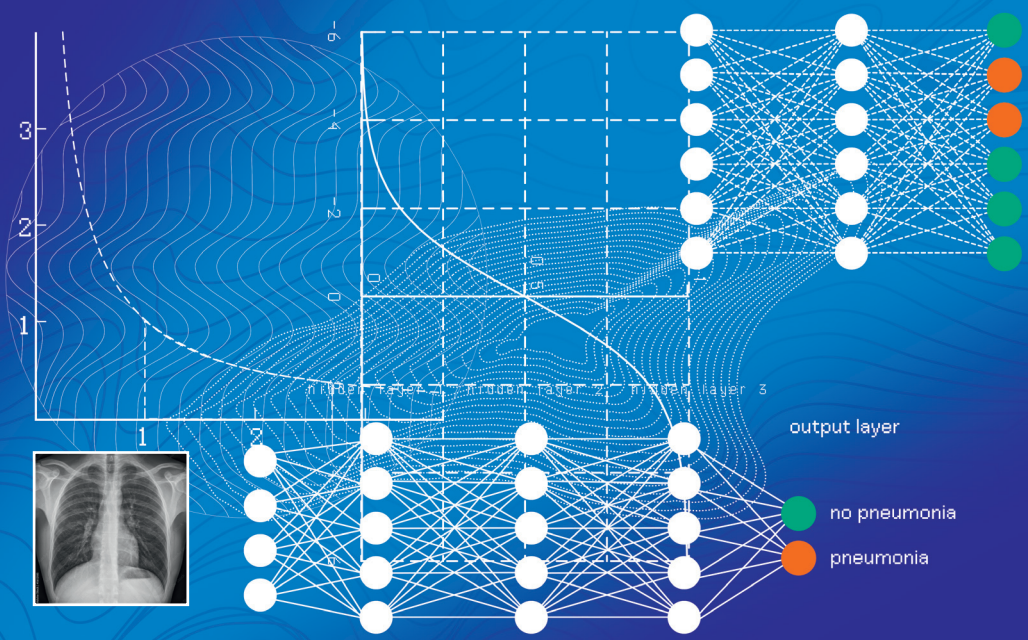
Tim Wiegand Laura Velezmoro

Künstliche Intelligenz in der Medizin

Anwendungen, Algorithmen und Programmierung

Für Studium, Forschung, Klinik und Wirtschaft



LESEPROBE

ELSEVIER
Urban & Fischer

Fünf Meinungen zum Einsatz von KI in der Medizin

Dieses Buch soll die Funktionsweise der KI und deren Anwendungen in der Medizin neutral und sachlich vermitteln. Dennoch haben wir Meinungen, die den Einsatz von KI im Gesundheitswesen betreffen und die Zusammenstellung der Buchkapitel und Beispiele im Buch beeinflusst haben. Wir möchten unsere Meinungen und Biases daher vorab offenlegen und dich gleichzeitig ermutigen, dir eigene Meinungen zu bilden und sie mit Kraft und Ehrgeiz zu vertreten. Das Feld der KI in der Medizin ist ein relativ junges, der Fortschritt allerdings überfällig. Es ist daher höchste Zeit, dieses Feld zu formen und am Fortschritt aktiv mitzuwirken.

Meinung 1: Datenschutz ist ein wichtiges Gut, aber nicht das einzige wichtige Gut in der Gesundheitsversorgung. Daher gilt es abzuwägen zwischen Risiken wie einer Verletzung des Datenschutzes und Chancen wie einer besseren Gesundheitsversorgung durch KI.

Meinung 2: KI trotz aller Fortschritte nicht zu nutzen ist nicht ethisch – insbesondere angesichts des Fachkräftemangels, des demografischen Wandels mit der Überalterung der Gesellschaft bei einer immer geringeren Zahl Erwerbstätiger.

Meinung 3: Die Ergebnisse von KI-Systemen sollten erklärbar sein. Diese sog. Explainability ist entscheidend, um das Vertrauen der Behandelnden und Behandelten in den Einsatz von KI in der Medizin zu schaffen.

Meinung 4: Hilfreiche diagnostische KI-Systeme müssen auch die seltenen und schwer zu erkennenden Erkrankungen diagnostizieren können, nicht nur die häufigen und einfach zu diagnostizierenden.

Meinung 5: Ein Grundverständnis der Behandelnden zum Thema KI ist notwendig, so wie bereits Behandelnde vieler Fachrichtungen über ein Grundverständnis zu den technischen Hintergründen ihrer medizinischen Instrumente verfügen.

Benutzerhinweise

MERKE
In den Merke-Kästen finden Sie für das Verständnis besonders wichtige Zusammenhänge.

CAVE
In den Cave-Kästen wird sowohl auf typische Fallstricke hingewiesen als auch auf nicht eindeutig verwendete Begriffe, die einer Klarstellung bedürfen.

AUFGABE
In Kapitel 5 werden bei der Einführung in die Python-Programmierung Aufgaben gestellt, mit dem sich der Inhalt des Kapitels vertiefen lässt.

Zusammenfassung

In den Zusammenfassungen sind noch einmal die wichtigsten Inhalte der vorausgegangenen Abschnitte aufgelistet.

Hinweis zu den Patientenfällen

Die Namen der Patientinnen und Patienten, die Beschreibungen ihrer Erkrankungen, die Orte und Situationen sind alle realitätsnah, aber frei erfunden. Jegliche Ähnlichkeiten mit echten Personen und Geschehnissen sowie Orten und Institutionen sind daher rein zufällig und nicht beabsichtigt.

Github-Repository: Programmierbeispiele und Quiz

FPO

https://github.com/timwgnd/Lehrbuch-Kuenstliche-Intelligenz-in-der-Medizin

Zu diesem Buch gibt es ein Online-Verzeichnis auf Github. Dort finden Sie alle im Buch aufgeführten Programmierbeispiele sowie die verwenden Datensätze.

Außerdem steht dort ein Quiz mit 40 Fragen zu den Buchinhalten zur Verfügung, mit dem Sie Ihr Wissen testen können.

Fehler gefunden?

FPO

https://else4.de/978-3-437-XXXXX-X #Kurzlink einfügen => kopieren aus Exceldatei: K:\QR-Codes_Erratum#

An unsere Inhalte haben wir sehr hohe Ansprüche. Trotz aller Sorgfalt kann es jedoch passieren, dass sich ein Fehler einschleicht oder fachlich-inhaltliche Aktualisierungen notwendig geworden sind.

Sobald ein relevanter Fehler entdeckt wird, stellen wir eine Korrektur zur Verfügung. Mit diesem QR-Code gelingt der schnelle Zugriff.

Wir sind dankbar für jeden Hinweis, der uns hilft, dieses Werk zu verbessern. Bitte richten Sie Ihre Anregungen, Lob und Kritik an folgende E-Mail-Adresse: kundendienst@elsevier.com

Inhaltsverzeichnis

KAPITEL

1 Geschichte der künstlichen Intelligenz

Die Geschichte der künstlichen Intelligenz (KI) ist eine über mehrere Jahrzehnte kontinuierliche Entwicklung mit Fortschritten und Rückschlägen. Grundlagen und einige Meilensteine dieser Geschichte werden in diesem Kapitel dargestellt.

1.1 Algorithmen

KI beschreibt die Fähigkeit von Computersystemen, Aufgaben auszuführen, die normalerweise menschliche Intelligenz erfordern. Hierzu werden sog. Algorithmen genutzt, also Sammlungen von Handlungsanweisungen, um Informationen bzw. Daten so zu verarbeiten, dass ein Problem intelligent gelöst werden kann.

Der Begriff Algorithmus geht dabei auf den persischen Rechenmeister und Astronomen Muhammad ibn Musa Al-Chwarizmi (783–847 n. Chr.) zurück. In seinem Lehrbuch beschrieb er die Gesamtheit der Regeln zur formalen Lösung von Gleichungen. Diese Handlungsvorschriften bildeten lange Zeit die Grundlage für die Gleichungslehre, sodass sich für Handlungsvorschriften der Begriff „Algorithmus“ aus dem Namen von „Al-Chwarizmi“ entwickelte.

Heute sind es Computerprogramme, die immer komplexere Algorithmen in immer kürzerer Zeit ausführen. Der kontinuierliche Fortschritt in der Rechenleistung war dabei ein entscheidender Schritt in der Entwicklung von KI-Systemen und eröffnete neue Möglichkeiten für die Lösung komplexer Probleme.

1.2 Geburtsstunde der künstlichen Intelligenz

Der Begriff „künstliche Intelligenz“ wurde bereits Mitte des 20. Jahrhunderts geprägt. Seitdem wurden unzählige mathematische und computerwissenschaftliche Durchbrüche erzielt, die dazu geführt haben, dass die KI heute in vielen Bereichen des Lebens präsent ist.

1936 – Turing-Maschine

Der britische Mathematiker Alan Turing entwickelte in den 1930er-Jahren das theoretische Konzept der sog. Turing-Maschine. Hierbei handelt es sich um ein Modell, das beschreibt, wie ein Rechner eine formalisierte Aufgabe löst und welche Anforderungen an den Algorithmus gestellt werden müssen, damit der Rechner die Aufgabe verstehen und ausführen kann. Dies war ein bahnbrechender Schritt, der die Idee aufwarf, dass Maschinen Aufgaben ausführen könnten, die zuvor ausschließlich menschlicher Intelligenz vorbehalten waren.

Die Turing-Maschine verfügt nur über begrenzte Fähigkeiten. Sie kann Symbole oder Zeichen schrittweise verändern, die nach bestimmten Regeln auf ein Speicherband geschrieben und von dort abgelesen werden.

Hierzu besteht die Maschine aus wenigen einfachen Komponenten (➢ Abb. 1.1):

- Ein „unendlich" langes **Speicherband** mit Symbolen oder Zeichen, wie z. B. Zahlen, dient als Datenspeicher, der vom Schreib-Lese-Kopf modifiziert werden kann.
- Ein **Schreib-Lese-Kopf** kann sich in beide Richtungen über das Speicherband bewegen, um Zeichen zu lesen oder zu schreiben.
- **Verarbeitungsregeln** geben an, wie sich die Maschine in Abhängigkeit vom aktuellen Zustand und dem gelesenen Symbol verhalten soll.

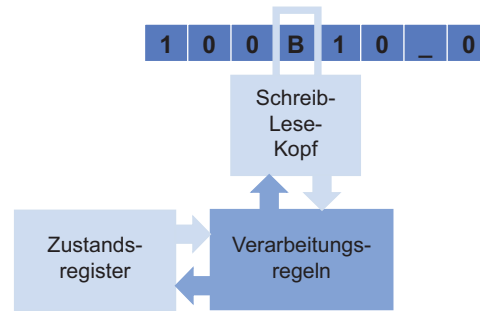


Abb. 1.1 Bestandteile einer Turing-Maschine. Basierend auf dem Zeichen unter dem Schreib-Lese-Kopf und dem aktuellen Zustand der Maschine wählt diese die anzuwendende Verarbeitungsregel bzw. Aktion aus und führt diese durch. [P1522/P1523]

Die Turing-Maschine wendet – je nach dem aktuellen Zustand der Maschine und dem gelesenen Zeichen auf dem Speicherband – vordefinierte Verarbeitungsregeln an. Diese Regeln entsprechen also dem Algorithmus, nach dem die Turing-Maschine arbeitet. Für jede mögliche Kombination von ausgelesenen Daten bzw. Zeichen auf dem Speicherband und Zuständen der Maschine legen diese Regeln die entsprechenden Aktionen der Maschine fest. Diese Aktionen können u. a. den Übergang zu einem neuen Zustand der Maschine, das Überschreiben von Daten in der aktuellen Zelle des Speicherbands oder die Bewegung des Schreib-Lese-Kopfes zur nächsten (rechten oder linken) Zelle auf dem Speicherband umfassen.

