CHRISTIAN DOPPLER GYMNASIUM

VORWISSENSCHAFTLICHE ARBEIT

Deep Learning – verschiedene Vorgehensweisen beim Training neuronaler Netze

Verfasserin:

Sarah Götz

Vorgelegt bei:

Dipl.Ing. Günther Kron

5020 Salzburg, Franz-Josef-Kai 41

Christian-Doppler-Gymnasium

Klasse 8A-BU

Schuljahr 2022/23

Februar 2023

Abstract

Sei es der Sprachassistent, die Gesichtserkennung des Smartphones oder der Chatbot – all das sind alltägliche Anwendungen von Künstlicher Intelligenz beziehungsweise einem Teilbereich dieser: dem „Deep Learning“. Deep Learning ist eine revolutionäre Technik, die auf dem Prinzip von künstlichen neuronalen Netzen basiert und sich in den letzten Jahren als eine vielversprechende Methode zur Lösung komplexer Probleme in verschiedenen Bereichen der Informatik etabliert hat.

In dieser vorwissenschaftlichen Arbeit liegt der Schwerpunkt auf der Erklärung der Funktionsweise von Deep Learning sowie der Vorgehensweisen, die beim Training von neuronalen Netzen zum Einsatz kommen. Außerdem werden die vielfältigen Anwendungsmöglichkeiten von Deep Learning und aufgrund der aktuellen rasanten Entwicklung dieser Technologien auch ein Blick in die Zukunft aufgezeigt. Zur Veranschaulichung des Deep Learnings wird auch ein eigens programmiertes neuronales Netz zur Erkennung handschriftlicher Ziffern vorgestellt.

Das Ergebnis der Arbeit zeigt, dass mit dem richtigen Training starke Deep Learning-Modelle erstellt und sinnvoll genutzt werden können, um komplexe Aufgaben zu lösen. Damit werden neue technologische Möglichkeiten eröffnet.

Inhaltsverzeichnis

1 Einleitung

Die rasante Entwicklung von Deep Learning in den letzten Jahren hat es zu einer der am meisten diskutierten und vielversprechenden Technologien der Informatik gemacht. Deep Learning ist eine Technik, die auf dem Prinzip von künstlichen neuronalen Netzen basiert und es ermöglicht, komplexe Probleme durch maschinelles Lernen zu lösen. Es wird in vielen Bereichen eingesetzt, von der Medizin bis hin zu selbstfahrenden Autos und hat das Potenzial, in Zukunft noch weiter an Bedeutung zu gewinnen.

In dieser vorwissenschaftlichen Arbeit steht die Auseinandersetzung mit der Funktionsweise von Deep Learning und der Vorgehensweisen beim Training von neuronalen Netzen im Vordergrund. Auch die vielfältigen Anwendungsmöglichkeiten sowie die Wichtigkeit von Deep Learning in zukünftigen Entwicklungen werden aufgezeigt.

Das Ziel dieser Arbeit ist es, einen Überblick über Deep Learning zu geben, vorzustellen wie die Erstellung eines neuronalen Netzes abläuft sowie auch aufzuzeigen, auf was man dabei achten muss, um nutzbare und sinnvolle Ergebnisse zu erzielen.

Diese Arbeit ist eine überwiegend literaturbasierte Arbeit. Zudem beinhaltet sie eine Programmiertätigkeit – ein, im Rahmen der vorwissenschaftlichen Arbeit, von mir programmiertes neuronales Netz zur Erkennung handschriftlicher Ziffern.

Der Arbeit vorangestellt sind die Definitionen wichtiger Begriffe, um nachfolgende Erklärungen und Beschreibungen des Deep Learnings nachvollziehen zu können. Darauf aufbauend werden die Vorgehensweise beim Training neuronaler Netze sowie die verschiedenen Konzepte und Techniken, die dabei zum Einsatz kommen, erläutert. Um das an einem praktischen Beispiel zu veranschaulichen, wird anschließend ein eigens programmiertes Neuronales Netz zur Erkennung handschriftlicher Ziffern vorgestellt. Abschließend werden verschiedene Anwendungen von Deep Learning beschrieben und es wird ein Blick in die Zukunft von Deep Learning und der Künstlichen Intelligenz geworfen.

2 Begriffsdefinitionen

Bevor neuronale Netze und die Funktionsweise sowie das Training dieser verstanden werden können, müssen die zugrundeliegenden Begriffe erklärt und verinnerlicht werden. Nur so können die folgenden Erklärungen nachvollzogen werden.

2.1 „Künstliche Intelligenz“

Der Oberbegriff „Künstliche Intelligenz“ bzw. abgekürzt „KI“ umfasst eine Vielzahl von Aufgabenbereichen. Bereiche, wie das Maschinelle Lernen oder Deep Learning, welche in folgenden Kapiteln erläutert werden, sind Teilgebiete der KI *[Abb. 1]*.[1]

