

daniel BASLER



# Neuronale Netze mit C# programmieren

Mit praktischen Beispielen für Machine  
Learning im Unternehmenseinsatz



Beispielcode unter  
[plus.hanser-fachbuch.de](http://plus.hanser-fachbuch.de)

HANSER

## Neuronale Netze mit C# programmieren



### Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf [plus.hanser-fachbuch.de](http://plus.hanser-fachbuch.de) einfach diesen Code ein:

plus-5db90-her61



### Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

[www.hanser-fachbuch.de/newsletter](http://www.hanser-fachbuch.de/newsletter)





Daniel Basler

# Neuronale Netze mit C# programmieren

Mit praktischen Beispielen  
für Machine Learning  
im Unternehmenseinsatz

HANSER

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



Bibliografische Information der Deutschen Nationalbibliothek:

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

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2021 Carl Hanser Verlag München, [www.hanser-fachbuch.de](http://www.hanser-fachbuch.de)

Copy editing: Walter Saumweber, Ratingen

Umschlagdesign: Marc Müller-Bremer, [www.rebranding.de](http://www.rebranding.de), München

Umschlagrealisation: Max Kostopoulos

Titelmotiv: © [istockphoto.com/Artystarty](http://istockphoto.com/Artystarty)

Layout: Manuela Treindl, Fürth

Druck und Bindung: Kösel, Krugzell

Ausstattung patentrechtlich geschützt. Kösel FD 351, Patent-Nr. 0748702

Printed in Germany

Print-ISBN: 978-3-446-46229-8

E-Book-ISBN: 978-3-446-46426-1

E-Pub-ISBN: 978-3-446-46635-7

# Inhalt

<b>Vorwort</b> .....	<b>XIII</b>
<b>Aufbau des Buches</b> .....	<b>XV</b>
<b>1 Künstliche Intelligenz</b> .....	<b>1</b>
1.1 Grundlagen .....	1
1.1.1 Schwache künstliche Intelligenz .....	2
1.1.2 Starke künstliche Intelligenz .....	3
1.1.3 Hybride künstliche Intelligenz .....	3
1.2 Themenfelder der künstlichen Intelligenz .....	4
1.2.1 Machine Learning .....	5
1.2.2 Deep Learning .....	5
1.2.3 Cognitive Computing .....	6
1.2.4 Big Data und Data Science .....	6
1.2.5 Predictive Analytics .....	7
1.2.6 Natural Language Processing .....	7
1.3 KI-Service-Plattformen .....	8
1.3.1 Amazon .....	8
1.3.2 Google .....	9
1.3.3 Microsoft Cognitive Services .....	11
1.3.4 IBM .....	12
1.4 Künstliche neuronale Netze .....	13
1.4.1 Funktionsweise .....	13
1.4.2 Netztypen .....	14
1.4.3 Anwendungsbereiche .....	16
1.5 Grundbaustein Neuron .....	16
1.5.1 Aktivierungsfunktion .....	17
1.5.2 Matrizendarstellung .....	20
1.6 Architekturprinzipien .....	21
<b>2 Konzepte und Methoden von Machine Learning</b> .....	<b>23</b>
2.1 ML - Machine Learning .....	23
2.2 Algorithmen und Modelle .....	25

2.3	Die Schritte in einem Machine-Learning-Projekt	26
2.4	Machine-Learning-Verfahren	28
2.4.1	Klassifikation	29
2.4.2	Regression	29
2.4.3	Clustering	29
2.4.4	Bayes-Klassifikation	30
2.4.5	Künstliche neuronale Netze	30
2.5	Lernformen	31
2.5.1	Überwachtes Lernen	31
2.5.2	Unüberwachtes Lernen	31
2.5.3	Semi-überwachtes Lernen	32
2.5.4	Verstärkendes Lernen	32
2.6	Machine-Learning-Algorithmen	33
2.6.1	$k$ -Nearest-Neighbour	34
2.6.2	Support Vector Machine	35
2.6.3	Entscheidungsbäume	37
2.6.4	Decision Tree und Random-Forest	38
2.6.5	Clustering	38
2.6.5.1	K-Means Clustering	38
2.6.5.2	EM-Clustering	39
2.6.5.3	Hierarchische Clusteranalyse	39
2.7	Training und Validierung des ML-Modells	40
2.8	Das einfache neuronale Netz	41
2.9	Deep Learning	48
2.10	Einsatzgebiete und Anwendungen	49
<b>3</b>	<b>Neuronale Netze</b>	<b>51</b>
3.1	Vom Problem zum KNN	51
3.2	KNN-Modelle	52
3.3	Mathematik neuronaler Netze	55
3.3.1	Lineare Algebra	55
3.3.2	Vektor	56
3.3.2.1	Rechnen mit Vektoren	57
3.3.2.2	Skalarprodukt	58
3.3.3	Matrix	58
3.3.3.1	Rechnen mit Matrizen	59
3.3.3.2	Matrizenmultiplikation	59
3.3.3.3	Transponieren	61
3.3.4	Tensor	61
3.3.5	Eigenwert- und Singulärwertzerlegung	61
3.4	Mehrschichtige neuronale Netze	62
3.4.1	Multilayer Perceptron (MLP)	62
3.5	Predictive Maintenance	65

3.6	Maschinensimulation mit MLP	67
3.6.1	Datenmodellierung	67
3.6.1.1	Ziel des Feedforward-Netzes	68
3.6.1.2	Mehrklassen-Klassifikation	68
3.6.2	Entwurf	70
3.6.3	Projekt anlegen	71
3.6.4	Erfassung und Berechnung der Daten	73
3.6.5	Bias-Neuron	75
3.6.6	Die Programmierung	76
3.6.7	Aktivierungsfunktionen implementieren	85
3.6.8	Fazit	86
3.7	Lernalgorithmus für Neuronen	87
3.7.1	Kostenfunktion	87
3.7.2	Gradientenabstiegsverfahren	88
3.7.3	Backpropagation-Algorithmus	89
3.8	Backpropagation programmieren	92
3.9	Implementierung	97
<b>4</b>	<b>Training von neuronalen Netzen</b>	<b>111</b>
4.1	Trainings- und Testphase	111
4.1.1	Generalisierung	112
4.1.2	Dimensionsreduzierung	112
4.2	Batch-, inkrementelles und Mini-Batch-Training	113
4.2.1	Batch-Training	113
4.2.2	Inkrementelles Training	113
4.2.3	Mini-Batch-Training	114
4.3	Lernprozess beim Backpropagation-Algorithmus	114
4.3.1	Problemstellung	116
4.3.2	Vorbereiten der Daten	116
4.3.3	Das neuronale Netz programmieren	117
4.3.4	Benutzeroberfläche	118
4.3.4.1	Code-Behind der MainWindow-Klasse	121
4.3.4.2	Nutzen der Hold-Out Validation	124
4.3.5	Programmablauf	127
4.3.6	Das neuronale Netz implementieren	127
4.3.7	Auswertung ermitteln	137
4.4	Simulationsergebnis	138
4.5	Parameteranpassungen	140
<b>5</b>	<b>Recurrent Neural Networks</b>	<b>141</b>
5.1	Sequenzen und Rückkopplung	142
5.2	Architektur eines RNN	144
5.3	Backpropagation Through Time	147