Eine Aufgabe könnte es beispielsweise sein, alle As und Bs in einer auf dem Speicherband stehenden Zeichenfolge „1A0B10" durch Nullen zu ersetzen. Zu Beginn steht der Schreib-Lese-Kopf über dem ersten Zeichen der Zeichenfolge (also der 1). Das Programm wird fortgesetzt, bis der Kopf das Ende der Zeichenfolge erreicht hat. Zur Erfüllung des Ziels gibt es die folgenden Handlungsanweisungen:

- Wenn das aktuelle Zeichen 0 ist, dann den Kopf nach rechts bewegen.
- Wenn das aktuelle Zeichen 1 ist, dann den Kopf nach rechts bewegen.
- Wenn das aktuelle Zeichen A ist, dann das Zeichen zu 0 ändern und den Kopf nach links bewegen.
- Wenn das aktuelle Zeichen B ist, dann das Zeichen zu 0 ändern und den Kopf nach links bewegen.
- Wenn das aktuelle Zeichen ein Leerzeichen ist bzw. die Sequenz endet, das Programm beenden.

Die Verarbeitungsschritte sind dann (die aktuelle Position des Schreib-Lese-Kopfes ist fett hinterlegt):

1. Zeichen (1): **1**A0B10 → 1A0B10
2. Zeichen (A): 1**A**0B10 → 100B10
3. Zeichen (1): **1**00B10 → 100B10
4. Zeichen (0): 1**0**0B10 → 100B10
5. Zeichen (0): 10**0**B10 → 100B10
6. Zeichen (B): 100**B**10 → 100010
7. Zeichen (0): 10**0**010 → 100010
8. Zeichen (0): 100**0**10 → 100010
9. Zeichen (1): 1000**1**0 → 100010
10. Zeichen (0): 10001**0** → 100010
11. Zeichen (Trennzeichen): Ende

Es können jedoch auch deutlich komplexere Anweisungen definiert werden, sodass sich auch Operationen, die heute von modernen Computern ausgeführt werden, mit der Turing-Maschine modellieren ließen. Mit diesem einfachen Modell gelang es Turing, den Begriff des Algorithmus, wie er in der Mathematik und Informatik heute genutzt wird, fassbar zu machen.

1950 – Turing-Test

In seinem Aufsatz „Computing Machinery and Intelligence“ [1] aus dem Jahr 1950 warf Turing die Frage auf, ob Maschinen ein Maß an Intelligenz erreichen können, das dem des Menschen ähnelt. In diesem Zusammenhang formulierte er den Turing-Test: Eine verblindete Testperson kommuniziert sowohl mit einem Menschen als auch mit dem zu testenden KI-System (➢ Abb. 1.2). Kann sie dabei nicht zuverlässig feststellen, welcher der beiden Gesprächspartner der Mensch ist und welcher das KI-System, hat das System den Test bestanden bzw. gilt als intelligent. Es kann dann gemäß Turing davon ausgegangen werden, dass das System in der Lage ist, menschenähnliche Unterhaltungen zu führen. Dieser Test diente als frühes theoretisches Konstrukt zur Beschreibung des Konzepts der Intelligenz sowie zur Prüfung, ob Maschinen diese imitieren bzw. erreichen können.

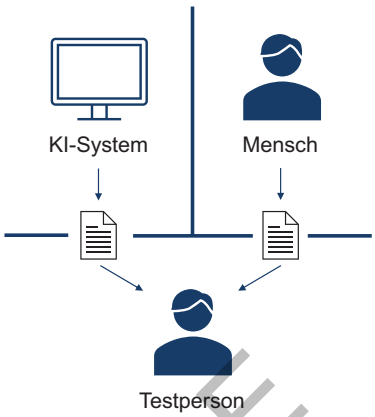


Abb. 1.2 Turing-Test. Der Turing-Test kann angewendet werden, um festzustellen, ob ein KI-System menschenähnliche Intelligenz aufweist. Dazu soll eine verblindete Testperson unterscheiden, ob sie mit dem zu testenden KI-System oder einem Menschen kommuniziert. Gelingt die Unterscheidung nicht, kann angenommen werden, dass das System menschenähnliche Intelligenz aufweist. [P1522/P1523]

1956 – Dartmouth-Konferenz und „künstliche Intelligenz“

Im Jahr 1956 fand die sog. Dartmouth-Konferenz statt („Dartmouth Summer Research Project on Artificial Intelligence“), die oft als Geburtsstunde der KI angesehen wird. Führende Forscher wie John McCarthy, Marvin Minsky, Nathaniel Rochester und Claude Shannon diskutierten über die Möglichkeiten der künstlichen Nachahmung menschlicher Intelligenz und prägten dabei den Begriff „künstliche Intelligenz“.

1958 – „Machine Learning“

Arthur Samuel, ein US-amerikanischer Elektroingenieur und Informatiker, verwendete wenig später erstmals den Begriff „Machine Learning“ (maschinelles Lernen): Computer sollten in der Lage sein, zu lernen, ohne explizit darauf programmiert worden zu sein. Sie erhalten also nicht eine explizite Vorgabe, wie ein Problem gelöst werden soll, sondern lernen durch Wiederholung und aus ihren Fehlern, ähnlich wie Menschen auch. Samuel entwickelte mehrere Programme, die durch maschinelles Lernen Brettspiele wie z. B. Dame spielen konnten.

Machine Learning wird heute als Teilbereich der KI aufgefasst (vgl. ➢ Kap. 2) und findet auch in der Medizin breite Anwendung, um Computer z. B. darauf zu trainieren, Diagnosen zu stellen oder Therapieansätze zu entwickeln.

1958 – Perzeptron-Algorithmus

Der Psychologe Frank Rosenblatt stellte 1958 den Perzeptron-Algorithmus vor, der das Fundament für künstliche neuronale Netze legte, der heute wohl wichtigsten Gruppe an KI-Algorithmen. In seiner grundlegendsten Form besteht das Perzeptron aus einem einzelnen künstlichen Neuron. Ähnlich wie biologische Neurone bzw. Nervenzellen erhält das Perzeptron mehrere Eingaben (Inputs), die in eine Ausgabe (Output) umgewandelt werden (➢ Abb. 1.3).

Zur Berechnung der Ausgabe ($\hat{y}$) werden die Eingaben (x_1 und x_2) mit sog. Gewichten (w_1 bzw. w_2) multipliziert und das Ergebnis summiert (Σ). Die Summe wird anschließend in eine sog. Aktivierungsfunktion eingesetzt, deren Ergebnis schließlich der Ausgabe des Perzeptrons entspricht. Das klassische von Rosenblatt beschriebene Perzeptron nutzte als Aktivierungsfunktion eine sog. Heaviside-Funktion H. Sie ist eine stufenförmige Funktion, die für jede beliebige negative Eingabe die Zahl 0 und andernfalls den Wert 1 ausgibt (➢ Abb. 1.4). Die Formel zur Berechnung der Ausgabe $\hat{y}$ des Perzeptrons lautet:

$$\hat{y} = H\left(x_1 \bullet w_1 + x_2 \bullet w_2\right)$$

Die Eingaben wurden also in binäre Ausgaben (0 oder 1) umgewandelt und auf diese Weise eine binäre Klassifikation erzielt. Künstliche neuronale Netze bestehen aus einer großen Zahl künstlicher Neuronen und nutzen viele verschiedene Aktivierungsfunktionen, sodass sie heute deutlich komplexere Zusammenhänge modellieren können.

Rosenblatt realisierte das von ihm entwickelte Konzept des Perzeptrons 1960 am Cornell Aeronautical Laboratory in Buffalo mit dem sog. Perceptron Mark I, einer 2 Meter hohen und 3,5 Meter langen Rechenmaschine, die Bilder klassifizieren sollte. Zur Umsetzung nutzte er verschiedene elektrische Bauteile wie z. B. Fotozellen, Relais und Transistorverstärker, um die Funktion eines einfachen neuronalen Netzes mit einer Neuronenschicht in der Maschine zu implementieren. Mark I war also keine rein digitale Implementierung eines Algorithmus, wie wir es heute von neuronalen Netzen kennen, sondern eine elektromechanische Maschine.

Jedoch bot das Konzept des Perzeptrons auch die Möglichkeit, komplexere Netze aus mehreren Schichten zu entwickeln. Diese werden auch als sog. **Multi-Layer-Perceptron (MLP)** bezeichnet und stellen eine einfache Form eines neuronalen Netzes dar (➢ Abb. 1.5).

Obwohl sein Modell bestenfalls sehr einfache Muster erkennen konnte, war Rosenblatt davon überzeugt, dass zukünftige Perzeptrone in der Lage sein würden, Personen zu erkennen, ihre Namen zu erlernen, und gesprochene oder gedruckte Sprache zu entschlüsseln. Die KI-Pioniere Marvin Minsky und Seymour Papert waren jedoch anderer Meinung (➢ Abb. 1.6). Obwohl sich das Konzept von Rosenblatt später durchsetzte, wog ihre Kritik, die sie 1969 im Buch „Perceptrons“ [2] veröffentlichten, so schwer, dass manche Minsky und

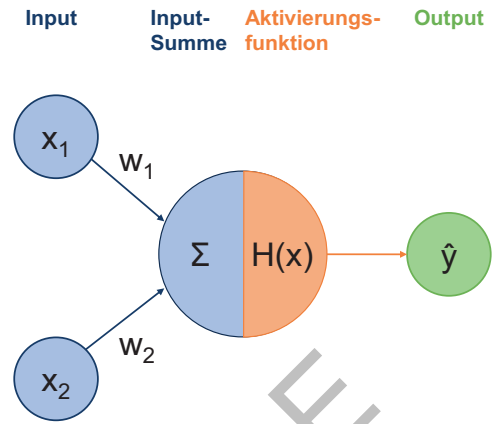


Abb. 1.3 Aufbau eines einfachen, aus einem einzelnen künstlichen Neuron bestehenden Perzeptrons. Es erhält Eingabedaten (x_1, x_2), die mit Gewichten (w_1, w_2) multipliziert und anschließend zu einer Input-Summe addiert werden (Σ). Das Neuron wendet die sog. Heaviside-Funktion H als Aktivierungsfunktion auf die Input-Summe an, um eine Ausgabe ($\hat{y}$) zu erzeugen. [P1522/P1523]

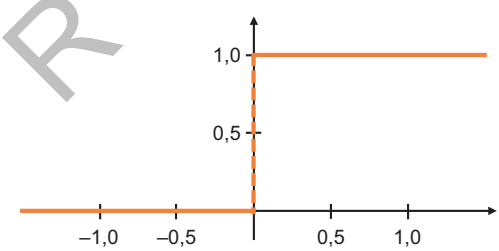


Abb. 1.4 Die Heaviside-Funktion H gibt den Wert 0 für negative Argumente ($x < 0$) und den Wert 1 für nicht-negative Argumente ($x \geq 0$) aus. [P1522/P1523]

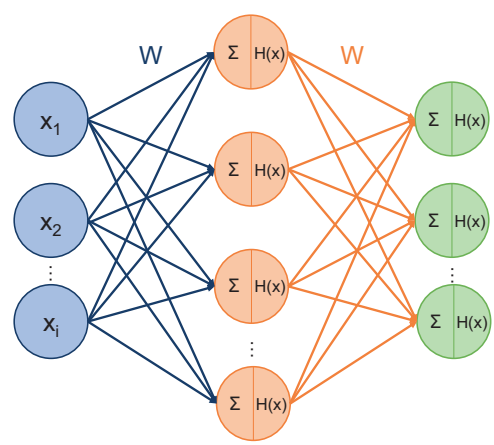


Abb. 1.5 Multi-Layer-Perceptron. Mehrere Perzeptrone (➢ Abb. 1.3) können in Schichten sortiert und miteinander verknüpft werden. Dadurch entsteht Netz aus Perzeptronen bzw. ein neuronales Netz. [P1522/P1523]

Papert sogar für den Beginn des später einsetzenden „KI-Winters“ verantwortlich machten, während dem neuronale Netze und KI für mehr als ein Jahrzehnt lang wenig Aufmerksamkeit erhielten (➢ Kap. 1.3).