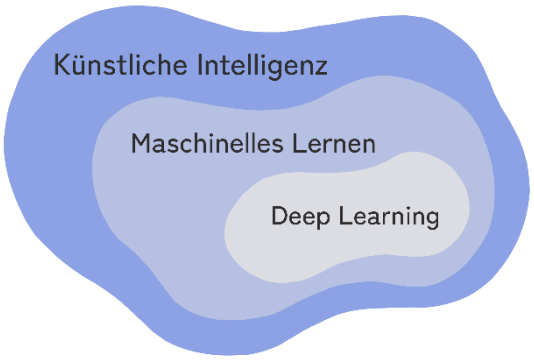


Abb. 1 - KI-Schema

„Künstliche Intelligenz“ exakt zu definieren erweist sich als schwierig und komplex, da bereits die Definition von „Intelligenz“ umfangreich und oftmals verschieden ist, je nachdem aus welcher Sicht man Intelligenz betrachtet – sei es aus psychologischer, technischer oder neurowissenschaftlicher Sicht. Bestimmte vorausgesetzte Fähigkeiten, um von Intelligenz sprechen zu können, wie Lernen, Verstehen, Schlussfolgern, Vorausschauen, Beurteilen oder Anpassen an Veränderungen in der Umwelt sind hierbei bei den meisten Definitionen kongruent.[2] Wenn „Intelligenz“ somit definiert ist, besitzt die Definition von „Künstlicher Intelligenz“ noch die Erweiterung, dass ein künstliches System – also ein Computer, Roboter, etc. – Entscheidungen trifft und Aktionen ausführt, für die „Intelligenz“ benötigt wird.[3] Das bedeutet, ein intelligentes System ist in der Lage, mit neuen Informationen zu arbeiten und diese somit aufzunehmen, zu verarbeiten, auszuwerten, zu verstehen und im Anschluss Zusammenhänge zwischen Daten zu erkennen. Durch diese Analyse kann ein solches System vorausschauend urteilen und handeln. Nichtsdestotrotz „versteht“ ein Computer Informationen nicht in der

[1] Vgl. Mueller et al. (2020), S. 30
[2] Vgl. Mueller et al. (2020), S. 30f; vgl. Alpaydın. (2008), S. 2; vgl. Goerzel et al. (2007), S. 17-23
[3] Vgl. Kersting et al. (2019), S. 6

gleichen Art und Weise, wie Menschen es tun – für einen Computer sind Daten nur Zahlenwerte, die er rein mathematisch auswerten kann. Er hat schließlich weder Gedanken noch ein Bewusstsein, sondern er führt rein logische, mathematische Berechnungen aus (sh. Info „Matrizen“).[4] Daher ist eine menschliche Interpretation der Daten für die korrekte Einordnung dieser zu Beginn des Prozesses sowie der herausgegebenen Ergebnisse erforderlich.[5]

KI wird für alltägliche Aufgaben genutzt – Beispiele dafür sind selbstfahrende Autos, Sprachassistenten, angepasste Werbungen, Spam-Mail-Erkennung, Schach- und weitere Spieleroboter oder die einfache Erkennung und Übersetzung von Sprachen.

> *INFO - Matrizen:*
>
> Da ein Computer Daten nicht versteht, müssen diese in abgewandelter Form eingegeben werden, mit denen der Computer dann logische Berechnungen durchführen kann. Die meistgenutzte Methode ist dabei die Matrix. Eine Matrix *[sh. Abb. 2]* ist eine tabellarische Anordnung von Elementen bestehend aus mehreren Zeilen und Spalten. Jedes dieser Elemente ist durch einen Doppelindex aus einem Index für die Zeile und einem für die Spalte gekennzeichnet, wodurch die Position bestimmt ist. Mit diesen Matrizen kann ein Computer nun Berechnungen durchführen und die Daten verarbeiten.[6]

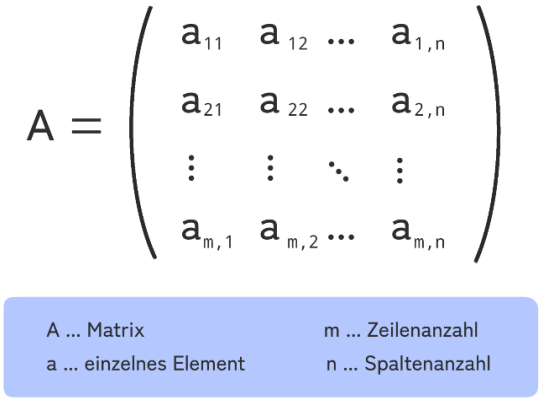


Abb. 2 - Matrix

[4] Vgl. Mueller et al. (2020), S. 30-32
[5] Vgl. Mueller et al. (2020), S. 117
[6] Vgl. Matrix. (o.D.)

2.2 „Maschinelles Lernen“

Maschinelles Lernen (engl. Machine Learning, ML) ist ein wesentlicher Teilbereich der Künstlichen Intelligenz, der anders als herkömmliche Programme funktioniert. Beim Maschinellem Lernen wird Computerprogrammen beigebracht, wie diese durch eigene Erfahrungen mit gegebenen Daten lernen können, anstatt bereits das gewünschte Verhalten fest einzuprogrammieren. Das ist vor allem von Vorteil, wenn ein Programm mit vielen verschiedenen und ständig neuen Daten arbeiten muss – hierbei muss nicht manuell für jede Veränderung ein neuer Code geschrieben werden, sondern ein Programm lernt mithilfe von sich anpassenden Algorithmen (sh. Info „Algorithmus“) und riesigen Datenmengen (sh. Kapitel „Daten“), welche Merkmale bei den Daten entscheidend sind und wie er diese identifizieren kann.[7]

Ein Beispiel, das diesen Vorteil veranschaulicht, ist die Bilderkennung von handgeschriebenen Zahlen. Da die Handschrift jeder Person individuell ist, muss das Programm lernen und verstehen, welche Merkmale die jeweiligen Ziffern haben (sh. Kapitel „Programmierung eines neuronalen Netzes“).