5.4	Long Short-Term Memory Networks	149
5.4.1	Funktionsweise von LSTMs	151
5.4.1.1	Forget-Gate	152
5.4.1.2	Input-Gate	152
5.4.1.3	Output-Gate	154
5.4.1.4	Zusammenfassung	154
5.4.2	Gradient Clipping	155
5.4.3	Varianten	155
5.4.4	LSTM-Implementierung	156
<b>6</b>	<b>Convolutional Neural Networks</b>	<b>159</b>
6.1	Aufbau eines CNN	160
6.2	Detektionsteil	162
6.2.1	Kantenerkennung	162
6.2.2	Pooling	164
6.2.3	Schrittweite	165
6.2.4	2D- und 3D-Volumen	165
6.2.5	Aktivierungsfunktion	166
6.2.6	Ein sehr einfaches CNN	167
6.2.7	Subsampling	168
6.2.8	CNN mit Pooling Layer	169
6.3	Identifikationsteil	170
6.4	Schlussbemerkung	171
<b>7</b>	<b>Machine Learning Frameworks</b>	<b>173</b>
7.1	Einbindung von ML-Frameworks in C#	174
7.2	TensorFlow	175
7.2.1	Ablauf in TensorFlow	176
7.2.2	Das TensorBoard	177
7.2.3	Begriffe	178
7.2.4	TensorFlow Playground	178
7.3	Keras	179
7.4	Infer.NET	180
7.4.1	Probabilistische Programmierung	181
7.4.2	Arbeitsweise von Infer.NET	182
7.4.3	Infer.NET-Architektur	184
7.4.4	Infer.NET Modelling-API	185
7.4.5	Lernen und Trainieren	186
7.4.6	Infer.NET in der Anwendung	186
7.4.7	Das Modell entwerfen	187
7.4.8	Infer.NET anwenden	188
7.5	ML.NET mit AutoML und ModelBuilder	192
7.5.1	Einbinden von ML.NET	193

7.5.2	Was ist AutoML	193
7.5.3	Model Builder	195
7.5.4	Einbinden in das Projekt	195
7.5.5	Szenario	196
7.5.6	Daten	197
7.5.7	Training und Auswertung	200
7.5.8	Der Code	200
7.5.9	Automatisiert modellieren	201
7.5.10	Die Kommandozeile (CLI)	207
7.5.11	Die Zukunft von AutoML	207
7.6	Benutzerdefiniertes ML.NET	208
7.6.1	ML.NET-Komponenten	209
7.6.2	Benutzerdefinierter Workflow	211
7.6.3	Erstellen einer benutzerdefinierten Anwendung	212
7.6.4	Datentransformation	214
7.6.5	ML.NET-Algorithmus	215
7.6.6	Erstellen und Trainieren eines ML-Modells	215
7.6.7	Modellauswertung	216
7.6.8	Modellbereitstellung	218
7.6.9	TensorFlow, ONNX und ML.NET	218
<b>8</b>	<b>SciSharp Stack</b>	<b>221</b>
8.1	TensorFlow.NET	222
8.1.1	TensorFlow.NET-SDK installieren	222
8.1.2	Tensor	224
8.1.3	Platzhalter	225
8.1.4	Variable	226
8.1.5	Konstante	227
8.1.6	Berechnungsgraph	228
8.1.7	Lineare Regression	229
8.1.8	Von der Theorie zum Code	230
8.2	Keras.NET	233
8.2.1	Keras.NET installieren	233
8.2.2	Modelle erstellen	234
8.3	NeuralNetwork.NET	236
<b>9</b>	<b>Machine Learning as a Service</b>	<b>237</b>
9.1	Amazon Machine Learning und KI-Services	238
9.1.1	Amazon Lex	239
9.1.2	Die Lex-Chatbot-Struktur	240
9.1.3	Entwickeln mit AWS-Lambda-Funktionen	242
9.2	Erstellen eines Lex-Chatbots für .NET	244
9.2.1	Erste Schritte	244

9.2.2	Beispiel Chatbot . . . . .	245
9.2.3	Intents. . . . .	247
9.2.4	Testen Sie den Bot . . . . .	249
9.2.5	AWS-Lambda-Funktion . . . . .	251
9.2.6	Slots . . . . .	255
9.2.7	Error Handling. . . . .	255
9.2.8	Konfigurieren von Cognito . . . . .	256
9.2.9	Die Web-Applikation . . . . .	257
9.3	Azure Cognitive Services . . . . .	259
9.3.1	Intelligente kontextbasierte Suchfunktion . . . . .	260
9.3.1.1	Bing-Websuche . . . . .	260
9.3.1.2	Bing Suche über REST API. . . . .	262
9.3.1.3	Die eigene Suchmaschine. . . . .	263
9.4	Azure Machine Learning Studio . . . . .	269
9.4.1	Arbeitsbereich . . . . .	269
<b>10</b>	<b>Anwendungen entwerfen . . . . .</b>	<b>273</b>
10.1	Predictive Analytics . . . . .	273
10.1.1	Fallbeispiel: Energiebranche . . . . .	274
10.1.2	Zeitreihenanalyse . . . . .	274
10.1.3	Beispielprogramm und Anwendung der Prognose . . . . .	276
10.1.4	Definieren der Pipeline. . . . .	277
10.2	Bildklassifikation . . . . .	280
10.2.1	Benötigte Daten . . . . .	280
10.2.2	Projekt konfigurieren . . . . .	281
10.2.3	Importieren des MNIST-Datensatzes. . . . .	283
10.2.4	Aktivierungsfunktion . . . . .	287
10.2.5	Input Layer . . . . .	288
10.2.6	Hidden Layer . . . . .	289
10.2.7	Output Layer . . . . .	291
10.2.8	Neural Network . . . . .	293
10.2.9	Initialisierung und Auswertung . . . . .	297
10.2.10	Training und Backpropagation . . . . .	300
10.2.11	Auswertung und Verbesserung . . . . .	301
10.3	Visuelle Muster erkennen . . . . .	302
10.3.1	Aufgabenstellung. . . . .	302
10.3.2	Convolutional Layer. . . . .	303
10.3.3	Pooling Layer . . . . .	303
10.3.4	Flatten Layer . . . . .	305
10.3.5	Fully Connected Layer . . . . .	306
10.3.6	Methoden. . . . .	306
10.3.7	Training . . . . .	307
10.4	Objekterkennung . . . . .	309

10.4.1	Transferlernen mit ML.NET	310
10.4.2	Neue Bilddaten vorbereiten	311
10.4.3	Trainiertes TensorFlow-Modell verwenden	313
10.4.4	MLContext, Pipeline und Prognose	315
10.5	Natural Language Processing	317
10.5.1	Textklassifikation	318
10.5.2	Merkmalsvektoren (Feature Vectors)	320
10.5.3	Texterkennung mit CNN	322
10.5.4	Textklassifikation mit RNN	323
10.5.5	Word Embedding mit ML.NET	325
10.5.6	Stoppwörter	328
10.6	Stanford CoreNLP für .NET	331
10.7	Sentiment-Analyse	333
10.7.1	Sentiment	333
10.7.2	Sentiment-Analyse mit ML.NET	333
10.7.3	Sentiment-Analyse mit AutoML	338
10.7.4	Modell erstellen mit dem Model Builder	338
10.7.5	Das Modell als Web-App	343
	<b>Referenzen und Quellen</b>	<b>347</b>
	<b>Stichwortverzeichnis</b>	<b>351</b>



# Vorwort

Neuronale Netze sind seit einiger Zeit überall im Gespräch. Als Softwareentwickler stellt man fest, dass es zurzeit keinen anderen Bereich in der Softwareentwicklung gibt, der sich so rasant verändert und weiterentwickelt.

Eine attraktive Alternative zu Python für den Entwurf von neuronalen Netzen ist eine modulare und objektorientierte Programmiersprache wie Microsoft C#. Die Objektorientierung bietet den Vorteil, dass sich sehr gut modulare Machine-Learning-Modelle entwerfen lassen, die konfigurierbar, erweiterbar und wiederverwendbar sind.