1965 – Dendral-Projekt

Das Dendral-Projekt an der Stanford University bzw. der daraus hervorgehende **Dendritic Algorithm** war der Beginn der Nutzung von KI in den Naturwissenschaften. Es erlaubte die Aufklärung molekularer Strukturen organischer Verbindungen anhand ihres Massenspektrums und nahm Chemiker*innen erstmals die mühsame Aufgabe ab, zu identifizieren, welches Molekül ein vorliegendes Massenspektrum abbildet. Die größere Bedeutung lag jedoch im theoretischen und wissenschaftlichen Beitrag zur Entwicklung wissensbasierter Computersysteme.

Abb. 1.6 Geometrische Figur, die auch auf dem Buchcover des Buches „Perceptrons“ von Marvin Minsky und Seymour Papert zu sehen ist. Sie soll die Kritik der Autoren veranschaulichen, dass die von Rosenblatt vorgestellten Perzeptrone nicht in der Lage sind, festzustellen, ob alle Komponenten in einer solchen Figur miteinander verbunden sind oder die Figur aus separaten Anteilen besteht. [P1522/P1523]

1966 – ELIZA

ELIZA, ein von Joseph Weizenbaum am Massachusetts Institute of Technology (MIT) entwickeltes Programm, war eines der ersten Chatbot-Systeme. Es konnte einfache Konversationen führen und erstmals zeigen, wie menschenähnliche Interaktionen mit Computern möglich sind. ELIZA konnte verschiedene Gesprächspartner simulieren, darunter auch einen Psychotherapeuten. Hierzu nutzte ELIZA ein Wörterbuch, um die Worteingaben zu verstehen bzw. nach Synonymen oder übergeordneten Begriffen für diese Wörter zu suchen. Anschließend wählte ein Skript, das Phrasen zu verschiedenen Themenbereichen enthielt, basierend auf dem übergeordneten Begriff eine Phrase aus. Die Phrasen waren i. d. R. Fragen oder Aufforderungen, die die Konversation weiterführen sollten, z. B.:

- Benutzer*in: „Ich habe ein angespanntes Verhältnis zu meiner Mutter.“
- ELIZA: „Erzählen Sie mir mehr über Ihre Familie!“

In diesem Beispiel hat das Programm die im Wörterbuch festgelegte Zuordnung von „Mutter“ zum Oberbegriff „Familie“ herausgesucht, dabei aber den Inhalt des Satzes keineswegs verstanden. So hätte es auch zu folgendem Dialog kommen können:

- Benutzer*in: „Vorsicht ist die Mutter der Porzellankiste.“
- ELIZA: „Erzählen Sie mir mehr über Ihre Familie!“

1.3 KI-Winter

1970er-Jahre – der erste KI-Winter

In den 1970er-Jahren setzte ein Zeitraum ein, der als erster „KI-Winter“ bekannt wurde. Während dieser Zeit gingen das Interesse und die Investitionen in die KI stark zurück, da die Erwartungen an die Technologie, wie sie z. B. Rosenblatt geweckt hatte, nicht erfüllt werden konnten und viele KI-Projekte als nicht ausgereift oder wirtschaftlich nicht tragfähig angesehen wurden.

1980er-Jahre – erste Expertensysteme

Durch die Veröffentlichung erster Expertensysteme in den 1980er-Jahren nahm das Interesse an der KI wieder zu. Expertensysteme sind Softwareanwendungen, die Expertenwissen eines Bereichs mit Algorithmen kombinieren, um Entscheidungsprozesse zu unterstützen. Eines der ersten Expertensysteme aus dem Bereich der Medizin war „MYCIN", das ab 1972 von Ted Shortliffe an der Stanford University entwickelt wurde. Es unterstützte Ärzt*innen bei der Diagnostik von Infektionskrankheiten und der Auswahl geeigneter Antibiotika, denn zu dieser Zeit wurde der übermäßige Einsatz von Antibiotika zunehmend kritisch betrachtet und nach Methoden gesucht, ihre Anwendung zu optimieren.

Allerdings wurden sowohl ELIZA (➢ Kap. 1.2) als auch Expertensysteme wie MYCIN sehr aufwendig von Menschenhand programmiert, mussten auf große Datenbanken zurückgreifen und konnten nur nach festen Regeln antworten. Dies wurde durch die zu dieser Zeit stark zunehmende Rechengeschwindigkeit und Speicherkapazität ermöglicht. Solche Programme hatten jedoch ihre Grenzen. Durch das hinterlegte Wissen und die definierten Entscheidungsregeln konnten sie zwar spezifische Probleme lösen, waren aber nicht in der Lage, flexibel und eigenständig zu lernen bzw. neue, unvorhergesehene Aufgaben zu lösen.

1990er-Jahre – der zweite KI-Winter

Entsprechend blieben die Resultate solcher Expertensysteme z. B. beim Erkennen von gesprochener Sprache hinter den Erwartungen zurück und läuteten den zweiten KI-Winter ein. KI, die wie von selbst lernen konnte, komplexe Eingabedaten zu verarbeiten und zu verstehen, wie es sich die Väter der KI in Dartmouth vorgestellt hatten, schien in weite Ferne gerückt.

1.4 Revolution der neuronalen Netze

In den 1980er- und 1990er-Jahren wurden große Durchbrüche in der Weiterentwicklung neuronaler Netze erzielt, einer Fortentwicklung von Rosenblatts Perzeptronen, die bis heute zu den wichtigsten Algorithmen der KI zählen. Diese Durchbrüche fielen damit u. a. in die Zeit des zweiten KI-Winters (➢ Kap. 1.3) bzw. sollten diesen schließlich beenden.

1986 – Backpropagation-Algorithmus

1986 veröffentlichten Geoffrey Hinton und seine Kollegen den Backpropagation-Algorithmus [3], mit dem ein neuronales Netz trainiert, also angepasst und in seiner Leistung verbessert werden kann (➢ Abb. 1.7). Hierzu werden sog. Label benötigt, also eine Art „Musterlösung", anhand derer die Leistung des Netzes gemessen werden kann. Beispielsweise soll ein neuronales Netz trainiert werden, Röntgenbilder mit dem Label „gesund" (z. B. repräsentiert durch die Zahl 0) oder dem Label „krankhaft" (z. B. 1) zu versehen. Für jedes Bild, das in das neuronale Netz eingeschleust wird, ist bekannt, ob es in die Klasse „gesund" oder „krankhaft" fällt. Wird ein erstes „krankhaftes" Röntgenbild in das Netz eingeschleust, wird der Input – ähnlich wie bei den Perzeptronen (➢ Kap. 1.2) – mit Gewichten W multipliziert, in eine Aktivierungsfunktion eingesetzt und diese Abfolge für jede weitere Schicht mit Neuronen wiederholt. Am Ende hat das Netz eine Ausgabe berechnet, z. B. die Zahl 0,217. Diesen gesamten Prozess der Berechnung eines Outputs „von vorne nach hinten" bezeichnet man auch als **Forward Pass** (➢ Kap. 12.2.1). Allerdings hätte das Netz eigentlich die Zahl 1 als Musterlösung bzw. Label des „krankhaften" Röntgenbildes ausgeben sollen. Wir können also einen sog. **Error** (Fehler) berechnen, der die Differenz zwischen der Musterlösung bzw. dem Label y und der aktuell berechneten Ausgabe des Netzes $\hat{y}$ darstellt. Aktuell war das Netz mit einer Ausgabe von 0,217 noch weit von der 1 entfernt. Der Error beträgt somit:

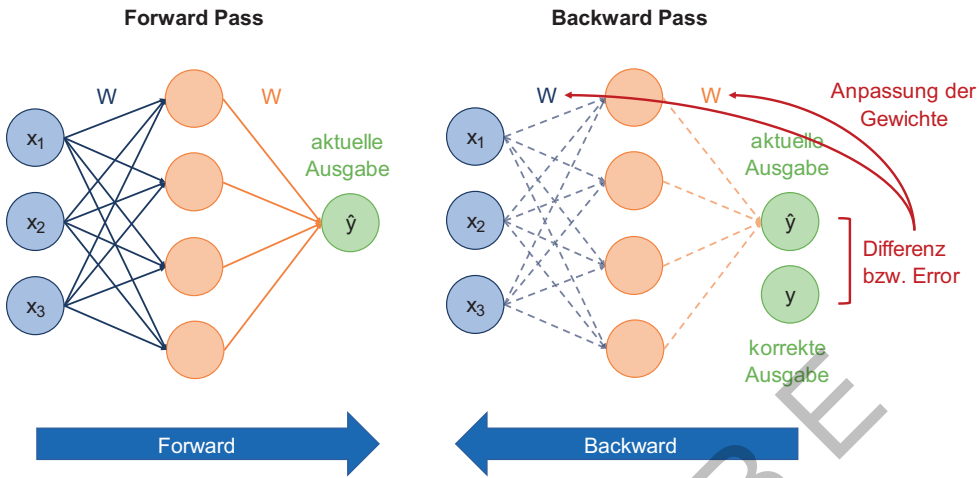


Abb. 1.7 Training eines neuronalen Netzes mittels Backpropagation. Der Backpropagation-Algorithmus ist eine Methode zur Verbesserung des neuronalen Netzes während des Trainingsprozesses. Dabei wird im Forward Pass zunächst die aktuelle Ausgabe des Modells berechnet, indem die Input-Werte x mit den Gewichten W multipliziert und die Aktivierungsfunktionen (hier nicht gezeigt) angewendet werden. Anschließend wird der Error ermittelt, also die Differenz zwischen der korrekten Ausgabe bzw. des Labels y und der aktuellen Ausgabe $\hat{y}$. Zur Minimierung dieses Error wird der Einfluss jedes Gewichts auf den Gesamtfehler bestimmt und das Gewicht entsprechend angepasst. Die Gewichte werden dabei rückwärts angewendet (Backward Pass), also in entgegengesetzter Richtung zur Berechnung der aktuellen Ausgabe. [P1522/P1523]

Error = y – $\hat{y}$ = 1–0,217 = 0,783. Dieser Error kann mithilfe der von Hinton entwickelten Methode genutzt werden, um die Gewichte des neuronalen Netzes so anzupassen, dass die Leistung des neuronalen Netzes immer besser wird. Im nächsten Schritt beträgt der Error also womöglich nur noch 0,612, im darauffolgenden 0,421 usw. Mit abnehmendem Error wird die Präzision des Netzes immer besser. Die rückwirkende Anpassung der Gewichte nach Berechnung des Errors wird auch als **Backward Pass** bezeichnet (➢ Kap. 12.2.2). Der Begriff **Backpropagation** bezieht sich auf Kombination aus Forward und Backward Pass.