Auch das Lösen komplexerer Aufgaben dieser Art, wie z.B. die Identifikation von Tumoren auf Röntgenbildern, sind durch Maschinelles Lernen möglich, solange ein Zusammenhang zwischen Ein- und Ausgabedaten besteht.[8]

> *INFO - Algorithmus:*
>
> Bei intelligenten Systemen werden Daten verarbeitet, neue Daten erfasst, diese werden wieder verarbeitet und der Vorgang wiederholt sich, bis das zuvor festgelegte Ziel erreicht wird. Um sich diesem Ziel anzunähern, gibt es Algorithmen, die Eingabedaten (den Input) nach einer eindeutigen Handlungsvorschrift (= Reihe von abzuarbeitenden Operatoren) bearbeiten, sodass sie in die gewünschte Ausgabe (den Output) umgewandelt werden *[sh. Abb. 3]*.[9] Zum besseren Verständnis kann ein Algorithmus mit einem Kochrezept verglichen werden: Die Eingabedaten (Zutaten) werden nach einer Handlungsvorschrift (Zubereitung) verarbeitet, um eine gewisse Ausgabe (fertiges Essen) zu erzielen.[10]

[7] Vgl. Kersting et al. (2019), S. 7; vgl. Stadler. (2022, 6. September)
[8] Vgl. Kersting et al. (2019), S. 7
[9] Vgl. Mueller et al. (2020), S. 32, 46; vgl. Kersting et al. (2019), S. 11
[10] Vgl. Kersting et al. (2019), S. 12f

Abb. 3 - Algorithmus

Beim Maschinellen Lernen besteht diese Handlungsvorschrift jedoch nicht aus einer genauen Vorgabe, wie ein Problem zu lösen ist, sondern wie aus Erfahrungen gelernt werden kann, um Probleme besser zu lösen – ein sogenannter „Lernalgorithmus“. Ein Lernalgorithmus muss, bevor er auf eine Problemstellung angewendet werden kann, trainiert werden. Dabei wird zuerst ein sog. Modell anhand von Trainingsdaten (sh. Info „Daten“) erstellt, angepasst und optimiert (Trainingsphase). Dieses Modell beschreibt den Zusammenhang zwischen den Eingaben und Ausgaben und kann in seiner einfachsten Form eine kurze Formel wie z.B. „f(x)=3x+2“ sein, wodurch der Algorithmus nun in der Lage ist, für jegliche Eingabedaten die korrekte Ausgabe zu berechnen. Nachdem ein Modell trainiert wurde, kann in der Test- bzw. Anwendungsphase durch Testdaten die Funktionalität überprüft werden, um dann schlussendlich auf neue, unbekannte Datensätze angewendet werden zu können.[11]

2.2.1 Daten

Daten spielen beim Maschinellen Lernen eine sehr große Rolle, da die genutzten selbstständig lernenden Algorithmen nur mit einem großen Datensatz wie gewollt funktionieren und Tendenzen und Korrelationen – mit der Sicherheit, dass es kein Zufall ist – erkennen können.[12] Dennoch ist ein größerer Datensatz nicht immer gleichzusetzen mit einem besseren ML-Modell, da man hierbei auch auf Probleme wie einer Überanpassung stoßen kann (sh. Kapitel „Schwächen“). Außerdem wichtig für ein aussagekräftiges Ergebnis und die korrekte Interpretation der Daten ist die Datenaufbereitung: Vor der Nutzung von Daten, müssen diese auf Messfehler überprüft und vorbereitet werden, um die Vollständigkeit und Kompatibilität dieser sicherzustellen.[13] Auch die Qualität der Daten hinsichtlich ihrer

[11] Vgl. Kersting et al. (2019), S. 19; vgl. Mueller et al. (2020), S. 46
[12] Vgl. Kersting et al. (2019), S. 29, 142f
[13] Vgl. Kersting et al. (2019), S. 36

Diversität ist ausschlaggebend, um ein einseitiges und dadurch nicht allgemein gültiges Ergebnis auszuschließen.[14]

Einteilung der Daten:

Um mit den Daten ein Modell eines Algorithmus zu trainieren, zu testen, etc. müssen die gesammelten Daten eingeteilt werden in „Trainingsdaten“ und „Testdaten“ und – bei ausreichender Menge an Daten – zusätzlich noch in „Validierungsdaten“ *[sh. Abb. 4 – beispielhafte Verteilung]*. Mithilfe der Trainingsdaten lernt der Algorithmus Muster und Zusammenhänge der Daten, wodurch der Algorithmus angepasst wird, sodass am Ende auch unbekannte Daten korrekt identifiziert und bewertet werden können. Während dieses Prozesses des Trainings wird der Algorithmus so lange angepasst, bis der Fehler minimiert ist und die Trainingsdaten korrekt und sinnvoll verarbeitet sind. Sofern der Datensatz groß genug ist, werden vom Trainingsdatensatz die Validierungsdaten abgespalten – ein weiterer Beispielsdatensatz. Dieser wird genutzt, um den Algorithmus zu optimieren und eine Überanpassung auf die Trainingsdaten (sh. Kapitel „Schwächen“) auszuschließen. Der letzte Schritt ist schließlich das Testen des Modells durch die Testdaten, die für den Algorithmus unbekannt und somit unabhängig von den Trainings- und Validierungsdaten sein müssen. Hierbei wird überprüft, wie gut das trainierte Modell funktioniert. Am Ende dieses Prozesses ist der Algorithmus bzw. das gelernte Modell bereit dazu auf neue Daten angewendet zu werden.[15]