Dieses Buch richtet sich an C#-Entwickler, die einen möglichst umfassenden Blick über neuronale Netze erlangen wollen. Es möchte Sie beim Kennenlernen, Experimentieren und Arbeiten mit neuronalen Netzen und Machine-Learning-Modellen anleiten und unterstützen. Dabei wendet es sich an im Umgang mit neuronalen Netzen unerfahrene Programmierer.

Sie sollten über Grundkenntnisse in der Programmierung mit C# verfügen, sodass Begriffe wie Variablen, Schleifen etc. Ihnen vertraut sind. Wegen des mathematischen Anteils müssen Sie sich keine Sorgen machen. Um die Buchinhalte zu verstehen, sind wirklich nur gute Kenntnisse in dem Konzept der linearen Algebra erforderlich.

Insgesamt erhebt das Buch auch keinen Anspruch auf Vollständigkeit. Es verzichtet auf tiefgehende mathematische und programmiertechnische Details, die nicht wirklich notwendig sind, um die Programmierung von neuronalen Netzen und Machine Learning zu verstehen.

Anhand von Anwendungsbeispielen lernen Sie neuronale Netze und Machine-Learning-Modelle zu entwickeln. Sie lernen auf diese Weise dynamische Datenstrukturen, Feedforward-Netze, Backpropagation-Algorithmen sowie Convolutional Neural Networks und Natural Language Processing kennen. Das Buch möchte Ihnen die Leistungsvielfalt neuronaler Netze vermitteln und Ihnen helfen diese in eigenen Programmierprojekten zu nutzen.

Den Mitarbeiterinnen und Mitarbeitern des Hanser-Verlages, besonders Frau Sylvia Hasselbach, danke ich für die Sorgfalt und Unterstützung bei der Veröffentlichung dieses Buches.

Ihnen, liebe Leserin und lieber Leser, wünsche ich viel Freude und Erfolg beim Kennenlernen und Arbeiten mit neuronalen Netzen mit C#.

*Daniel Basler*

Herford, April 2021



# Aufbau des Buches

Die wichtigsten Begriffe und Konzepte werden gleich in den ersten beiden Kapiteln erläutert. Die weiteren Kapitel sind thematisch so aufgebaut, dass man das Buch von vorne nach hinten durcharbeiten kann.

Nachfolgend erhalten Sie einen Überblick über den Inhalt des Buches:

- **Kapitel 1** führt in das Thema künstliche Intelligenz ein. Dabei geht es um die Anwendungsgebiete, den Aufbau von neuronalen Netzen und das Neuron als Grundbaustein.
- **Kapitel 2** beschreibt die Konzepte und Methoden von Machine Learning. Es nennt wichtige Algorithmen und zeigt ein erstes einfaches neuronales Netz als Programmierbeispiel.
- **Kapitel 3** beschreibt Schritt für Schritt den Aufbau eines künstlichen neuronalen Netzes in C# und geht auf die wichtigsten Mathematik-Grundlagen ein. Anhand von Aufgabenstellungen werden mehrschichtige neuronale Netze entwickelt. Des Weiteren beschäftigen Sie sich in diesem Kapitel mit Lernalgorithmen und der Programmierung des Backpropagation-Algorithmus.
- **Kapitel 4** zeigt das Training von neuronalen Netzen. Sie lernen hier unterschiedliche Lernprozesse kennen und implementieren Trainingsmethoden in ein neuronales Netz.
- **Kapitel 5** führt Sie weiter zu den sogenannten Recurrent Neural Networks (RNN), die hauptsächlich in der Verarbeitung von Textsequenzen oder Zeitreihen eingesetzt werden. Sie lernen, wie die einzelnen Schichten in einem RNN aufgebaut sind.
- **Kapitel 6** erläutert ausführlich den Aufbau von Convolutional Neural Networks (CNN), die vornehmlich für die Verarbeitung von Bild- und Audiodateien eingesetzt werden. Ein CNN ist ein spezielles mehrschichtiges Feedforward-Netz.
- **Kapitel 7** beschäftigt sich mit den im Moment aktuellen Machine Learning Frameworks, die Sie bei der Entwicklung von Machine-Learning-Modellen in C# unterstützen können.
- **Kapitel 8** beschreibt ein weiteres nützliches ML-Framework im .NET Umfeld. Der sogenannte SciSharp-Technologie-Stack ermöglicht es, ML-Modelle für TensorFlow mit C# zu erstellen. Des Weiteren werden in diesem Kapitel die Frameworks Keras.NET und NeuralNetwork.NET vorgestellt.
- **Kapitel 9** gibt einen Einblick in Machine Learning as a Service und stellt Amazon Lex und Azure Cognitive Services an einem Anwendungsbeispiel näher vor.
- **Kapitel 10** setzt die in den vorherigen Kapiteln aufgezeigten Methoden in einzelnen Beispielanwendungen für Zeitreihenanalyse, Bildklassifikation, Objekterkennung und Natural Language Processing um.

## Programmbeispiele

Um den Praxisbezug zu gewährleisten, wird gezeigt, wie die beschriebenen Themen in C#-Programme umgesetzt werden. In den meisten Fällen wird der vollständige Beispielcode abgebildet, um die Programmierung in C# zu verdeutlichen. Sie können die Buchbeispiele komplett von meinem GitHub-Repository herunterladen (<https://github.com/DanielBasler/NeuralNetwork>) oder von der Plus.Hanser-Webseite:



### Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel die Code-Beispiele aus dem Buch. Geben Sie auf [plus.hanser-fachbuch.de](https://plus.hanser-fachbuch.de) einfach diesen Code ein:

```
plus-5db90-her61
```

## Zielsetzung des Buches

Mein Ziel ist es, Ihnen ein solides Fundament für das Entwerfen und Entwickeln von neuronalen Netzen und Machine-Learning-Modellen an die Hand zu geben, unterstützt von praktischen Beispielen. Des Weiteren möchte ich, dass Sie ein Gefühl für den Programmier-Aufwand von neuronalen Netzen und deren Leistung bekommen und die Vielfalt der Einsatzmöglichkeiten aber auch der Anforderungen kennenlernen.

Alle hier verwendeten Softwaretools, mit Ausnahme der KI-Services, sind kostenlos und Open Source. Die Beispiele lassen sich mit Visual Studio Community Version 2019 entwickeln und auf einem „normalen PC“ ohne besondere Hardware-Power erstellen und ausführen.

# 1

# Künstliche Intelligenz

„Okay Google“, „Alexa, spiel bitte ‚Eight days a week‘ von den Beatles“ oder „hey Siri“... kennen Sie vermutlich alle. Die sogenannte künstliche Intelligenz, kurz nur noch als KI bezeichnet, ist schon längst in unserem Alltag angekommen und dringt in immer mehr Bereiche vor.

Das heißt, wir sind alle mittendrin, ob es um Sprachassistenten als Helfer auf dem Smartphone, im Auto oder zu Hause geht, die digitale Vernetzung in allen Lebensbereichen schreitet mit unglaublicher Geschwindigkeit voran. Mit der digitalen Transformation, durch die auch die KI allgegenwärtig wird bzw. ist, müssen sich auch Unternehmen auf eine Umstrukturierung und gegebenenfalls sogar auf eine Neuorientierung ihrer Geschäftsprozesse einstellen. Heute müssen Sie sich als Entwickler mit Begriffen wie digitaler Zwilling, Product Performance Management und digitale Fabrik auseinandersetzen. So stellt zum Beispiel das Zusammenspiel von Robotic Process Automation (RPA) und künstlicher Intelligenz ganz neue Symbiosen im Bereich der Automation dar.