Der Backpropagation-Algorithmus funktioniert grundsätzlich bei einer beliebigen Anzahl an Schichten, einer beliebigen Anzahl an Neuronen sowie auch bei anderen Aktivierungsfunktionen als etwa der Heaviside-Funktion (➢ Kap. 1.2). Somit können mittels Backpropagation auch Netze trainiert werden, die sehr komplexe Modellierungen vornehmen (➢ Abb. 1.8).

1989 – Convolutional Neural Networks

Mit den Convolutional Neural Networks (CNNs) führten Yann LeCun und seine Kollegen 1989 eine neue, besonders für die Bildverarbeitung geeignete Form der neuronalen Netze ein [4]. CNNs betrachten größere Bildbestandteile (also mehrere Pixel) auf einmal und können Bildmuster und -merkmale besser erkennen als die klassischen neuronalen Netze. In den ersten Schichten werden zunächst feine Muster erkannt (z. B. Bögen und Kurven), die dann über mehrere Schichten hinweg zu größeren Mustern (z. B. der Zahl 6) zusammengesetzt werden (➢ Abb. 1.9). Am Ende steht ein klassisches neuronales Netz, das nach der Vorarbeit der sog. **Convolutional Layer** die Klassifizierung des Bildes vornimmt (s. a. ➢ Kap. 13).

Als die CNN-Architektur AlexNet 2012 die ImageNet Large Scale Visual Recognition Challenge gewann, einen jährlich ausgetragenen Wettbewerb, bei dem die standardisierte Bilddatenbank ImageNet verarbeitet und klassifiziert werden soll, wurde die Überlegenheit der CNNs deutlich. Das gilt auch für den medizinischen Bereich, wo CNNs z. B. zur Analyse radiologischer Bilder etwa zum Nachweis und zur Klassifizierung von Tumoren genutzt werden.

1990er-Jahre – neuronale Netze für die Sprachverarbeitung

Damit neuronale Netze auch sequenzielle Daten wie Sprache verarbeiten können, müssen sie vorherige Modellzustände bzw. Eingabedaten berücksichtigen können, um zukünftige Zustände vorherzusagen. Soll z. B. ein Satz interpretiert werden, muss das Modell die Bedeutung vorangegangener Wörter berücksichtigen, um die des nächsten Wortes in den richtigen Kontext setzen zu können. Dass das entscheidend ist, zeigt das Beispiel ELIZA (➢ Kap. 1.2), das die Wörter des Eingabesatzes nur getrennt voneinander analysieren konnte und somit nicht über das Sprachverständnis und die Flexibilität heutiger Sprachsysteme verfügte.

Um dies zu ermöglichen, wurden sog. **Recurrent Neural Networks (RNNs)** entwickelt, die wie die CNNs eine Weiterentwicklung einfacher neuronaler Netze sind. Das erste lernfähige RNN wurde 1972 vom japanischen Neuro- und Computerwissenschaftler Shun'ichi Amari beschrieben [5]. Im Gegensatz zu den klassischen neuronalen Netzen, bei denen Daten im Rahmen des Forward Pass nur in eine Richtung fließen (➢ Abb. 1.7 links), verfügen RNNs über Rückkopplungsschleifen, die es ihnen ermöglichen, Informationen aus vorherigen Verarbeitungsschritten zu speichern und in zukünftigen Berechnungen erneut zu berück-

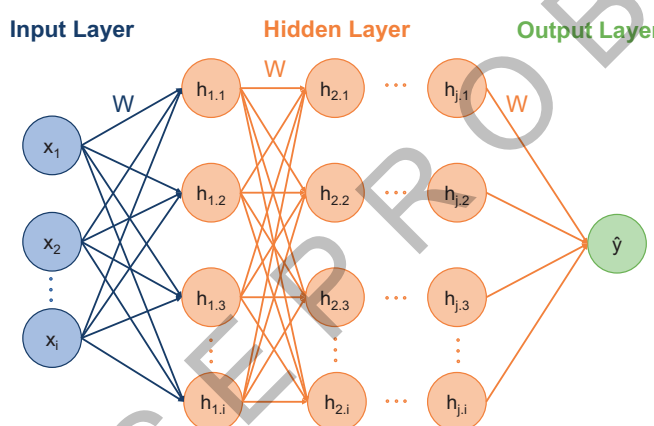


Abb. 1.8 Großes neuronales Netz. Die erste Schicht wird als Input Layer (Eingabeschicht) bezeichnet, die Schichten in der Mitte des Netzes als Hidden Layer (versteckte Schicht), abschließend folgt ein Output Layer (Ausgabeschicht). Auch in großen neuronalen Netzen funktioniert Backpropagation. [P1522/P1523]

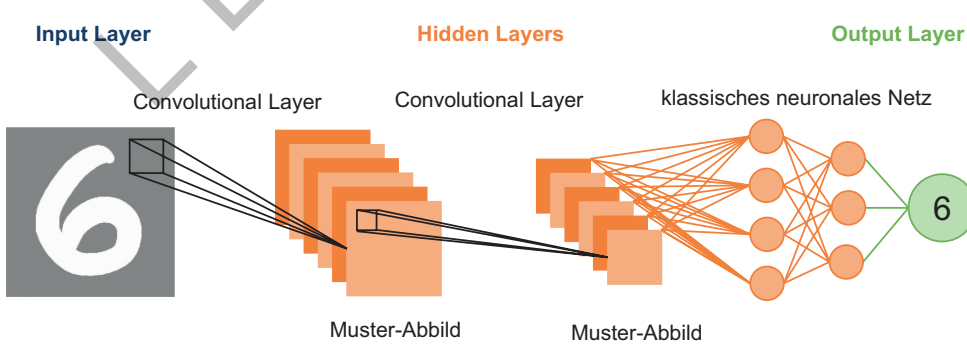


Abb. 1.9 Convolutional Neural Networks (CNNs) sind neuronale Netze, die sich besonders für die Bildverarbeitung und -analyse eignen. Sie enthalten Convolutional Layers, also spezielle Schichten, in denen Muster wie Kanten, Textur und Formen aus den Eingabebildern extrahiert werden können. Die extrahierten Muster werden in einem Muster-Abbild zusammengefasst und an die nächste Schicht weitergegeben. Anhand der in den Convolutional Layern identifizierten Muster wird das Eingabebild am Ende in einem klassischen neuronalen Netz klassifiziert. [P1522/P1523]

sichtigen (➢ Abb. 1.10). Auf diese Weise eignen sich RNNs für die Verarbeitung sequenzieller Daten wie Sprache (➢ Kap. 15.5). Somit handelte es sich um wichtige Fortschritte in dem Feld des sog. **Natural Language Processing (NLP),** der natürlichen Sprachverarbeitung.

In den 1990er-Jahren entwickelten der deutsche Informatiker Jürgen Schmidhuber und sein Mitarbeiter Sepp Hochreiter das Konzept der **Long Short-Term Memory (LSTM).** Diese Weiterentwicklung der RNNs ermöglicht es, Zusammenhänge zwischen noch weiter voneinander entfernt liegenden Sequenzabschnitten zu berücksichtigen (➢ Kap. 15.5.3).

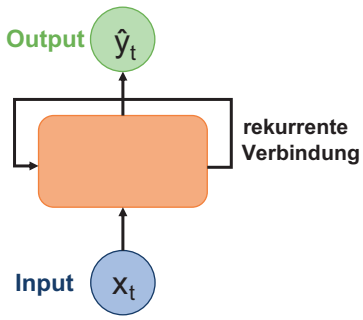


Abb. 1.10 Ein Recurrent Neuronal Network (RNN) ist ein neuronales Netz, das speziell für die Verarbeitung sequenzieller Daten wie Sprache entwickelt wurde. Es verwendet rekurrente Verbindungen bzw. Rückkopplungsschleifen, um Informationen aus vorherigen Verarbeitungsschritten in die aktuellen Berechnungen einzubeziehen. [P1522/P1523]

1.5 Moderne Durchbrüche in der KI

1997 – IBM Deep Blue

Der Schachcomputer Deep Blue der Firma IBM schlug im Jahr 1997 den Schachweltmeister Garry Kasparov und zeigte öffentlichkeitswirksam die Leistungsfähigkeit von Computern in strategischen Spielen. Kasparov war von der Fähigkeit der Maschine allerdings nicht überzeugt und wurde mit den Worten zitiert, der Computer sei nicht intelligenter gewesen als ein „programmierbarer Wecker" [6]. So habe Deep Blue durch seine schiere Rechenleistung gewonnen, die es ihm ermöglichte, rund 200 Millionen mögliche Schachzüge pro Sekunde durch Brute-Force-Berechnungen zu analysieren. Deep Blue nutzte schließlich kein Machine Learning, sondern basierte auf klassischer algorithmischer Schachprogrammierung mit einer umfangreichen Datenbank bekannter Schachzüge, um seine Entscheidungen zu treffen.

1998 – humanoider Roboter Kismet

Anders als Deep Blue verwendete der humanoide Roboter Kismet Machine Learning, um menschenähnliche Interaktionen und soziale Fähigkeiten zu zeigen. Mithilfe neuronaler Netze war Kismet zu Gesichts- und Spracherkennung fähig und konnte auf menschliche Emotionen und soziale Signale reagieren. Es handelt sich um eines der frühen Beispiele für den Einsatz neuronaler Netze in der Robotik sowie der Mensch-Maschinen-Interaktion. Entwickelt wurde Kismet von Cynthia Breazeal im renommierten MIT MediaLab. Später wurden soziale Roboter entwickelt, die über wesentlich komplexere Fähigkeiten als Kismet verfügten und heute z. B. in der Kranken- und Altenpflege zum Einsatz kommen. Artikel, die in dieser Zeit über Kismet berichteten, erhielten Überschriften wie „Die Geburt des Robo sapiens" und adressierten bereits die Sorge, dass der Homo sapiens eines Tages durch diese „neue Gattung" verdrängt werden könnte.