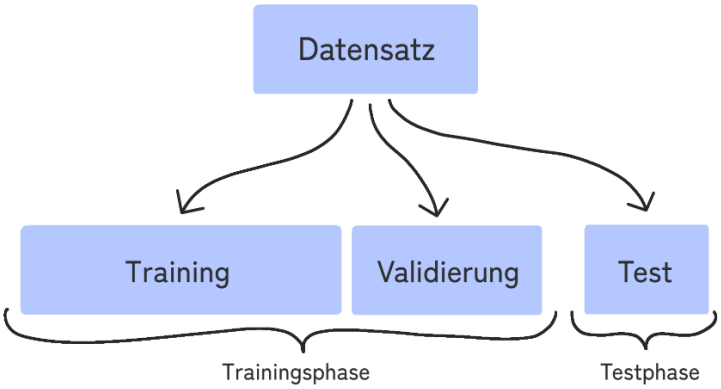


Abb. 4 - Dateneinteilung

[14] Vgl. Kersting et al. (2019), S. 147
[15] Vgl. Mueller et al. (2020), S. 49f; vgl. Wuttke. (2021, 19. Oktober)

Wie genau die Daten ausgewertet werden und welche Lernstrategien wann benutzt werden, hängt von dem Anwendungsfall ab – im Kapitel „verschiedene Konzepte zum Trainieren neuronaler Netze“ werden die bekanntesten Ansätze beschrieben.[16]

2.3 „Deep Learning“

Ein Teilgebiet des Maschinellen Lernens ist das Deep Learning (DL). Es funktioniert prinzipiell wie jedes ML-Modell, hat aber einen grundlegenden Unterschied bezüglich der Komplexität: Ein Deep Learning Netz ist für besonders intensive und komplexe Analysen ausgelegt. Grundsätzlich werden beim Maschinellen Lernen sogenannte „neuronale Netze“ (sh. Kapitel „Neuronale Netze“) genutzt, welche je nach Anwendung einen unterschiedlichen Aufbau haben. Beim Deep Learning wird eine Weiterentwicklung dieser neuronalen Netze – „Tiefe neuronale Netze“ (engl. Deep Neural Networks, DNN) mit mehreren Schichten von Neuronen – genutzt *[sh. Abb. 5 – Beispiel für dein Tiefes neuronales Netz]*. Durch diese Architektur können spezielle Lernaufgaben wie die Spracherkennung, Bilderkennung, Textauswertung oder das Spielen von „Schach“ oder „Go“ bewältigt bzw. verbessert werden. Bei all diesen Aufgabenfeldern müssen Muster und Zusammenhänge in großen Datenmengen erkannt und verarbeitet werden.[17]

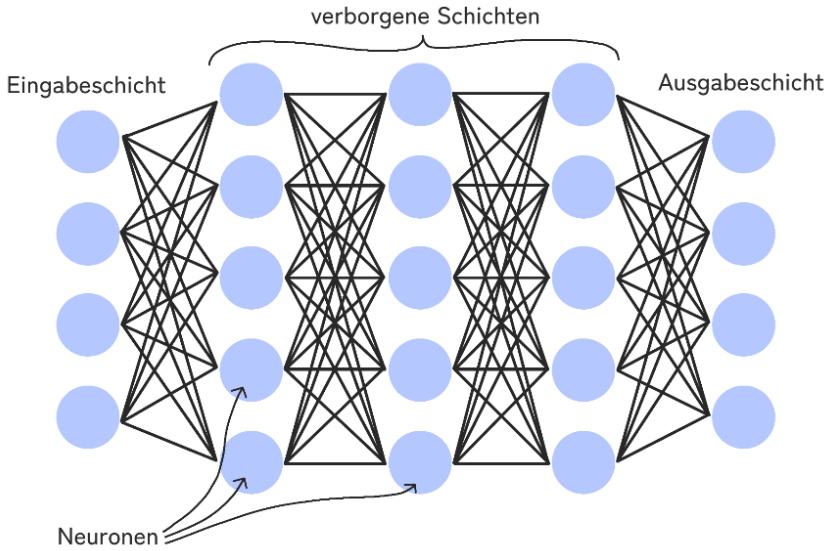


Abb. 5 - DNN

[16] Vgl. Mueller et al. (2020), S. 47
[17] Vgl. Mueller et al. (2020), S. 29, 37-40, 58

2.4 „Neuronale Netze“

Künstliche neuronale Netze (KNN) sind für einige Bereiche des Maschinellen Lernens, sowie auch für das Deep Learning, essenziell. Sie werden ergänzend zu herkömmlichen Computertechnologien angewendet, weil sie den Vorteil haben, auch bei unvollständigen und verrauschten Daten einsetzbar zu sein und komplexe Aufgaben lösen zu können. Anwendungsbereiche sind Bild-/ Mustererkennung (z.B. Erkennung von Texten, Unterschriften oder Gesichtern), die Unterstützung bei Entscheidungen (z.B. Kreditvergabe), Finanzanalysen, medizinische Diagnosen, die autonome Steuerung von Fahrzeugen und Robotern, etc.[18]