In der Diskussion über KI tauchen heute sehr viele verschiedene Begriffen auf. Es ist die Rede von Machine Learning, neuronalen Netzen, Representation Learning, Natural Language Processing (NLP) oder auch Deep Learning. Selbst Begriffe wie Big Data und Data Science werden in die Runde geworfen. Sie sehen also, wer sich mit KI befasst, wird sehr schnell mit einem Begriffswirrwarr von Schlagwörtern überzogen, die durchaus für Verwirrung sorgen können.

Man kann KI als Überbegriff sehen, unter dem unterschiedliche Techniken und Bezeichnungen versammelt sind, und bevor wir uns dem Hauptthema des Buches – der Programmierung neuronale Netze – widmen, möchte ich einige Begriffe und Technologien klären, die in diesem Buch benutzt werden.

## ■ 1.1 Grundlagen

Dabei ist KI keine neue Wissenschaft oder Technologie. Die Anfänge der KI-Forschung gehen bis in die 1950er-Jahre zurück, in denen Alan Turing den Aufsatz „Computing Machinery and Intelligence“ [1] vorgelegt hat. Auf Turing geht auch der nach ihm benannte Turing-Test zurück, der dazu dient, zu unterscheiden, ob eine Maschine ein gleichwertiges Denkvermögen aufweist wie ein Mensch oder nicht.

Zum ersten Mal wurde der Begriff „artificial intelligence“ von John McCarthy [2] verwendet bzw. geprägt. Laut McCarthy ist KI eine Informations- und Ingenieurwissenschaft, die dem Herstellen „intelligenter“ Maschinen und speziellen intelligenten Computerprogrammen gewidmet ist. Für McCarthy besteht der rechnerische Teil der Intelligenz in der Fähigkeit, die Ziele in der Welt zu erreichen. Das heißt für ihn, ein Computer soll so gebaut oder programmiert werden, dass er eigenständig Programme bearbeitet und löst, aus den gemachten Fehlern lernt, Entscheidungen trifft und mit seiner Umgebung kommuniziert.

Aufgrund dessen kann man auch vereinfacht sagen, KI beschäftigt sich mit der Entwicklung von Systemen, die eigenständig Probleme lösen und analog zu menschlichen Denk- und Verhaltensmustern intelligent handeln.

Dass die KI seit den letzten Jahren einen dermaßen großen Boom erlebt, hat sie folgenden Faktoren zu verdanken:

- Die Menge der zur Verfügung stehenden Daten hat extrem zugenommen. Durch Big Data [3] erweitern sich auch die Anwendungsfelder für den Einsatz von KI.
- Die Daten die über Big Data gewonnen werden, lassen sich immer preisgünstiger speichern. Die Preise für Storage sind in den vergangenen Jahren deutlich gesunken. Das macht das Speichern und Auswerten großer Datenmengen inzwischen für Unternehmen rentabel.
- Grafikprozessoren (GPU) werden günstiger und leistungsfähiger. Durch den Einsatz von GPUs können Berechnungen massiv parallelisiert und dadurch beschleunigt werden. Vor allem in diesem Bereich waren die Fortschritte in den vergangenen Jahren enorm.
- Frameworks und Cloud Services erleichtern den Einsatz. Neben der Rechenleistung und dem Speicher in der Cloud können hier auch ML-Modelle erstellt und auf verschiedenen Rechnern in der Cloud-Infrastruktur getestet werden. Hier haben Frameworks wie TensorFlow und vorgefertigte KI-Services aus der Cloud die Einstiegsschwelle deutlich gesenkt.

Durch diese Faktoren finden immer mehr KI-Lösungen ihren Weg in die Praxis. Allerdings sprechen wir hier nur über die sogenannte schwache KI, die inzwischen sehr gut erforscht und sich in zahlreichen Produkten und Diensten etabliert hat.

### 1.1.1 Schwache künstliche Intelligenz

Als schwache künstliche Intelligenz (engl. *weak AI* oder *narrow AI*) werden Systeme bezeichnet, die auf die Lösung konkreter Anwendungsprobleme fokussieren.

Das heißt, bei der schwachen KI geht es um die Simulation eines gewissen Bereiches des intelligenten Verhaltens mit Mitteln der Mathematik und der Informatik. Mit schwacher KI haben wir es mittlerweile im Alltag ständig zu tun. Beispiele hierfür sind unter anderem:

- Zeichen- und Texterkennung
- Bilderkennung
- Spracherkennung
- Automatische Übersetzung
- Expertensysteme
- Navigationssysteme
- Autovervollständigung und Korrekturvorschläge bei Suchvorgängen

Besondere Aufmerksamkeit erhalten hier auch die sogenannten intelligenten Chatbots, die den Kundensupport revolutionieren sollen. Aber auch die Buchhaltung wird heute schon mit schwacher KI automatisiert und optimiert, ebenso wie der Posteingang beim E-Mail-Verkehr und Aufgaben im Backoffice allgemein.

### 1.1.2 Starke künstliche Intelligenz

Starke künstliche Intelligenz (engl. *strong AI* oder *general AI*) hat das Ziel, eine Intelligenz zu erschaffen, die menschliches Denken, Bewusstsein und Emotionen oder die gleichen intellektuellen Fertigkeiten von Menschen erreichen oder übertreffen kann.

Starke KI müsste somit folgende Eigenschaften aufweisen:

- Logisches Denkvermögen
- Entscheidungsfähigkeit auch bei Unsicherheit
- Planungs- und Lernfähigkeit
- Fähigkeit zur Kommunikation in natürlicher Sprache
- Kombinieren aller Fähigkeiten

Allerdings ist es bis heute noch nicht gelungen, eine starke KI zu entwickeln. Auch die Diskussion, ob eine solche Entwicklung überhaupt möglich ist, hält weiterhin an. Verstärkt finden wir heute schon Ansätze in der Form der hybriden KI.

### 1.1.3 Hybride künstliche Intelligenz

Bei der hybriden künstlichen Intelligenz handelt es sich um eine KI-Technologie die versucht in Verbindung mit agiler Optimierung in der Unternehmensplanung oder auch in der Operationsforschung eine Reihe von Prozess- Technologien und -Modellen mit Machine Learning und Deep Learning zu vereinen.

Das heißt, man macht sich die Algorithmen aus Machine Learning zunutze um entsprechende ML-Modelle zu entwickeln, welche wiederum mit menschlichem Expertenwissen hinsichtlich Geschäftsprozessen, Verhaltensmustern und Planungszielen erweitert und trainiert werden. Somit ist es möglich, potenzielle Entscheidungsräume innerhalb dieser Modelle in kürzester Zeit durch den Experten auszuloten und zu bewerten.

Google nutzt die hybride künstliche Intelligenz schon bei der Sprachsuche. Wird von dem ML-System eine Äußerung richtig verstanden, folgt darauf das Markieren eines Treffers durch einen menschlichen Experten. Das ML-System ist so angelegt, dass es sich durch diese Markierung selbst trainieren kann, es verbessert so seine Erkennungsmöglichkeiten, je häufiger es eingesetzt wird.