2007 – ImageNet, ILSVRC und die Nutzung von Grafikkarten

2007 veröffentlichte Fei Fei Li an der Stanford University die bereits beschriebene Datenbank ImageNet, die dem Training von Algorithmen dienen sollte. Die Datenbank enthält viele Millionen mit Labeln versehene Bilder, wobei jedes Bild einem von mehr als 20.000 englischsprachigen Substantiven zugeordnet wurde. Seit

dem Jahr 2010 organisiert das ImageNet-Projekt alljährlich die bereits erwähnte ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Der Sieg des AlexNet in der ILSVRC 2012 war nicht nur durch die erstmalige Nutzung einer CNN-Architektur möglich. Der Algorithmus war zudem so entwickelt worden, dass er sich auf Grafikkarten (Graphics Processing Units, GPUs) trainieren ließ. Grafikkarten sind besonders schnell bei der parallelen Berechnung von Matrix-Multiplikationen, einem der wesentlichen Rechenschritte neuronaler Netze. Auf diese Weise konnte die Trainingszeit des Algorithmus erheblich verkürzt werden.

Inzwischen gibt es standardisierte Datensätze wie ImageNet auch für den medizinischen Bereich, darunter CT-Bildsammlungen mit verschiedenen Erkrankungen sowie klinische und biomedizinische Daten wie Proteinstrukturen und Sequenzierungsdaten. Zudem helfen **Programmierbibliotheken für Python** wie TensorFlow oder PyTorch Entwickler*innen bei der Ausführung von Code auf Grafikkarten, sodass dafür kein eigener Code geschrieben werden muss. Diese Frameworks übernehmen auch die Datenübertragungen zwischen Grafikkarten (GPUs) und dem Hauptprozessor des Computers (Central Processing Unit, CPU) und beschleunigen das Modelltraining, sodass die Entwicklung komplexer KI-Modelle heute einfacher und schneller möglich ist.

2011 – IBM Watson

Knapp 15 Jahre nach Deep Blue stellte IBM Watson vor, ein KI-basiertes Expertensystem, das bei der TV-Quizshow „Jeopardy!“ die bis dahin besten Spieler Ken Jenning und Brad Rutter besiegte. Watson nutzte dabei NLP, um die Jeopardy!-Fragen zu verstehen. Es wurde mit einer Vielzahl von Textquellen, darunter Enzyklopädien, Bücher und Websites trainiert, um ein breites Wissensspektrum zu entwickeln.

Im selben Jahr begann IBM eine Partnerschaft mit dem Krebsforschungsinstitut Memorial Sloan-Kettering Cancer Center in New York, aus der Watson for Oncology hervorging. Das Expertensystem wurde 2013 auf den Markt gebracht und unterstützt Ärzt*innen bei der Diagnostik und Behandlung von Krebserkrankungen. IBM verwendet den Namen Watson noch heute für viele seiner Softwareangebote. Neben IBM entwickelt heute eine Vielzahl weiterer Firmen KI-gestützte Expertensysteme für die Medizin.

2011 – Sprachassistenten

Durch bedeutende Fortschritte in der Rechenleistung sowie immer leistungsstärkere Algorithmen zur Sprachverarbeitung fand KI in den 2010er-Jahren zunehmend ihren Weg in den Alltag der Allgemeinbevölkerung. Moderne Prozessoren und Grafikkarten in Computern, Smartphones und Tablets ermöglichen es Verbrauchern, in jeder Situation auf KI-Programme zuzugreifen. Mit „Siri“ veröffentlich Apple 2011 zuerst ein Sprachassistenten-System, das als intelligente persönliche Assistentin dienen soll. Wenige Jahre später brachte Microsoft das Äquivalent „Cortana“ auf den Markt (2014), und Amazon präsentierte „Alexa“ (2015).

KI-basierte Sprachdienste finden auch zunehmend im Krankenhaus Anwendung und unterstützen z. B. die Dokumentation, geben Informationen an Patient*innen weiter, organisieren die Terminvergabe oder werden in sprachgesteuerte Geräte integriert.

2014 – Generative Adversarial Networks (GANs)

Ian Goodfellow und seine Kollegen an der University of Montreal stellten 2014 die GAN-Architektur (GAN = Generative Adversarial Network) vor [7]. Damit wurde das Zeitalter der „generative AI“ (generative KI) eingeläutet, denn mithilfe dieser Algorithmen lassen sich Bilder, Texte, Musik u. v. m. künstlich erzeugen. Dazu nutzen GANs 2 spezielle neuronale Netze – den **Generator** und den **Diskriminator.** Der Generator erzeugt neue und möglichst originalgetreue Daten, wohingegen der Diskriminator versucht, zwischen den echten und den neu generierten Daten zu unterscheiden. Diese beiden Netzwerke trainieren also gegeneinander, was dazu führt, dass der Generator im Lauf der Zeit immer realistischere Daten erzeugt, bis generierte und originale Daten kaum noch zu unterscheiden sind (➤ Abb. 1.11, ➤ Kap. 15.4).

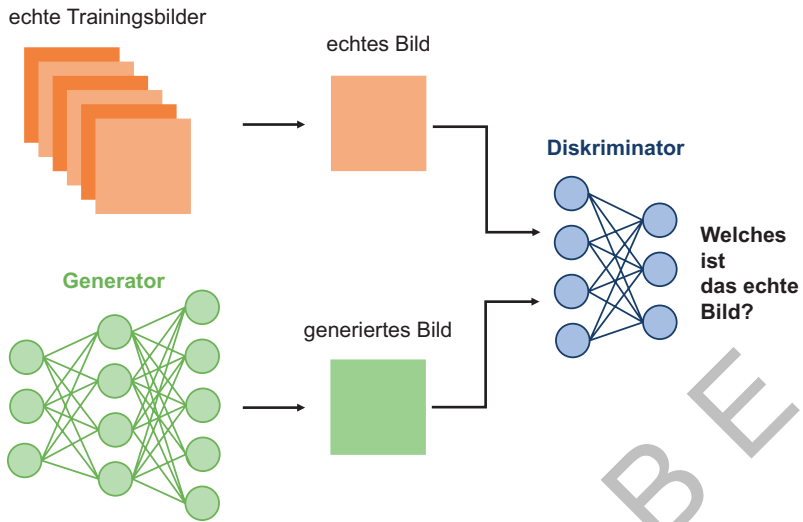


Abb. 1.11 Ein Generative Adversarial Network (GAN) ist eine besondere neuronale Netzarchitektur, die neue Daten wie z. B. Bilder generieren kann. GANs bestehen aus 2 Hauptkomponenten, einem Generator und einem Diskriminator. Der Generator erzeugt neue Daten, während der Diskriminator zwischen echten und generierten Daten zu unterscheiden versucht. Durch diesen Wettbewerb zwischen beiden Komponenten lernt das GAN, künstliche, aber realistische und den Trainingsdaten ähnliche Daten zu produzieren. [P1522/P1523]

In der Medizin können GANs z. B. genutzt werden, um künstliche Daten für das Training anderer Algorithmen zu generieren. Außerdem unterstützen sie die Medikamentenentwicklung, denn mit ihrer Hilfe können Moleküle de novo generiert werden, die die optimalen Eigenschaften eines gesuchten Pharmakons besitzen und bisher ggf. noch nicht synthetisiert worden sind. Das beschleunigt die Entwicklung neuer Medikamente und senkt die Kosten. Ein prominentes Beispiel dafür, wie Algorithmen die Entwicklung neuer Therapeutika unterstützen können, sind die COVID-19-Impfstoffe. Hier wurde generative KI verwendet, um die Struktur der SARS-CoV-2-RNA vorhersagen, sodass basierend auf den gewonnenen Erkenntnissen potenzielle mRNA-Impfstoffe in kürzerer Zeit entwickelt werden konnten.

2016 – AlphaGo und Partnership on AI

Im Jahr 2016 schlug das von Google DeepMind entwickelte Computerprogramm AlphaGo den amtierenden Weltmeister im Brettspiel Go, Lee Sedol. Der Sieg gegen Sedol zeigte nach IBM Deep Blue und Watson erneut die Fähigkeit von KI, selbst komplexe Strategiespiele zu beherrschen bzw. „übermenschliche" Fähigkeiten zu demonstrieren.

Im gleichen Jahr schlossen Google bzw. die Dachgesellschaft Alphabet sowie Amazon, Facebook, IBM und Microsoft die **„Partnership on AI"** – eine Allianz, um die verantwortungsbewusste Nutzung und Entwicklung von KI zu fördern. Die Grundsätze dieser Allianz lauten im Wesentlichen, dass sie

- KI entwickeln will, die der Menschheit dient und ihr nicht schadet,
- sich Kritik und Fragen aus der Öffentlichkeit stellt und
- offen für Forschung sowie einen Dialog über ethische, soziale und wirtschaftliche Fragen ist.

Die verantwortungsvolle Anwendung von KI, insbesondere in der Medizin, ist Inhalt vieler ethischer und juristischer Debatten. Obwohl sowohl auf europäischer als auch auf nationaler Ebene Gesetze verabschiedet und Vorgaben eingeführt wurden, die u. a. die Entwicklung, den Vertrieb und die Nutzung intelligente Systeme regeln sollen, sind viele Fragen z. B. zum Datenschutz oder der Haftbarkeit solcher Systeme noch offen (s. a. ➢ Kap. 4).

2017 – Transformer

2017 beschrieben Ashish Vaswani und Kolleg*innen die Grundlagen von **Transformern,** besonders leistungsstarken KI-Algorithmen für die Verarbeitung sequenzieller Daten wie Sprache [8]. 2018 veröffentlichte die Firma OpenAI die erste Version ihres Algorithmus **Generative Pre-trained Transformer Version (GPT-1).** 2022 veröffentlichte Open AI schließlich den Chatbot ChatGPT, in den die GPT-Modelle eingebettet sind. Auch Microsoft, Google und andere Firmen veröffentlichten frei verfügbare große Sprachmodelle, sodass leistungsstarke KI für die gesamte Bevölkerung zugänglich wurde.

Auch in der Medizin scheinen solche fortschrittlichen Sprachmodelle neue Türen zu öffnen. So können sie von Patient*innen potenziell dafür genutzt werden, die möglichen Ursachen von Symptomen zu recherchieren oder den Inhalt von Untersuchungsbefunden oder ärztliche Briefen besser zu verstehen. Ärzt*innen könnten sie die Möglichkeit bieten, Vorbefunde zu sichten und ggf. zusammenzufassen, diagnostische Prozesse zu automatisieren und zu optimieren, bei Therapieentscheidungen zu assistieren oder ärztliche Briefe zu schreiben. Damit rücken jedoch auch die enormen ethischen und juristischen Implikationen weiter in den Vordergrund.