2.4.1 Biologischer Hintergrund

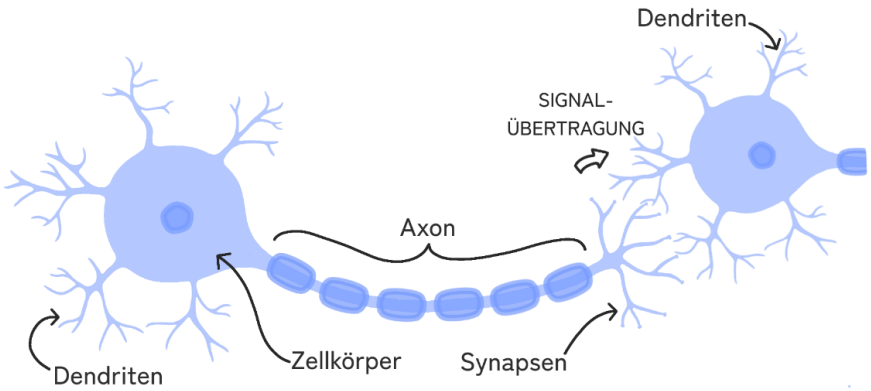


Abb. 6 - Nervenzelle

Die Grundidee hinter neuronalen Netzen stammt von dem biologischen Vorbild des menschlichen Gehirns bzw. genauer gesagt der Gehirnfunktion. Das Denken und intelligente Verarbeiten von Informationen laufen im Gehirn über Nervenzellen bzw. Neuronen. Eine Nervenzelle setzt sich im Grundaufbau aus Dendriten, dem Zellkörper, dem Axon und den Synapsen zusammen *[sh. Abb. 6]*. Übertragen auf künstliche neuronale Netze stehen die Dendriten für die Eingabe, da sie die Informationen in Form von elektrischen Signalen empfangen. Anschließend werden die Signale verarbeitet und aufsummiert. Wird dabei ein gewisser Schwellenwert überschritten, wird vom Zellkörper ein elektrischer Impuls – was die Ausgabe der Daten widerspiegelt – entlang des Axons abgefeuert und

[18] Vgl. Kaffka. (2017), S. 22f, 203

schließlich über die anliegenden Synapsen an anknüpfende Neuronen weitergegeben.[19]

2.4.2 Aufbau

Ein künstliches neuronales Netz besteht aus mehreren Neuronen (= Recheneinheiten), die in hierarchischen Schichten angeordnet sind. In der ersten Schicht (Eingabeschicht), befinden sich die Neuronen, die die Eingabedaten aufnehmen und verarbeiten. Die letzte Schicht (Ausgabeschicht), besteht aus Neuronen, die am Ende des Prozesses die Ausgabewerte liefern. Dazwischen ist mindestens eine sog. verborgene Schicht mit beliebig vielen Neuronen, die anwendungsabhängige Operationen durchführen – wie bereits im Kapitel „Deep Learning“ gesehen, werden meist mehrere dieser verborgenen Schichten hintereinander angeordnet.[20] Diese verborgenen Schichten werden auch oft als „Black-Box“ bezeichnet, da man nicht genau erklären kann, wie das neuronale Netz schlussendlich zu dem gewünschten Ergebnis kommt.[21] Um die Signale auch weitergeben zu können, ist jedes dieser Neuronen mit dem vorherigen und nachfolgenden Neuron verknüpft, um so vom Vorgänger die Eingaben anzunehmen und die Ausgabe an den Nachfolger weiterzugeben. Diese Verbindungen sind verschieden stark gewichtet, um am Ende ein korrektes Ergebnis zu erhalten. Eine kleine Gewichtung bewirkt dabei einen kleinen Einfluss auf das endgültige Ergebnis, eine große – sowohl positive als auch negative – Gewichtung bewirkt eine starke Beeinflussung *[sh. Abb. 5 in Kapitel „Deep Learning“]*.[22]

2.4.3 Netzarchitekturen

Je nach Anwendungsfall, gibt es auch eine große Anzahl verschiedener Netzarchitekturen, die für bestimmte Aufgaben optimiert wurden. Beispielsweise ist für die Spracherkennung eine andere Struktur und Topologie nötig, als für die Bilderkennung. Im Kapitel „Deep Learning“ ist bereits eine Art des Aufbaus eines neuronalen Netzes dargestellt. Folgend ein Einblick in weitere häufig genutzte Architekturen:[23]

[19] Vgl. Kaffka. (2017), S. 22, 25f; vgl. Kersting et al. (2019), S. 151f
[20] Vgl. Kersting et al. (2019), S. 159f
[21] Vgl. Kaffka. (2017), S. 203
[22] Vgl. Mueller et al. (2020), S. 165, 175
[23] Vgl. Brunton. (2019, 6. Juni), 1:57-4:13

Feedforward Neural Network (FNN) *[sh. Abb. 7]:*

Unter FNNs versteht man vorwärtsgerichtete Netze, also jene Netze, die Informationen von der Eingabeschicht über beliebig viele Neronen direkt zur Ausgabeschicht leiten. Diese Netzarchitektur wird vorranging bei Regressions- und Klassifikationsproblemen des überwachten Lernens (sh. Kapitel „Überwachtes Lernen“) verwendet.[24]