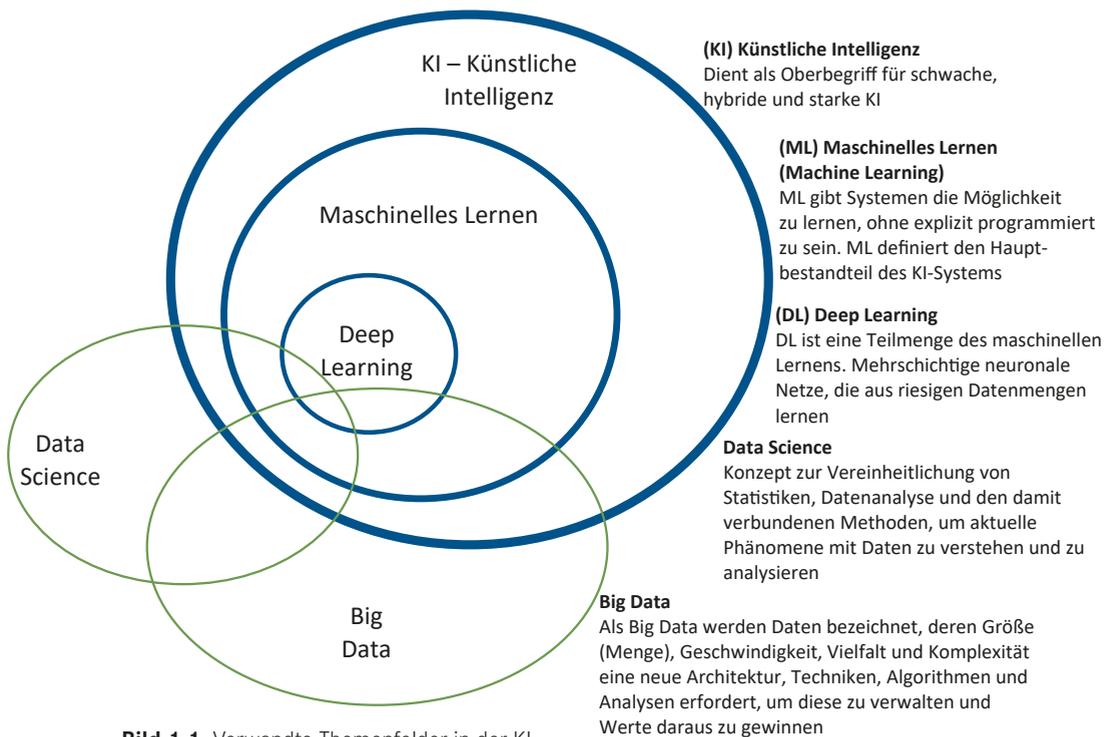
## ■ 1.2 Themenfelder der künstlichen Intelligenz

In den meisten Fällen wird KI als Bezeichnung für Computersysteme benutzt, die Aufgaben abarbeiten, nachdem sie mit großen Datenmengen trainiert wurden. Die Industrie fasst den Begriff KI noch weiter, hier versteht man KI als technologische Methode, die es ermöglicht, menschliche Wahrnehmung und Handeln durch Maschinen nachzubilden. Die Einsatzgebiete der KI sind dabei sehr vielfältig. So gehört auch der erfolgreiche Einsatz in der Robotik dazu, wenn es z. B. im Maschinenbau um die Bearbeitung in Bereichen von Tausendstel-Millimetern geht.

Des Weiteren spielt auch Robotic Process Automation (RPA) eine große Rolle. Hier werden Routine-Aufgaben automatisiert, indem der Software-Roboter diese nachahmt. In einigen Fällen kann sogar der gesamte Geschäftsprozess durch einen Software-Roboter abgebildet werden.

Allerdings wird KI immer nur als reiner Oberbegriff behandelt, es ist daher wichtig zu klären, was man wirklich meint, wenn man über künstliche Intelligenz, maschinelles Lernen und Deep Learning spricht und wie sich die Begriffe zueinander verhalten.

Das Mengendiagramm in Bild 1.1 zeigt, dass künstliche Intelligenz ein umfangreiches Set an Methoden, Verfahren und Technologien bildet. So stellt jeder Kreis im Mengendiagramm eine entsprechende KI-Technologie dar und somit auch, dass die KI zwar das maschinelle Lernen und damit auch Deep Learning einschließt, darauf aber nicht begrenzt ist. Im täglichen Umgang werden die aufgezeigten Begriffe sehr häufig synonym gebraucht.



**Bild 1.1** Verwandte Themenfelder in der KI

### 1.2.1 Machine Learning

Machine Learning, kurz als ML bzw. auch als maschinelles Lernen bezeichnet, ist eines der bekanntesten Teilgebiete der Künstlichen Intelligenz und eine Datenanalyse-Methode. ML verfolgt das Ziel, aus Daten zu lernen und möglichst treffende Vorhersagen zu generieren. Das heißt, ein künstliches System lernt aus Beispielen und kann nach Beendigung der Lernphase das Gelernte verallgemeinern. Es lernt hierbei aber die Beispiele nicht einfach auswendig, sondern es erkennt in den Lerndaten Gesetzmäßigkeiten.

Daher werden im Machine Learning Daten gesammelt und aufbereitet, um sie mithilfe von speziellen Machine-Learning-Algorithmen auf einem oder mehreren Rechnern zu trainieren. Dabei leitet der Algorithmus aus den Daten ein Modell ab, das in der Lage ist, bestimmte Eigenschaften in den Daten zu erkennen. Das Ergebnis eines ML-Algorithmus ist immer ein Machine-Learning Model. Fast alle Modelle, die ML-Algorithmen aus Beispielen erzeugen, sind letztlich statistische Modelle, weshalb Statistik ein Fundament für die Theorie des maschinellen Lernens bildet.

Für den Softwarebereich hat Thomas Mitchell es wie folgt beschrieben: Ein Computerprogramm lernt beim Lösen einer bestimmten Klasse von Aufgaben (T), wenn seine messbare Leistung (P) sich mit der Erfahrung (E) im Lauf der Zeit erhöht [4]. Diese Logik zu finden wird maschinelles Lernen genannt. Ausführlicher wird ML im nachfolgenden Kapitel 2 erläutert.

### 1.2.2 Deep Learning

Deep Learning, kurz als DL oder auch als tiefgehendes Lernen bezeichnet, ist eine Teildisziplin des maschinellen Lernens und wird derzeit am häufigsten im Zusammenhang mit dem Begriff KI verwendet. Die Grundlage von DL sind künstliche neuronale Netze und somit eine spezielle Methode der Informationsverarbeitung. DL wird besonders auf große Datenmengen angewandt, die mithilfe von neuronalen Netzen analysiert werden. Über diese Technologie schafft es das System, Strukturen zu erkennen, Informationen zu sortieren und zu evaluieren. Dabei vollzieht sich ein permanenter Prozess, das Gelernte wird immer wieder mit neuen Inhalten verknüpft und erweitert. Dies führt mit hoher Wahrscheinlichkeit dazu, dass ein richtiges Ergebnis erkannt und ausgegeben wird. DL kommt zum Einsatz, wenn andere ML-Verfahren an ihre Grenzen stoßen.

Auch der Prozess und die benötigten Ressourcen unterscheiden sich im Vergleich zum Machine Learning. Bei ML gibt der Mensch Testdaten mit korrekten Antworten vor, sodass der Algorithmus basierend auf diesen manuell klassifizierten Daten lernt. Beim Deep Learning fällt die Vorgabe der Lösung weg, stattdessen wird die Zuordnung und Klassifizierung der Daten automatisiert. Mehr Informationen zu DL finden Sie in Abschnitt 2.9.

### 1.2.3 Cognitive Computing

Cognitive Computing stellt einen Teilbereich der KI dar, der eine natürliche, möglichst menschliche Interaktion mit Maschinen anstrebt. Hierfür werden bei Cognitive Computing Technologien der KI genutzt, um menschliche Denkprozesse zu simulieren. Dabei geht es darum, auf Basis von Erfahrungen eigene Lösungen und Strategien zu entwickeln.