1.6 Hype oder Hope?

Es dürfte deutlich geworden sein, dass die Erforschung und Weiterentwicklung der KI keineswegs ein linearer Prozess war (➢ Abb. 1.12). Insbesondere während des zweiten KI-Winters in den 1970er- und 1990er-Jahren wurden der KI nur wenig Forschungsinteresse bzw. Fördergelder gewidmet, nachdem die hohen Erwartun-

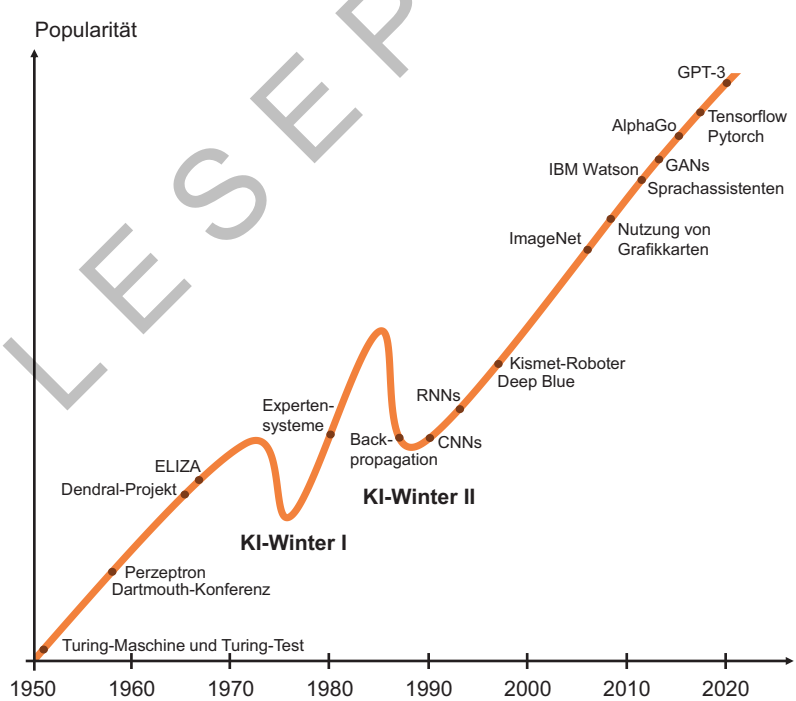


Abb. 1.12 Entwicklungen im Bereich der KI mit Fortschritten und Rückschlägen über mehrere Jahrzehnte; CNNs = Convolutional Neural Networks, RNNs = Recurrent Neural Networks, GANs = Generative Adversarial Networks, GPT-3 = Generative Pre-trained Transformer Version 3 [P1522/P1523]

gen nicht erfüllt werden konnten. Auch heute wird KI einerseits als Revolution aufgefasst, andererseits gibt es jedoch auch viele kritische Stimmen, die die Leistungsfähigkeit und Sicherheit von KI hinterfragen oder die großen ethischen Herausforderungen betonen.

Ein Modell zur Beschreibung der allgemeinen Akzeptanz von bzw. Erwartungen an Technologien und Innovationen ist der sog. **Gartner-Hype-Cycle,** der in den 1990er-Jahren von der IT-Beratungsfirma Gartner entwickelt wurde (➤ Abb. 1.13). Dieser Zyklus kann grundsätzlich auch mehrfach durchlaufen werden. Wichtig zu betonen ist, dass es sich jedoch lediglich um ein Modell handelt, das kaum Vorhersagen über die zukünftige Entwicklung der KI zulässt. Nachdem KI mittlerweile in vielen Bereichen einen festen Stellenwert gewonnen hat, könnte es sein, dass wir uns aktuell im „Plateau der Produktivität" befinden. Jedoch bleibt abzuwarten, wie weit die Fortschritte in der KI reichen werden oder ob Rückschläge eines Tages womöglich in einen dritten KI-Winter münden.

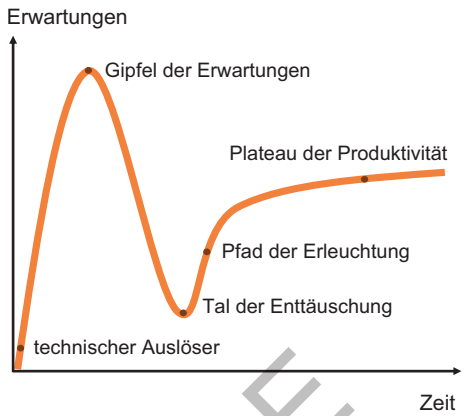


Abb. 1.13 Nach dem Gartner-Hype Cycle lässt sich die Entwicklung der Erwartungen an eine technische Innovation in mehrere Phasen einteilen: Nach der Erfindung entsteht zunehmendes Interesse mit einem Höhepunkt in der öffentlichen Aufmerksamkeit, verknüpft mit hohen Erwartungen („Gipfel der Erwartungen"). Können diese nicht erfüllt werden, entsteht Enttäuschung („Tal der Enttäuschung"). Später folgt dann ein besseres Verständnis und eine realistischere Einschätzung des Nutzens der Technologie („Pfad der Erkenntnis"). Dies mündet schließlich in eine Phase, in der die Technologie zunehmend angewendet und weiterentwickelt wird („Plateau der Produktivität"). [P1522/P1523]

Zusammenfassung

- 1936: Alan Turing beschreibt die **Turing-Maschine,** ein Modell dessen, wie ein Rechner eine formalisierte Aufgabe lösen kann.
- 1950: Alan Turing beschreibt den **Turing-Test.** Demnach gilt ein Algorithmus als intelligent, wenn ein Mensch nicht zwischen den Ausgaben des Algorithmus und den eines Menschen unterscheiden kann.
- 1956: Die **Dartmouth-Konferenz** gilt als Geburtsstunde der KI als Forschungsgebiet.
- 1958: Frank Rosenblatt entwickelt den **Perzeptron-Algorithmus,** der das Fundament für künstliche neuronale Netze legte.
- Erster (1970er-Jahre) und zweiter **KI-Winter** (1990er-Jahre): Rückgang des Interesses und der Investitionen in KI, da die hohen Erwartungen nicht erfüllt werden konnten.
- 1986: Geoffrey Hinton et al. beschreiben den **Backpropagation-Algorithmus,** mit dem neuronale Netze trainiert werden können.
- 1989: Yann LeCun et al. beschreiben **Convolutional Neural Networks** (CNNs), die sich besonders für die Bildverarbeitung eignen.
- 2000er-Jahre: Die Sprünge in der Rechenleistung und der Zuwachs frei verfügbarer Datensätze beschleunigen die Entwicklung und das Training von KI-Algorithmen.
- 2014: Ian Goodfellow et al. beschreiben **Generative Adversarial Networks** (GANs), mit der sich Daten künstlich generieren lassen (generative AI).
- 2017: Ashish Vaswani et al. beschreiben **Transformer,** besonders leistungsstarke Algorithmen für die Verarbeitung sequenzieller Daten wie Sprache.
- Der **Gartner-Hype-Cycle** ist ein Modell zur Beschreibung der Phasen der öffentlichen Aufmerksamkeit, die eine neue Technologie bei ihrer Einführung durchläuft.

KAPITEL

2 Was ist künstliche Intelligenz und was kann sie?

Thema dieses Kapitels sind wichtige Grundbegriffe, die im Alltag häufig verwendet werden und die Funktionsweise oder Fähigkeiten der Algorithmen beschreiben.

Ein **Algorithmus** ist ein Verfahren zur schrittweisen (mathematischen) Lösung eines Problems (➢ Kap. 1). In der Informatik versteht man unter einem Algorithmus meist eine Anweisung oder eine Gruppe an Anweisungen, die festlegt, wie Eingabeinformationen bzw. -daten mathematisch verarbeitet werden, um eine gewisse Ausgabe zu erzielen. Programmiersprachen wie Python, Java, oder C wurden entwickelt, um dem Computer den Algorithmus so zu erklären, dass dieser die Berechnungen durchführen kann. Im Zusammenhang mit der KI erstellt ein Algorithmus meist ein sog. **Modell,** also eine Repräsentation bzw. Abbildung der Realität, z. B. mithilfe einer mathematischen Formel. Der Begriff **System** (z. B. KI-System) meint meist eine gesamte Software und/oder Hardware, die KI nutzt bzw. beinhaltet.

CAVE

Für die meisten der in diesem Kapitel eingeführten Begriffe existieren keine offiziellen und allseits anerkannten Definitionen – das beginnt schon beim Begriff „künstliche Intelligenz". Da die Begriffe dennoch oft verwendet werden, ist ein grundlegendes Verständnis wichtig.

2.1 Künstliche Intelligenz, Machine Learning und Deep Learning

Künstliche Intelligenz (KI) ist der Bereich der Informatik, der all jene Algorithmen umfasst, die in irgendeiner Form biologischer Intelligenz ähneln bzw. diese nachahmen. In den 1980er-Jahren beschrieb die US-amerikanische Informatikerin Elaine Rich KI wie folgt: *„Artificial intelligence is the study of how to make computers do things at which, at the moment, people are better"* [9]. Allerdings sind wir mittlerweile in einem Zeitalter angelangt, in der KI den Menschen in mancherlei Hinsicht zu überholen scheint.

Je nach Definition umfasst der Begriff KI eine Vielzahl unterschiedlicher Algorithmen von einfacher Logik über Regressionen bis hin zu großen neuronalen Netzen. Die KI weist damit große Überschneidungen mit der Statistik auf, die ebenfalls der Entdeckung von Zusammenhängen und dem Treffen von Vorhersagen dient.

Zwei Begriffe, die häufig im Zusammenhang mit KI auftauchen und manchmal auch fälschlicherweise als Synonyme für KI verwendet werden, sind Machine Learning (maschinelles Lernen) und Deep Learning (tiefes Lernen): **Machine Learning** ist ein Teilbereich der KI mit Algorithmen, die lernen und sich anpassen können, ohne ausdrückliche Anweisungen zu befolgen. Gemeint sind hierbei etwas komplexere KI-Algorithmen wie Support Vector Machines (SVM) oder auch neuronale Netze. **Deep Learning** ist wiederum ein Teilbereich des Machine Learning (und damit auch der KI), der umfangreiche („tiefe") neuronale Netze beinhaltet, die besonders komplexe Berechnungen durchführen und Muster in Daten erkennen können. Mit Deep Learning lassen sich statische oder bewegte Bilder auswerten, natürliche Sprache verarbeiten oder sogar das Faltungsverhalten oder die Interaktion von Proteinen prüfen. In der Medizin ist Deep Learning daher aktuell sicherlich einer der wichtigsten Teilbereiche der KI (➢ Kap. 3).

KAPITEL

5 Einführung in die Programmierung mit Python

In Teil II werden einige KI-Algorithmen im Detail erläutert. Neben Erklärungen zu den Anwendungsbereichen und der mathematischen Funktionsweise der Algorithmen ist hier auch beispielhafter Code in der Programmiersprache Python enthalten, die sich besonders für KI-Anwendungen eignet. Nachdem die Programmierung eine Kunst und Herausforderung für sich ist, beinhaltet das aktuelle Kapitel eine allgemeine Einführung in die Programmierung.

Bei einem Algorithmus handelt es sich im Allgemeinen um ein Verfahren zur schrittweisen (mathematischen) Lösung eines Problems (➢ Kap. 1.1). Programmiersprachen wie Python, Java, oder C wurden entwickelt, um dem Computer den Algorithmus so zu erklären, dass dieser die Berechnungen automatisiert durchführen kann.

5.1 Wie kommuniziert man eigentlich mit einer Maschine?

Computer kommunizieren in binärer Sprache, d. h. mit einer Abfolge aus 1en und 0en. Warum ist das so und was haben Programmiersprachen damit zu tun? Ein Computer besteht vereinfacht aus einer Recheneinheit (dem Prozessor), Speichereinheiten sowie Ein- und Ausgabegeräten zur Ein- und Ausgabe von Informationen.