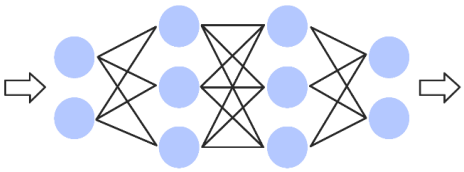


Abb. 7 - FNN

Recurrent Neural Network (RNN) *[sh. Abb. 8]:*

RNNs sind rückgekoppelte neuronale Netze und werden häufig für Sprach- und Audioerkennung sowie Text- und Videoerkennung genutzt – also jene Aufgabenfelder, bei den die geordnete und chronologische Abfolge der Daten von großer Wichtigkeit ist. Hierbei wird mit sogenannten „Feedback-Loops“ gearbeitet, die in der Lage sind Informationen entweder erneut zu sich selbst oder beliebig viele Schichten zurück zu leiten. Damit können sie sich Informationen „merken“ und besitzen eine Art Gedächtnis. Das Signal wird somit nicht wie bei FNNs nur in eine Richtung von Anfang bis Ende transportiert, sondern immer wieder zurück geleitet. Dadurch ist die Ausgabe neben den Eingabedaten auch von den vorhergegangenen Berechnungen abhängig.[25]

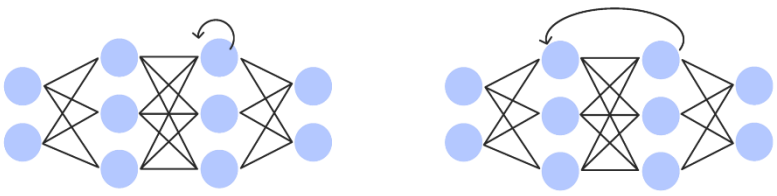


Abb. 8 - RNN

[24] Vgl. Oppermann. (2021, 31. August)
[25] Vgl. Brunton. (2019, 6. Juni), 5:33-6:36; vgl. Oppermann. (2021, 31. August)

Convolutional Neural Network (CNN):

Diese Art der neuronalen Netze wird vor allem für die Bilderkennung genutzt und arbeitet mit sogenannten Konvolutionen bzw. Faltungen. Dafür legt der Computer eine Art Filter über das Bild und überprüft die Übereinstimmung des Filters mit einem Bildausschnitt, um bei hoher Übereinstimmung Formen und Muster zu erkennen. Zuerst wird das Bild dabei auf einfache, grundlegende Formen untersucht, weiters dann auf komplexere, kombinierte Formen, um schlussendlich zusammengesetzte Formen und damit charakteristische Merkmale (z.B. Ohren oder Schnauze bei Hunden) zu erkennen.[26]
Der Vorteil dieser Faltungs-/ Konvolutionsnetze ist, dass sie unabhängig von der Position, Rotation oder dem Blickwinkel eines Objekts, dieses identifizieren können.[27]

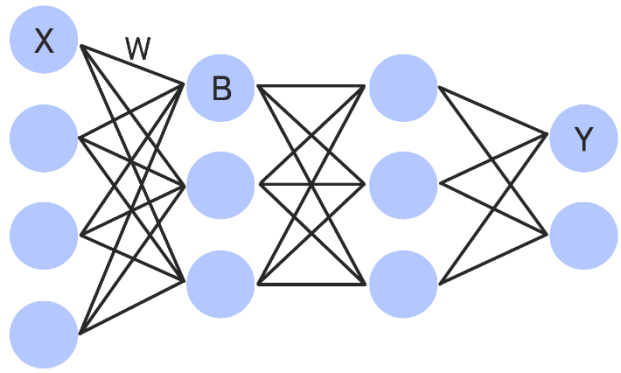
3 Arbeiten mit neuronalen Netzen

3.1 Grundsätzliche Vorgehensweise beim Training von neuronalen Netzen

Die Informationen bzw. Daten sind in numerischer Form in den Neuronen der Eingabeschicht repräsentiert. Diese numerische Darstellung ist essenziell, da ein neuronales Netz mit qualitativen Variablen wie beispielsweise Farbnennungen nicht umgehen und kein Ergebnis liefern kann *[sh. Abb. 9 (1)]*.[28] Diese Werte werden anschließend mit den Gewichten der jeweiligen Verbindungen multipliziert und folglich in den Neuronen der ersten verborgenen Schicht aufsummiert *[sh. Abb. 9 (2)]*. Die Neuronen der verborgenen Schichten sind auch wiederum mit einem Wert assoziiert – dem sogenannten „Bias“ (eine Konstante) – welcher ebenfalls addiert wird *[sh. Abb. 9 (3)]*. Die allgemeine Gleichung setzt sich somit wie folgt zusammen: $y = X \beta + \alpha$, wenn y die abhängige Variabel (Ausgabe), X die Eingabedaten als Matrix, β die Gewichtung und α den Bias darstellt.[29] Der berechnete Wert dieser Gleichung wird mithilfe einer Aktivierungsfunktion ausgewertet *[sh. Abb. 9 (4)]*. Je nachdem, ob der dabei resultierende Wert einen be-