Mithilfe von KI und Cognitive Computing soll ein System entstehen, das Bilder und Sprache interpretiert und auch noch schlüssig antworten kann. Die wichtigste Voraussetzung für das Cognitive Computing ist demnach die Fähigkeit, aus den gemachten Erfahrungen selbstständig zu lernen. Gleichzeitig soll das System die eigenen Lösungsansätze ständig hinterfragen.

Ein wesentliches Merkmal des Cognitive Computing ist, dass riesige Datenmengen unterschiedlichster Art gespeichert und binnen kürzester Zeit verarbeitet werden müssen. Da die Daten in der Regel in unstrukturierter Form vorliegen, lassen sich herkömmliche relationale arbeitende Systeme nicht für das Cognitive Computing effizient einsetzen. Daher kommen hier Techniken aus dem Big-Data-Umfeld in Verbindung mit Data Science zum Einsatz.

Das bekannteste Beispiel für Cognitive Computing ist wohl der Sieg 2011 von IBM Watson bei Jeopardy. Hierbei war das System in der Lage, Fragen selbstständig und sinnvoll zu beantworten.

### 1.2.4 Big Data und Data Science

Big Data ist ein allgemeiner Begriff, der für die Beschreibung umfangreicher Mengen unstrukturierter und semi-strukturierte Daten verwendet wird. Demzufolge bezeichnet Big Data die immer rasanter wachsenden Datenmengen zum Beispiel aus den sozialen Netzwerken. Die wichtigsten Datenquellen für Big Data sind:

- Mobile Internetnutzung
- Social Media
- Geo-Tracking
- Cloud Computing
- Vitaldaten-Messung
- Media-Streaming

Der Begriff Big Data meint aber nicht nur die Daten selbst, sondern auch deren Analyse und Nutzung. Hierfür wird Big Data auch mit Cloud Computing und Machine Learning in Verbindung gebracht, um die Echtzeit-Analyse von großen Datenmengen zu bewerkstelligen. Da die gesammelten Datenmengen allerdings komplex, schnelllebig und unstrukturiert sind, stellt sich die Aufbereitung und Analyse sehr häufig als schwierig und aufwendig dar. An dieser Stelle kommt dann Data Science zum Einsatz.

Bei Data Science handelt es sich um eine Wissenschaft, die sich mit der Extraktion von Wissen aus großen Datenmengen (auch Big Data) beschäftigt. Mit verschiedenen Methoden aus der Datenanalyse und Visualisierung werden die gesammelten Daten erfasst und die benötigten relevanten Informationen extrahiert. Gegenwärtig werden im Data Science Techniken und Methoden aus der Mathematik, der Statistik, der Stochastik und der Informatik angewendet.

### 1.2.5 Predictive Analytics

Predictive Analytics stellt eine Analysemethode dar, die sowohl neue als auch historische Daten zur Vorhersage von Aktivitäten, Verfahren und Trends verwendet. Man nutzt hier Machine Learning um Vorhersagemodelle zu erstellen, die einen numerischen Wert für die Wahrscheinlichkeit des Eintretens eines bestimmten Ereignisses berechnen. Bei Predictive Analytics kommen folgende Methoden zum Einsatz:

- **Logistische Regression:** Eine Analysemethode, die zur Vorhersage eines Datenwertes auf Grundlage früherer Beobachtungen eines Datensatzes verwendet wird.
- **Zeitreihenanalyse:** Eine Darstellung von Datenpunkten in aufeinanderfolgenden Zeitintervallen.
- **Entscheidungsbaum:** Ein Diagramm, das mithilfe einer Verzweigungsmethode jedes mögliche Ergebnis einer Entscheidung darstellt.

Durch den Einsatz von ML hat Predictive Analytics in den letzten Jahren große Fortschritte gemacht und viel neue Aufmerksamkeit erfahren.

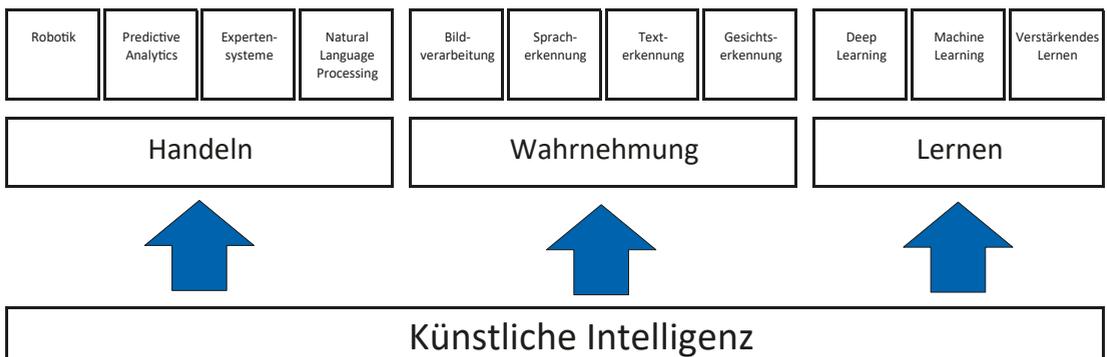
### 1.2.6 Natural Language Processing

Natural Language Processing, kurz NLP, beschreibt Techniken und Methoden zur maschinellen Verarbeitung natürlicher Sprache. Das heißt, NLP versucht, natürliche Sprache zu erfassen und mithilfe von Regeln und maschinengestütztem Lernen zu verarbeiten. NLP kommt als Teilbereich in folgenden Anwendungen zum Einsatz:

- Spracherkennung
- Segmentierung zuvor erfasster Sprache in einzelne Wörter und Sätze
- Erkennen der Grundformen der Wörter und Erfassung grammatischer Informationen
- Extraktion der Bedeutung von Sätzen und Satzteilen

Ein praktisches Beispiel für den Einsatz von NLP finden Sie in Kapitel 8.

Bild 1.2 zeigt noch einmal alle Teilbereiche der KI in Bezug auf Handeln, Wahrnehmung und Lernen.



**Bild 1.2** Die Teilbereiche Handeln, Wahrnehmen und Lernen in der KI

## ■ 1.3 KI-Service-Plattformen

Alle großen Internetkonzerne bieten inzwischen eine Zusammenstellung verschiedener APIs (Application Programming Interface), SDKs (Software Development Kit) und Services (Dienste) an, um Ihnen als Entwickler dabei zu helfen, ihre Programme intelligenter und benutzerfreundlicher zu machen.

Über diese KI-Plattformen können Sie auf ausgereifte und schon fertige Services für Lösungen auf Basis von Machine-Learning-Modellen oder Services für Sprach- und Text-Erkennung zurückgreifen. Neben den verschiedenen KI-Services und -Tools bieten diese Plattformen durch die Nutzung in der Cloud auch eine gute Unterstützung bei der Infrastruktur, der Sicherheit, der Verfügbarkeit und natürlich auch den Vorteil der zur Verfügung stehenden Rechenleistung.

Neben diesen KI-Services veröffentlichen die Anbieter wie Google, Amazon, Microsoft und IBM auch den entsprechenden Code ihrer Open Source ML- und DL-Frameworks. So können Sie als Entwickler an der direkten Weiterentwicklung der Frameworks teilnehmen und sich in der jeweiligen Community austauschen.

### 1.3.1 Amazon

Amazon betreibt mit seinem Web Service (AWS) den größten Cloud Computing Dienst weltweit. Neben den bekannten Infrastruktur-Diensten bietet Amazon auch Services für künstliche Intelligenz an.