Die meiste Magie geschieht im Prozessor, der das „Gehirn" des Computers darstellt. Er wird daher auch als zentrale Verarbeitungseinheit bzw. **Central Processing Unit (CPU)** bezeichnet. Dort werden eingegebene Befehle interpretiert und ausgeführt. Der Prozessor setzt sich heute i. d. R. aus mehreren **Kernen** zusammen, die die eigentlichen Recheneinheiten sind, also Orte, an denen die Datenverarbeitung stattfindet. Je mehr Kerne ein Prozessor aufweist, desto leistungsstärker ist er – vereinfacht gesagt. Darüber hinaus gibt es neben der CPU auch noch weitere spezialisierte Prozessoren, wie die **Graphics Processing Unit (GPU),** die insbesondere für die Verarbeitung grafischer Prozesse wichtig ist. Damit sind nicht nur Bild- und Video-Verarbeitungsprogramme gemeint, sondern auch die visuelle Aufbereitung von Software wie etwa bei Computer-Spielen. GPUs eignen sich auch für KI-Anwendungen, da sie große Datenmengen verarbeiten und für die KI wichtige Matrixoperationen effizient durchführen können.

Die kleinste funktionelle Einheit eines Prozessors ist der **Transistor.** Ein Transistor ist ein winzig kleiner Schalter, der entweder Strom durchlässt oder nicht. Hier liegt der Ursprung der binären Sprache von Computern in Form von 1en und 0en. Die 1en stehen vereinfacht gesagt für Stromfluss und die 0en für das Ausbleiben von Stromfluss. Man kann sich einen Transistor somit vorstellen wie einen Schalter, der den Stromkreis zu einer Lampe öffnen und schließen kann (➢ Abb. 5.1). Ist der Schalter geöffnet, fließt kein Strom und die Lampe leuchtet nicht. Dieser Zustand kann als 0 interpretiert werden. Ist der Schalter geschlossen und die Lampe leuchtet, spiegelt das den Zustand 1 wider. Mit einer Lampe sind selbstverständlich nur 2 Zustände möglich: an und aus. Mit 2 Lampen lassen sich jedoch bereits 4 Zustände darstellen: an-an, an-aus, aus-an, aus-aus; mit 3 Lampen dann 8 und so weiter. Es lassen sich also 2^n Zustände darstellen, wobei n der Anzahl an Lampen bzw. Transistoren entspricht.

Die Abfolge aus 1en und 0en bezeichnet man auch als **Maschinensprache.** Es wäre allerdings wenig praktikabel, wenn Programmierung nur durch Eingabe von 1en und 0en funktionieren würde. Besser wäre eine nutzer*innenfreundlichere, wenn auch klar definierte, eindeutige Programmiersprache. Daher wurden sog. **höhere Programmiersprachen** (z. B. Python, Java, C) entwickelt, die für den Menschen besser verständlich und einfacher zu handhaben sind. Im Computer sitzt ein Dolmetscher, der sog. **Compiler,** der die höhere Programmiersprache in Maschinensprache übersetzt, damit die Recheneinheit die menschlichen Befehle versteht. Es gibt mittlerweile unzählige höhere Programmiersprachen, die unterschiedlich populär und für manche Anwendungen jeweils besonders geeignet sind (➢ Tab. 5.1).

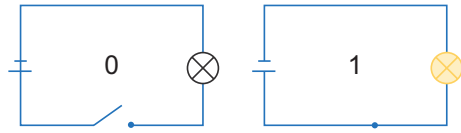


Abb. 5.1 Lichtschalter zur Verbildlichung der Funktionsweise eines Transistors. Transistoren sind die kleinsten funktionellen Einheiten eines Computer-Prozessors. Sie entsprechen winzig kleinen Schaltern und stellen die Grundlage der binären Computersprache dar. Zur Beschreibung der Funktionsweise eignet sich ein Lichtschalter. Ist dieser Schalter geöffnet, fließt kein Strom und die Lampe leuchtet nicht; dies entspricht der Zahl 0. Ist der Schalter geschlossen, fließt Strom und die Lampe leuchtet; dies entspricht der Zahl 1. [P1522/P1523]

Obwohl es viele verschiedene Programmiersprachen gibt, hat sich Python im Feld der KI durchgesetzt. Dies hat mehrere Gründe:

- Python weist eine einfache Syntax und eine gute Lesbarkeit auf, die es einfacher machen, komplexe KI-Algorithmen zu programmieren und zu verstehen.
- Python verfügt über eine große Community und eine Fülle sog. **Bibliotheken** für KI-Anwendungen (z. B. TensorFlow, PyTorch, scikit-learn und NumPy). Diese Bibliotheken erleichtern die Entwicklung von KI-Anwendungen, da sie viele vereinfachte Programmierbefehle zu Verfügung stellen, mit denen man selbst mathematisch komplexe Algorithmen in oft nur wenigen Zeilen programmieren kann.
- Python ist plattformübergreifend kompatibel und kann auf verschiedenen Betriebssystemen wie Windows, Mac und Linux ausgeführt werden.

Tab. 5.1 Überblick über einige heute populäre Programmiersprachen

Programmiersprache	Vorteile	Wichtige Anwendungen
Python	Einfache Syntax, große Community und viele Bibliotheken, Kompatibilität, Flexibilität, Skalierbarkeit	Jegliche KI-Anwendungen, Statistik, Web Development
R	Große Community und viele Bibliotheken, einfache Handhabung großer Datenmengen, Visualisierungsfunktionen	KI, Statistik
C/C++	Hohe Geschwindigkeit und Effizienz, Kontrolle über Hardware, Ausführung von Systemebenen-Operationen	Maschinen inkl. Robotik, Systemprogrammierung, Computerspiele
C# („C sharp")	Hohe Geschwindigkeit und Effizienz, viele Bibliotheken	Windows-Apps, Windows-Desktop-Anwendungen, Web-Development
Java	Gute Kompatibilität, Sicherheitsfunktionen, einfache Handhabung von Threads	Android-Apps, Desktop-Anwendungen, Web-Entwicklung
JavaScript	Einfache Syntax, einfache Integration in Webseiten, viele Nutzer*innen-Umgebungen und Bibliotheken	Front-End-Webentwicklung, Apps
PHP	Einfache Handhabung von Datenbanken und deren Integration in Webanwendungen	Content-Management-Systeme, eCommerce-Plattformen, Webentwicklung
SQL	Weitverbreitet, effiziente und konsistente Datenmanipulation und -abfrage	Erstellung und Bearbeitung von Datenbanken

- Python ist eine flexible Sprache, die es ermöglicht, schnell Prototypen von KI-Anwendungen zu erstellen und zu testen, bevor man sich auf eine endgültige Implementierung festlegt.
- Python kann problemlos auf große Datenmengen und komplexe Anwendungen skaliert werden.

In diesem und den folgenden Kapiteln beschränken wir uns daher auf Python, um zu erklären, wie die mathematischen Konstrukte der KI in Programmiersprache aussehen.

5.2 Datentypen

Jede*r von uns weiß, „3 + 3“ ergibt „6“ und „3 + Auto“ ergibt keinen Sinn. Das liegt daran, dass wir wissen, dass man Zahlen addieren kann, nicht aber Buchstaben bzw. Worte. Jedoch können Buchstaben bzw. Worte auch stellvertretend für Zahlen stehen, sodass „3 + x“ als mathematische Formulierung wiederum Sinn ergibt.

In der Informatik spricht man in diesem Kontext von sog. **Datentypen.** Der Datentyp gibt an, von welcher Art die Daten sind (z. B. Zahl oder Wort) und welche Verarbeitungsschritte bzw. Operationen auf diesen ausgeführt werden können. Zu den wichtigsten Datentypen gehören:

- `int` (Integer): Integer sind ganze Zahlen, also z. B. 1 oder –15.
- `float` (Floating point value): Floats sind Zahlen mit Dezimalstelle, also z. B. 2,15.
- `bool` (Boolean): Booleans können lediglich 2 Zustände einnehmen, nämlich „wahr“ oder „falsch“.
- `char` (Character): Characters entsprechen numerischen Codierungen von Text, also z. B. 97 stellvertretend für den Buchstaben „a“.
- `string` (String of Characters): Strings sind Wörter, also z. B. „Hallo“.

Es ist wichtig, diese Datentypen zu kennen, da vom Datentyp abhängt, welche Operationen möglich sind, wie bestimmte Daten also weiterverarbeitet werden können. So lassen sich in Anlehnung an unser initiales Beispiel etwa 2 Integer ohne Probleme addieren, nicht jedoch ein Integer und ein String. Auch ein Integer und ein Float lassen sich addieren, aber man sollte sich überlegen, welchen Datentyp das Ergebnis dieser Berechnung aufweisen soll. Soll das Ergebnis der Summe des Integers 1 und des Floats 1,7 ein Float sein, lautet die Ausgabe 2,7. Soll das Ergebnis hingegen ein Integer sein, lautet die Ausgabe 2, da alle Dezimalstellen entfernt werden.

Um mit dem Computer über die Programmiersprache richtig kommunizieren zu können, sollte man also immer im Auge behalten, um welchen Datentyp es gerade geht. In der Praxis reichen oft die automatisierten Annahmen von Python zum Datentyp. Texteingaben werden von Python meist automatisch als String erkannt oder Ganzzahlen als Integer. Manchmal muss man Python jedoch konkret mitteilen, um welchen Datentyp es sich handelt.

Zusammenfassung

- Die zentralen Funktionseinheiten des Computers sind die **Central Processing Unit (CPU)** und die **Graphics Processing Unit (GPU).**
- Sie bestehen aus einer riesigen Anzahl an **Transistoren,** mikroskopisch kleinen Schaltwerken, durch die Strom fließen kann – oder nicht.
- Transistoren sind die physikalische Grundlage der **binären Maschinensprache,** die aus 0en und 1en besteht.
- **Höhere Programmiersprachen** wie Python, Java, oder C sind für den Menschen besser verständlich. Der **Compiler** übersetzt diese Programmiersprachen in Maschinensprache.
- **Python** hat sich als Programmiersprache für die KI durchgesetzt, u. a., da sie über eine einfache Syntax verfügt, plattformübergreifend nutzbar ist und eine Vielzahl **Bibliotheken** existieren, die vorgefertigte Code-Befehle enthalten, mit denen sich KI-Algorithmen einfacher programmieren lassen.

5.3 Python

5.3.1 Installation

Nutzer*innen-Umgebungen

Auch wenn die Programmiersprache immer dieselbe ist, gibt es verschiedene Nutzer*innen-Umgebungen für Python. Nahezu alle Nutzer*innen-Umgebungen kann man kostenlos im Internet herunterladen, um mit der Programmierung auf dem eigenen Computer zu starten.