[26] Vgl. Mueller et al. (2020), S. 208, 220; vgl. Kersting et al. (2019), S. 165-167
[27] Vgl. Brunton. (2019, 6. Juni), 4:29-5:32
[28] Vgl. Mueller et al. (2020), S. 163
[29] Vgl. Mueller et al. (2020), S. 135

stimmten Schwellenwert überschreitet, aktiviert sich das Neuron und verstärkt das Signal oder schwächt oder gar verstummt dieses (sh. Kapitel „Aktivierungsfunktion“). Wenn das Neuron aktiviert wurde, wird die Information auf die nächste Schicht übertragen, dabei wieder mit den Gewichten der Verbindungen multipliziert, summiert und durch Aktivierungsfunktionen ausgewertet. Mittels dieses Vorgangs werden die Daten durch das Netz geleitet und die Ausgabeneuronen nehmen einen bestimmten Wert an. Letztendlich zeigt das Neuron mit dem höchsten Wert (vergleichbar mit einer hohen Wahrscheinlichkeit) am Ende das prognostizierte Ergebnis *[sh. Abb. 9 (5)]*. Diese Vorgehensweise nennt man „Forward Propagation“ bzw. „Vorwärtspropagation“ – die Information wird vorwärts von Eingabe- zu Ausgabeneuronen durch das Netz geschleust.[30]



Berechnung Bsp. (Ausschnitt): X * W + B → Aktivierungsfunktion

(1) X ... Eingabedaten
(2) W ... Gewichte
(3) B ... Bias
(4) Aktivierungsfunktion
(5) Y ... Ausgabedaten/Prognose

Abb. 9 - Training eines NN (Ablauf)

Da zu Beginn die Gewichte und Bias-Werte zufällig vergeben werden, ist das vom neuronalen Netz vorhergesagte Ergebnis erst einmal vollkommen zufällig und somit meist fehlerhaft. Damit das neuronale Netzwerk die Fehler beheben kann und lernt, wird die prognostizierte Ausgabe mit der korrekten Ausgabe verglichen und so der Fehler und die Stärke von diesem ermittelt – für diese Repräsentation des Fehlers wird die *Verlustfunktion* genutzt (sh. Kapitel „Verlustfunk-

[30] Vgl. Mueller et al. (2020), S. 161; vgl. Simplilearn. (2019, 19. Juni), 1:06-2:26

tion“). Mit diesem berechneten Fehler kann das Netz, durch die sogenannte *Backpropagation/ Rückwärtspropagierung* (sh. Kapitel „Backpropagation“), den Fehler minimieren, indem die Information über den Fehler rückwärts – also von Ausgabe- zu Eingabedaten – durch das Netz geleitet wird und dadurch die Gewichte und Werte angepasst werden können. Das jeweilige Minimum des Fehlers bzw. der Verlustfunktion und damit die optimalen Parameter eines Modells können mit dem *Gradientenabstiegsverfahren* ermittelt werden (sh. Kapitel „Gradientenabstiegsverfahren“). Dieser Zyklus von Forward- und Backpropagation wird mehrfach mit vielen verschiedenen Eingabedaten wiederholt, bis das künstliche neuronale Netz so angepasst ist, dass es die richtigen Ergebnisse vorhersagen kann und somit der Fehler auf ein Minimum reduziert wurde.[31]

3.1.2 Aktivierungsfunktion

Aktivierungsfunktionen sind für die Transformierung bzw. Auswertung des zuvor berechneten Wertes zuständig. Das Grundprinzip dahinter ist das gleiche, wie bei einer herkömmlichen Funktion: x-Werte werden in die Aktivierungsfunktion eingesetzt und je nach Verlauf der Funktion wird so der korrelierende y-Wert bestimmt.[32] Der Nutzen dahinter ist eine nicht-lineare Beziehung zwischen Eingabe- und Ausgabedaten herzustellen bzw. zu berechnen und so komplexe Zusammenhänge mathematisch darstellen zu können. Besonders beim Deep Learning mit Netzen mit mehreren Schichten ist eine Aktivierungsfunktion unumgänglich.[33] Durch diese Nichtlinearität kann solch eine Aktivierungsfunktion bedeutsame Signale erkennen und verstärken und schwache oder verrauschte Signale, die sich unter einem bestimmten Schwellenwert befinden, unterdrücken.[34]

Je nach neuronalem Netz und Aufgabenbereich werden unterschiedliche Arten von Aktivierungsfunktionen benötigt. Folgend die meistgenutzten Aktivierungsfunktionen beim Deep Learning:

[31] Vgl. Simplilearn. (2019, 19. Juni), 2:26-3:30
[32] Vgl. Mueller et al. (2020), S. 161
[33] Vgl. Oppermann. (2021, 2. August)
[34] Vgl. Mueller et al. (2020), S. 167

Sigmoidfunktion [sh. Abb. 10]:

Die Sigmoidfunktion (math. Def.: $f(x) = \frac{1}{1+e^{-x}}$) stellt einen s-förmigen Kurvenverlauf mit einem Wertebereich zwischen null und eins dar und ist somit klar begrenzt. Sie transformiert daher die Eingabewerte in Ausgabewerte zwischen null und eins.

Bei tiefen Netzwerken mit vielen Schichten – wie es beim Deep Learning der Fall ist – zeigt sich jedoch die Schwäche der Sigmoidfunktion, da hierbei das Problem der sogenannten „schwindenden Gradienten“ vorliegt. Wie in *Abb. 10* zu sehen, ist die Steigung und somit die Ableitung der Funktion bei großen negativen und positiven Eingabewerten sehr gering. Diese verschwindend kleine Ableitung führt zu einer zu geringen Steigung der Verlustfunktion, was wiederum eine verschlechterte oder gar unmögliche Anpassung der Gewichte bei der Backpropagation und somit einen ausbleibenden Lernprozess zur Folge hat.[35]

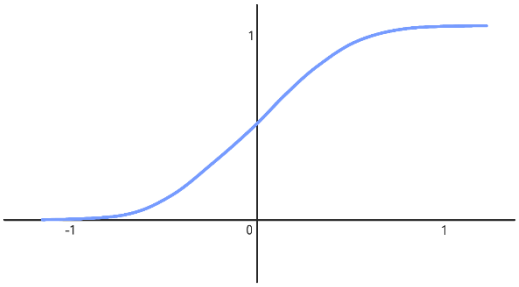


Abb. 10130 - Sigmoidfunktion

Tanh-Funktion [sh. Abb. 11]:

Die Tangens-Hyperbolicus Funktion (math. Def.: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$) ähnelt der Sigmoidfunktion in ihrer Form, kann aber Ausgaben im Intervall von minus eins bis eins liefern.