Mit Deep Scalable Sparse Tensor Network Engine (DSSTNE) hat Amazon seine Programm-bibliothek für maschinelles Lernen als Open Source zu Verfügung gestellt. Der Schwerpunkt von DSSTNE liegt auf Verfahren, die bei der Entwicklung von ML-Modellen mit nur wenigen eigenen Trainingsdaten auskommen, da das erstellte ML-Modell über einen Cloud Service verteilt trainiert werden kann. Die weiteren KI Services von Amazon bieten die folgenden Möglichkeiten.

#### **Amazon SageMaker**

Der Amazon SageMaker ist ein von Amazon vollständig verwalteter Service, der jedem Entwickler die Möglichkeit bietet, schnell Modelle für Machine Learning zu erstellen und diese Modelle zu trainieren, um sie dann für eine Anwendung bereitzustellen. Über dieses Tool wird der gesamte ML-Zyklus von der Datenaufbereitung bis zur Bereitstellung der ML-Modelle abgedeckt.

#### **Amazon Lex**

Amazon Lex ist ein Service zur Erstellung einer Schnittstelle für Sprache und Text. Amazon Lex bietet automatische Spracherkennung zur Umwandlung von Sprache in Text und eine Erkennung der Textabsicht, um eine realistische Gesprächsinteraktion in der Anwendung zu gewährleisten.

Amazon Lex bildet den Kern von Amazon Alexa und stellt somit jedem Entwickler diese Technologie zur Verfügung. Somit soll es möglich sein, schnell und einfach komplexe Chatbots zu entwickeln.

### **Amazon Translate**

Amazon Translate ist ein neuronaler, maschineller Übersetzungsservice, der kostengünstig Übersetzungen liefert. Hierbei bedeutet neuronale maschinelle Übersetzung, dass Amazon Deep-Learning-Modelle verwendet, um im Vergleich zu herkömmlichen regelbasierten Übersetzungsalgorithmen eine genauere und natürlicher klingende Übersetzung zu liefern.

### **Amazon Polly**

Amazon Polly ist ein Service, der Text in realistische Sprachausgabe verwandelt. Hier kommt über den Text-to-Speech-Service Deep Learning zum Einsatz, um natürlich klingende menschliche Sprache zu synthetisieren. Somit soll es möglich sein, neue sprachfähige Produkte zu entwickeln.

### **Amazon Rekognition**

Hinter Amazon Rekognition steht ein hochgradig skalierbares Deep-Learning-Modell, das es ermöglicht Objekte, Menschen, Text, Szenen und Aktivitäten in Bildern und Videos zu identifizieren. Der Service bietet weiterhin auch eine sehr genaue Gesichtsanalyse- und Gesichtsfunktionen an, mit denen Sie Gesichter analysieren und vergleichen können. Sie benötigen für diesen Service keinerlei Machine-Learning-Kenntnisse.

### **Amazon RoboMaker**

Mit dem Amazon RoboMaker steht eine umfassende Cloud-Lösung für die Entwicklung von Robotersystemen zur Verfügung. RoboMaker ermöglicht das Simulieren, Testen und das sichere Bereitstellen von Roboteranwendungen im großen Umfang.

### **Apache MXNet**

Amazon bietet einige optimierte Services für das Deep-Learning-Framework Apache MXNet an. Bei MXNet handelt es sich um ein schnelles und skalierbares Schulungs- und Inferenz-Framework mit einer kompakten und entwicklerfreundlichen API für Machine Learning. Apache MXNet bietet des Weiteren interessante Einblicke in den Aufbau tiefgreifender Deep-Learning-Systeme für Entwickler.

## **1.3.2 Google**

Google gehört heute zu den führenden Unternehmen im Bereich KI. Neben der Forschung betreibt Google auch eigene KI-Systeme auf der Google Cloud Plattform, die Ihnen als Entwickler zur Verfügung stehen. Die bekannteste Programmibliothek für ML von Google ist TensorFlow.

## **TensorFlow**

TensorFlow ist Googles plattformunabhängiges Open-Source-Framework für maschinelles Lernen. Der Schwerpunkt des Systems liegt in der Verarbeitung von Sprache und Bildern. In der Forschung und im Produktivbetrieb wird das Framework derzeit in verschiedenen Google-Produkten eingesetzt. Hierzu zählen die Google-Spracherkennung, Gmail, Google Fotos, Google Maps und die Google-Suche. Mehr Informationen zu TensorFlow finden Sie in Abschnitt 7.2 „TensorFlow“.

## **Google Vision API**

Die Google Vision API bietet über eine REST API-Schnittstelle leistungsstarke vorabtrainierte Modelle für Machine Learning. Sie können über die Schnittstelle Bilder mit Labels versehen und diese Bilder dann in kurzer Zeit Millionen von vordefinierten Kategorien zuordnen. Außerdem ist es möglich, Objekte und Gesichter zu erkennen sowie gedruckten und handgeschriebenen Text zu lesen.

## **Cloud AI Platform**

Die Cloud AI Platform ermöglicht es Entwicklern, den vollständigen Zyklus der Machine-Learning-Entwicklung in einem Projekt zu managen. Sie unterstützt mit einem integrierten Toolset alle Prozessketten von der Erstellung der Datenbasis bis zu Ausführung und Bereitstellung des ML-Projekts.

## **AutoML Video Intelligence**

AutoML Video Intelligence stellt eine grafische Benutzeroberfläche zur Verfügung, mit der Sie eigene ML-Modelle zur Klassifizierung und Nachverfolgung von Objekten in Videos trainieren können.

## **Translation API Basic**

Die Google Translation API übersetzt Texte aus Websites und Anwendungen in mehr als 100 Sprachen. Translation API verwendet hierfür vortrainierte neuronale Machine-Learning-Übersetzungen und liefert damit schnelle, dynamische Ergebnisse.

## **AutoML Natural Language**

AutoML Natural Language bietet eine Benutzeroberfläche, um eigene ML-Modelle zum Klassifizieren, Extrahieren und Erkennen von Stimmungen trainieren zu können. Sie können Ihre Trainingsdaten direkt über die Oberfläche hochladen und anschließend einfach Ihr erstelltes ML-Modell testen.

## **Vortrainierte Modelle**

Für viele gängige Anwendungsfälle im Bereich Sprachen- und Textanalyse sowie in der Bildverarbeitung stehen Ihnen von Google schon vortrainierte ML-Modelle als API-Schnittstelle für die Nutzung in eigenen Anwendungen zur Verfügung. Da Google laufend weitere Fortschritte in der KI-Forschung erzielt, haben Sie immer Zugriff auf die neusten vollständig trainierten ML-Modelle.

### 1.3.3 Microsoft Cognitive Services

Neben dem bekannten Open-Source-Framework ML.NET bietet das Softwareunternehmen Microsoft weitere APIs, SDKs und Dienste rund um das Thema KI an. Diese werden als Cognitive Services bezeichnet und sind über die Azure-Plattform erreichbar.

#### **Gesichtserkennungs-API**

Die Gesichtserkennungs-API nutzt Algorithmen, um menschliche Gesichter auf Bildern zu erkennen. Diese API bietet Erweiterungen wie Gesichtserkennung, Gesichtsabgleich und Gesichtsgruppierung, um Gesichter basierend auf ihrer Ähnlichkeit in Gruppen einzuordnen.

#### **Video API**

Die Video API erlaubt, das Erkennen von gesprochenen Wörtern, geschriebenem Text, Gesichtern, Emotionen, Marken oder Szenen automatisch aus Video- oder Audiodateien zu extrahieren. Über eine einfache Benutzeroberfläche kann die API-Schnittstelle konfiguriert werden.