Die Standard-Version von Python erhält man über folgenden Link: python.org/downloads (Stand: 01.07.2024)

In diesem Buch verwenden wir jedoch die Umgebung **Jupyter,** um zu programmieren, da sie eine übersichtlichere Anordnung der Code-Abschnitte in eigenen Bereichen ermöglicht: jupyter.org/install (Stand: 01.07.2024)

Alternativ ist auch **GoogleColab** sehr empfehlenswert, da sich der Code hier wie bei Jupyter einfach strukturieren lässt, viele nützliche Bibliotheken bereits hinterlegt sind, und die Programme im GoogleDrive gespeichert werden, sodass man diese einfach teilen bzw. rechnerunabhängig bearbeiten kann: colab.research.google.com (Stand: 01.07.2024)

Bibliotheken

Ist eine benötigte Bibliothek nicht bereits vorinstalliert, kann diese z. B. mit dem Befehl `pip install installiert` werden. Anschließend muss jede Bibliothek, egal ob vorinstalliert oder manuell installiert, noch **importiert** werden. Es reicht, dies einmal für einen Programmier-Rechner zu tun.
Mit dem folgenden Befehl kann die Bibliothek TensorFlow, die z. B. für die Programmierung neuronaler Netze

```
1 pip install tensorflow
```

Hiermit wird TensorFlow importiert:

```
1 import tensorflow as tf
```

5.3.2 Befehle

Variablen und Eingaben

In der Jupyter-Umgebung wird Code in grau hinterlegten Zellen eingegeben, und die Ausgabe direkt darunter angezeigt:

```
1 3 + 4
```

```
7
```

Python kann auch viele andere Berechnungen ausführen:

```
1 2 * 2
```

```
4
```

```
1 11.4 - 7.9
```

```
3.5
```

```
1 888/271
```

```
3.2767527675276753
```

Neben Zahlen können auch Worte, oder Zahlen und Worte ausgegeben werden. Hierzu verwendet man den Befehl `print()`. Aber Achtung – sollen ganz bestimmte Aussagen ausgegeben werden, müssen diese in Anführungszeichen stehen:

```
1 print("Ich lerne Python!")
```

```
Ich lerne Python!
```

```
1 print("Pi ist definiert als 3,14159...")
```

```
Pi ist definiert als 3,14159...
```

Ein wichtiges Konzept in der Programmierung – und in der Mathematik – sind **Variablen.** In Python lassen sich Variablen definieren und dann beliebig weiterverwenden. So können wir z. B. eine Variable x definieren und weitere Berechnungen mit ihr durchführen:

```
1 x = 5
2 x + 15 - 3
```

```
17
```

Python merkt sich Variablen, sodass man mit x auch in einer darauffolgenden Zelle rechnen kann. Möchte man den Zahlenwert der Variablen z anstatt den Buchstaben „z“ ausgeben, verwendet man keine Anführungszeichen:

```
1 y = 7.2
2 z = x + y
3
4 print(z)
```

```
12.2
```

Nach einem **#** oder zwischen jeweils **3 Anführungszeichen** lassen sich im Code **Kommentare** vermerken, die vom Computer nicht ausgeführt werden. Das ist gerade bei langem und komplexem Code sehr sinnvoll, um diesen näher zu erläutern und selbst den Überblick zu behalten:

```
# Das ist ein Kommentar
# Er wird nicht mit ausgeführt

"""
Das
hier
auch
"""

y = 7.2
z = x + y

print(z)
```

```
12.2
```

Mit dem `input()`-Befehl lassen sich Eingaben in Variablen speichern, so z. B. die Eingabe eines Namens in der Variable `name`. Auch hier möchten wir die Variable `name` ausgeben und nicht das Wort „name“, sodass wir im print-Befehl auf Anführungszeichen verzichten:

```
name = input("Wie heißt du? ")

print(name)
```

```
Wie heißt du? Tim
Tim
```

Durch Verwendung eines Kommas lassen sich innerhalb eines `print`-Befehls mehrere Ausgaben kombinieren:

```
name = input("Wie heißt du? ")
age = input("Wie alt bist du? ")

print("Du heißt", name, "und bist", age, "Jahre alt")
```

```
Wie heißt du? Tim
Wie alt bist du? 25
Du heißt Tim und bist 25 Jahre alt
```

Aufgabe 1: Frage nach dem Vornamen und dem Nachnamen einer Person und gib anschließend aus „Hallo, [Vorname] [Nachname]“.

Mit dem Befehl `len()` lässt sich die Länge einer Variable bestimmen:

```
name = input("Wie heißt du? ")

len(name)
```

```
Wie heißt du? Tim

3
```

Wie oben bereits erläutert, bestimmt die Art des Datentyps, welche weiteren Operationen mit einer Variable durchgeführt werden können. Je nach Befehl nutzt Python vorgefertigte Annahmen zu den Datentypen, die oft, jedoch nicht immer zutreffen. Betrachten wir dazu den folgenden Code, der zur Variable `age` die Zahl 1 addieren soll:

```
1 age = input("Wie alt bist du? ")
2 age = age + 1
3
4 print ("Nächsten Geburtstag wirst du", age)
```

```
Wie alt bist du? 25
TypeError: can only concatenate str (not "int") to str
```

Sobald wir die Zahl eingegeben haben, erhalten wir eine Fehlermeldung. Das liegt daran, dass `age` hier als String definiert wurde und mit Strings keine Berechnungen durchgeführt werden können. Daher müssen wir Python mitteilen, dass `age` in diesem Fall ein Integer (`int`) ist:

```
1 age = int(input("Wie alt bist du? "))
2 age = age + 1
3
4 print ("Nächsten Geburtstag wirst du", age)
```

```
Wie alt bist du? 26
Nächsten Geburtstag wirst du 27
```

Alternativ können wir auch definieren, dass es sich bei einer Variablen um eine Fließkommazahl (`float`) handelt:

```
1 centigrate = float(input("Nenne eine Temperatur in Grad Celsius "))
2 kelvin = centigrate - 273.15
3
4 print (centigrate, "Grad Celsius entspricht", kelvin, "Kelvin")
```

```
Nenne eine Temperatur in Grad Celsius 23.4
23.4 Grad Celsius entspricht -249.74999999999997 Kelvin
```

Aufgabe 2: Frage nach dem Preis des Abendessens. Dann frage, wie viele mitgegessen haben. Teile den Gesamtpreis durch die Zahl der Gäste und gib aus, wie viel jede Person zahlen muss. Tipp: Überlege hierzu, bei welcher Angabe Dezimalstellen Sinn ergeben (Datentyp: `float`) und bei welcher nur eine ganze Zahl (Datentyp: `int`).

Mathematik

Erste einfache Rechnungen haben wir bereits durchgeführt. Jetzt fügen wir unserem Repertoire noch einige Rechenoperatoren hinzu.
Zur Berechnung von **Potenzen** werden ** verwendet:

```
1 a = 10**2
2 b = 7**3
3 c = 10**-4
4
5 print(a, b, c)
```

```
100 343 0.0001
```

Aufgabe 3: Definiere eine beliebige quadratische Funktion. Frage nach einem x-Wert und gib den zugehörigen y-Wert aus.

Zur Berechnung von **Wurzeln** eignen sich selbstverständlich ebenfalls Potenzen. Alternativ können Wurzeln mit dem Befehl `math.sqrt()`. berechnet werden. Die Bibliothek math enthält zusätzliche Rechenbefehle und muss vorab mit dem Befehl `import` importiert werden:

```
100**0.5
```

```
10.0
```

```
import math

math.sqrt(100)
```

```
10.0
```

Mit der math-Bibliothek lassen sich auch vordefinierte Variablen wie π (pi) oder die Euler-Zahl e verwenden:

```
print("Pi:", math.pi)

print("e:", math.e)
```

```
Pi: 3.141592653589793
e: 2.718281828459045
```


Aufgabe 4: Frage nach dem Radius und der Höhe eines Zylinders. Berechne das Volumen des Zylinders mit der Formel *Volumen = (π · Radius2) · Höhe*. Gib das Volumen als ganzzahliges Ergebnis aus.

Mit dem `round()`-Befehl lassen sich Zahlen gemäß Angabe der Dezimalstellen runden:

```
print(round(math.pi, 2))

print(round(math.e, 5))

print(round(12.345, 0))
```

```
3.14
2.71828
12.0
```

Mit `//` lassen sich ganzzahlige Divisionen durchführen und mit `%` der Rest einer Division bestimmen:

```
num1 = int(input("Nenne eine erste Zahl: "))
num2 = int(input("Nenne eine zweite Zahl: "))

result = num1//num2
remainder = num1%num2

print(num1, "geteilt durch", num2, "ist", result, "Rest", remainder)
```

```
Nenne eine erste Zahl: 17
Nenne eine zweite Zahl: 4
17 geteilt durch 4 ist 4 Rest 1
```

Hier werden für die Medizin relevante KI-Modelle nachvollziehbar erklärt: Grundlagen und typische Anwendungsbereiche, mathematische Algorithmen, Daten und Datentypen, Programmierung in Python inkl. Programmcode sowie Erkennen und Vermeiden von Fehlern. Einsetzbar sowohl für wissenschaftliche Arbeiten als auch für KI-basierte Tools im medizinischen Alltag.
Teil I – Einführung: Was ist künstliche Intelligenz? Wo und wie kommt sie in der Medizin zum Einsatz? Welche Möglichkeiten und Grenzen bieten die Algorithmen? Welche Risiken gibt es, welche ethischen und rechtlichen Aspekte sind zu bedenken?
Teil II - Die wichtigsten Algorithmen werden detailliert vorgestellt
Für welche Art der Auswertung eignen sie sich? Welche mathematischen Modelle liegen zugrunde? Wie werden die Algorithmen programmiert und angepasst? Wie erkennt und vermeidet man fehlerhafte Auswertungen?

- Lineare Regression
- Logistische Regression
- Support Vector Machines
- Decision Trees
- Clustering
- Neuronale Netze
- Convolutional Neural Networks (CNNs)
- Graph Neural Networks (GNNs)
- Generative künstliche Intelligenz

Daneben werden die nötigen Grundlagen zu Daten, Datentypen und zur Programmierung in Python erläutert, der Python-Code ist als Download verfügbar. Dieses Werk eignet sich für alle, die medizinische Daten z.B. im Rahmen einer Promotionsarbeit selbst auswerten wollen oder die ein vertieftes Verständnis zur Anwendung von KI-Analysen in der Medizin anstreben, um bestehende Tools gezielt anwenden zu können oder um neue Anwendungen zu entwickeln: Medizinstudierende, aber auch Forschende, Ärztinnen und Ärzte oder Unternehmerinnen und Unternehmer.

Autor*innen
Tim Wiegand 2017-2024: Studium der Humanmedizin an der Ludwig-Maximilians-Universität (LMU) in München Seit 2019: Forschung und Promotion zu KI-Anwendungen in der Medizin und zu Neurotraumatologie an der LMU und der Harvard Medical School Seit 2020: Mitgründer von OneAIM („Artificial Intelligence in Medicine“), der deutschlandweit größten studentischen Arbeitsgruppe zu KI in der Medizin 2017-2022: Mitgründer des Lehr-Start-Ups erimed

Laura Velezmoro 2018-2025: Studium der Humanmedizin an der LMU in München Seit 2021: Forschung und Promotion zu KI-Anwendungen in der Strahlentherapie und zu molekularer Onkologie an der LMU Seit 2021: Teamleitung und Vorstandsmitglied bei OneAIM 2019-2023: Vorstandsmitglied der European University Alliance for Global Health (EUGLOH) und Mitglied der European Student Assembly

Künstliche Intelligenz in der Medizin: Anwendungen, Algorithmen und Programmierung

Wiegand, Tim (Autor),Velezmoro, Laura (Autorin)
2025. 192 Seiten., kt.
ISBN 9783437412080

ELSEVIER elsevier.de