 2600 30  
Sprachassistent 270  
Spracherkennung 19, 249  
Spracherzeugung 265  
Sprachmelodie 265  
Sprachmodell 184, 458  
Sprachmodell, maskiert 222  
Sprachsynthese 26, 265. . Siehe auch  
    Spracherzeugung  
Squad, Frage-Antwort-Daten 23, 226  
StackGAN TNN zur Bilderzeugung 345  
Starcraft 2 Videospiel 33  
starke Künstliche Intelligenz 39, 418  
stochastischer Gradientenabstieg 74, 459  
stochastische Strategie 310, 459  
Strategie beim Bestärkungslernen 48, 106,  
    298, 459  
Strategiegradient 310, 312  
Strategiegradientenmodell 459  
Strategie-Netz, tiefes 311  
Student Model 143  
Sutton, Richard 313  
SWAG Daten zum Textverständnis 228  
Switchboard Sprachdaten 19, 253, 259,  
    263, 264  
Symbol 170  
Symbol-Grounding 170, 425  
symbolische Künstliche Intelligenz 11, 23  
Synapsen 114  
Synonym 172  
Synthese von Arzneimitteln 380
- T**  
Tangens Hyperbolicus 83  
Teacher Model 143  
technologische Singularität 418  
Tencent chinesische Internetfirma 402  
Tensor 52, 459  
TensorFlow Toolkit 94, 460  
Testbeispiel 75, 460  
Testmenge 75, 460  
Text-to-Speech System 265  
Therapieempfehlung, KI-System 383  
Thrun, Sebastian 317  
tiefes Lernen 7, 88, 460  
tiefes Lernen, Toolkit zum 460  
tiefes neuronales Netz 6, 88, 460  
T-NLG Sprachmodell 354  
TNN 6, 460. . Siehe auch tiefes  
    neuronales Netz  
Toolkit zum tiefen Lernen 94, 460  
Top5-Fehlerrate 16, 131  
Training 53, 461  
Trainingsbeispiel 4, 75, 461  
Trainingsdaten 4  
Trainingsmenge 4, 75, 461  
Transferlernen 141, 221, 224, 461  
Transformation  
    lineare 59  
Transformer 208  
Transformer Decoder 212  
Transformer Encoder 212  
translationsinvariant 125  
Transparenz, KI-System 434  
Transposed-Convolution 148  
Trennfläche, gebogene 82  
Trennfläche, gerade 80  
TTS 265. . Siehe auch Text-to-Speech System  
Turing Natural Language Generation  
    (T-NLG) 354  
Turing-Test 2
- U**  
Überanpassung 97, 461  
Übersetzung, maschinelle 20, 198, 208  
Übersetzung, Simultan- 21, 219  
überwachtes Lernen 45, 461  
Überwachung durch KI 413

- Umgebung  
deterministische 295  
stochastische 296
- Unicorn Startup Unternehmen 385
- unscharfes Schließen 422
- Unsicherheit wegen fehlender  
Trainingsabdeckung 153
- Unsicherheit wegen Messfehlern 153
- Unteranpassung 97
- unüberwachtes Lernen 462
- V**
- Validationsmenge 96, 462
- Vektor 51, 462  
Reihe von Zahlen 52
- verallgemeinertes additives Modell 238
- Verankerung von Konzepten 170, 425
- Verarbeitung natürlicher Sprache 167, 462
- verdeckter Vektor 84, 462
- Verkehr und KI 392
- Verlässlichkeit, KI-System 436
- Verlust 53
- Verlustfunktion 54, 462  
Kleinstquadrat 64  
Log-Likelihood 64  
Maximum-Entropie 64
- verschwindender Gradient 188, 463
- verzögerter Satz 186
- Videoinhaltsanalyse 277
- Videokameras 318
- Videoklassifikation 277
- Videospiel Capture the Flag 30
- Videospiel Dota2 32
- Videospiele Atari 2600 30
- Videospiel Starcraft 2 33
- Videoüberwachung 415
- Video-Untertitelung 277
- Vokabular 173
- vollverbundene Schicht 85, 463
- vorzeitiges Stoppen 98
- W**
- Wahrnehmungsalgorithmus 319
- Wahrscheinlichkeitsvektor 60, 463
- Wartung, vorausschauende 386
- Wearables 396
- Websuche, KI-unterstützt 234
- WebText Textdatensatz 351
- Weight Decay 100
- Weltwissen 228, 426
- WER 253. . Siehe auch Wortfehlerrate
- white box-Attacke 162
- WiFi 463
- wireless local area network 463
- Wissensdatenbank 228, 421
- Wissensdatenbank Cyc 228
- WLAN 463
- WMT 2014 En-De-Übersetzungsdaten 216
- WMT 2014 En-Fr Übersetzungsdaten 206, 216
- WMT 2017 Übersetzungsdaten 207
- Word2Vec Embedding-Netz 173
- Wordsmith Texterzeugung 347
- Wort-Embedding 449
- Wortfehlerrate 19, 253
- X**
- XiaoIce Chatbot 359
- XOR-Problem 79
- Z**
- Zahlenpaket 7
- Zeitreihe von numerischen Daten 196
- Zustand beim Bestärkungslernen 295