#### **Speech-Service**

Der Speech-Service dient zur Vereinheitlichung von Spracherkennung, Sprachsynthese und Sprachübersetzung. Bis Anfang 2020 war dieser Bereich noch in einzelne API-Schnittstellen aufgeteilt. Auch Sprachassistenten können mit diesem umfangreichen Service entwickelt werden.

#### **Freihanderkennungs-API**

Diese KI-API unterstützt Sie bei der Erkennung digitaler Freihandinhalte wie handschriftlicher Texte, Formen und Layouts geschriebener Dokumente. Es handelt sich hierbei aber nicht um eine optische Zeichenerkennung wie bei OCR (Optical Character Recognition), sondern um eine Analyse der Daten anhand der Bewegung über Eingabertools wie digitaler Stifte oder Fingerzeichen. Der erkannte Inhalt wird dann über die API-Schnittstelle als JSON-Antwort zurückgeliefert.

#### **Cognitive Toolkit**

Auch das Cognitive Toolkit ist ein kostenloses Open-Source-Deep-Learning-Framework, mit dem künstliche neuronale Netze trainiert werden können. Das Toolkit beschreibt hierbei neuronale Netze als eine Reihe von Rechenschritten über einen gerichteten Graphen. Sie können mit dem Cognitive Toolkit beliebige ML-Modelltypen einfach realisieren und diese auch kombinieren. Das Toolkit ist mit vielen verschiedenen Programmiersprachen nutzbar und ermöglicht auch einen zuverlässigen Betrieb mit großen Datenmengen.

### 1.3.4 IBM

Auch IBM bietet über seine Watson Data Platform verschiedene Dienste für KI-Aktivitäten an. Diese werden als Watson Developer Cloud Services zur Verfügung gestellt.

#### **Watson Natural Language Classifier**

Der Natural Language Classifier unterstützt Sie, die Sprache kurzer Texte zu verstehen. Ein Klassifikationsmerkmal, der Classifier, lernt anhand Ihrer Beispieldaten und kann anschließend Informationen zu Texten zurückgeben, anhand derer das ML-Modell nicht trainiert wurde.

#### **Watson Natural Language Understanding**

Der Natural Language Understanding-Service dient zur Analyse von sprachlichen Merkmalen in Texteingaben. Es wird versucht, Kategorien, Konzepte oder auch Emotionen aus dem Text zu extrahieren.

#### **Watson Visual Recognition**

Auch bei IBMs Visual Recognition werden Deep-Learning-Algorithmen verwendet, um in Bildern Objekte und Szenen erkennen zu können.

#### **Vortrainierte Modelle**

Wie Google, stellt auch IBM vortrainierte Modelle für Machine Learning zur Verfügung. Hier haben die ML-Modelle aber einen eindeutigen Bezug auf entsprechende Anwendungsfälle. So gibt es ML-Modelle für die Fertigung, für Versicherungen, den Einzelhandel und den Schulungsbereich.

#### **Watson Assistant**

Bei IBMs Watson Assistant handelt es sich um eine sogenannte Konversations-KI-Plattform für die Erstellung von virtuellen Assistenten (Chatbots). Hier ist der Assistent ein kognitiver Bot, den Sie an Ihre Anforderungen anpassen und bereitstellen können, um Ihren Anwendern kontextbezogene Unterstützung anzubieten. Watson Assistant erstellt dynamisch ein Modell für Machine Learning, das auf Ihre definierten aber auch auf ähnliche Benutzeranfragen abgestimmt ist.

Wie Sie sehen, bieten die Konzerne schon eine beachtliche Anzahl von nützlichen Schnittstellen und KI-Diensten an, die auch entsprechend weiterentwickelt werden, und es kommen ständig neue APIs oder KI-Services hinzu.

Durch das Zur-Verfügung-stellen einer entsprechenden API oder auch eines Service ist es für Sie sehr viel einfacher, auf schon vorhandene Funktionen und Modelle zurückzugreifen. Profitieren Sie hier einfach von den Weiterentwicklungen und dem Austausch in der jeweiligen Community, um Ihre Programmentwicklung noch effektiver zu gestalten.

Dieses Buch möchte Sie darin unterstützen, selber Modelle für entsprechende Frameworks zu erstellen oder schon vorhandene zu modifizieren. Nutzen Sie hierfür aber auch die Vorteile der großen Lernplattformen von Microsoft, Google und Co.

## ■ 1.4 Künstliche neuronale Netze

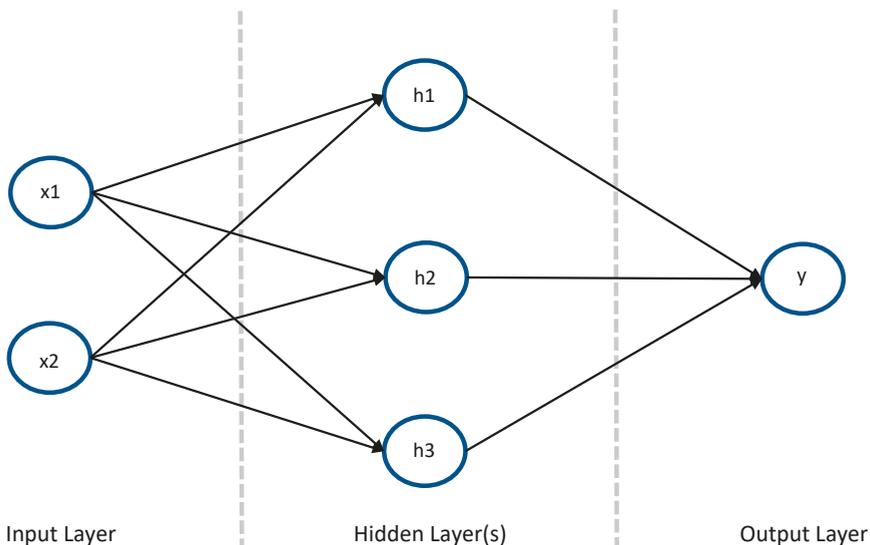
Unter den lernenden Algorithmen sind die künstlichen neuronalen Netze die Schlüsseltechnologie für Machine und Deep Learning. Um aber die Möglichkeit von ML und DL auszukosten und richtig zu nutzen, ist es hilfreich, wenn man weiß, wie neuronale Netze tatsächlich funktionieren.

Die Zusammenhänge im Detail versteht man am besten, wenn man ein eigenes neuronales Netz programmiert. Das so erworbene Wissen lässt sich dann bei der Verwendung eines KI-Services oder eines entsprechenden Frameworks sicher und effektiv nutzen.

### 1.4.1 Funktionsweise

Künstliche neuronale Netze (engl. *artificial neural network* - ANN), kurz als KNN bezeichnet, bestehen aus Knoten, die Sie als Analogie zu menschlichen Nervenzellen sehen können. Diese Knoten stellen die einzelnen Neuronen dar, die auch gerne als Units oder Einheiten bezeichnet werden.

Sie dienen dazu, Informationen aus der Umwelt oder von anderen Neuronen aufzunehmen und diese wiederum an andere Neuronen in modifizierter Form weiterzuleiten. Diese Neuronen sind in Schichten, den sogenannten Layers angeordnet, in denen sich mehrere Neuronen befinden. Jedes Neuron einer Schicht (Layer) ist mit allen Neuronen der nächsten Schicht verbunden. Somit besteht ein KNN aus Neuronen mit gerichteten und gewichteten Verbindungen. Bild 1.3 zeigt die schematische Darstellung eines neuronalen Netzes.



**Bild 1.3** Schematische Darstellung eines künstlichen neuronalen Netzes (KNN)