Dadurch, dass die tanh-Funktion nur eine skalierte Version der Sigmoidfunktion ist, liegt auch hier das Problem der „schwindenden Gradienten“ bei hohen negativen bzw. positiven Eingabewerten vor. Ein Vorteil gegenüber der Sigmoidfunktion ist die Nullzentriertheit und damit auch die Fähigkeit der negativen Beeinflussung.[36]

[35] Vgl. Mueller et al. (2020), S. 184; vgl. Oppermann. (2021, 2. August)
[36] Vgl. Mueller et al. (2020), S. 184; vgl. Oppermann. (2021, 2. August)

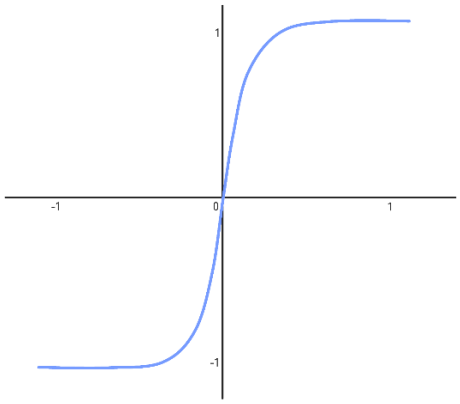


Abb. 147 - tanh-Funktion

ReLU-Funktion [sh. Abb. 12]:

Die Rectified Linear Unit (math. Def.: $f(x) = \max(0, x)$ *bzw.* $f(x) = \{^{0 \text{ für } x<0}_{x \text{ für } x\geq 0}$) ist die etablierteste und gebräuchlichste Aktivierungsfunktion heutzutage. Der Wertebereich liegt hier zwischen null und unendlich. Das Besondere an der Funktion ist, dass das Signal bei Werten unter null gestoppt wird und die Neuronen gar nicht erst aktiviert werden, wodurch Rechenleistung eingespart wird. [37] Für alle Eingaben die größer als null sind und den Schwellenwert somit überschreiten, gilt ein linearer Zusammenhang.

Die Schwäche dieses Funktionstyps sind sogenannte mögliche „tote Neuronen“. Es besteht nämlich die Möglichkeit, dass bei einer zu hohen Lernrate (sh. Kapitel „Gradientenabstieg“) die Gewichte stets so aktualisiert werden, dass einige Neuronen aufgrund von Eingaben x < 0 immer unter null liegen, dementsprechend nicht aktiviert werden und nach dem Training somit „tot“ sind.[38]

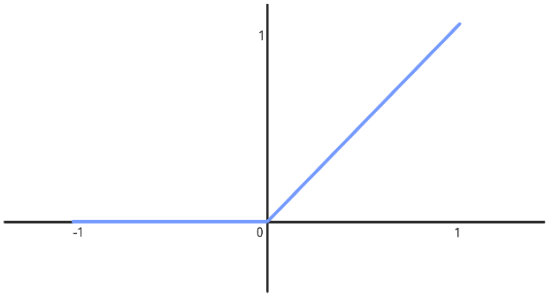


Abb. 12164 - ReLU-Funktion

[37] Vgl. Mueller et al. (2020), S. 198
[38] Vgl. Mueller et al. (2020), S. 184; vgl. Oppermann. (2021, 2. August)

Leaky ReLU [sh. Abb. 13]:

Die Leaky ReLU (math. Def.: $f(x) = \{^{ax\ f\ddot{u}r\ x<0}_{x\ f\ddot{u}r\ x\geq 0}$) ist eine verbesserte, weiterentwickelte Version der ursprünglichen ReLU-Funktion, jedoch ohne den Nachteil der „toten Neuronen“. Die Änderung bei dieser Variante ist, wie in *Abb. 13* zu erkennen, der Abschnitt für alle Eingaben x < 0, der hier durch eine Linie mit positiver Steigung dargestellt wird.[39]

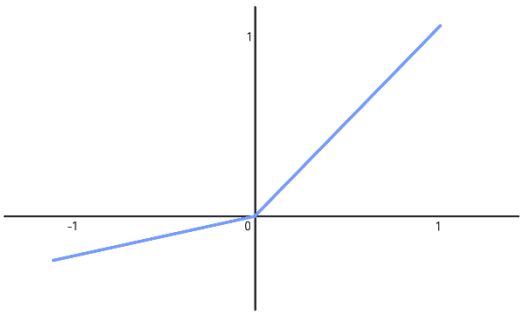


Abb. 13 - Leaky ReLU-Funktion

Softmax-Funktion:

Die Softmax-Funktion wird ausschließlich in der Ausgabeschicht verwendet und dient bei Klassifizierungsaufgaben dem Vorhersagen von Wahrscheinlichkeiten. Diese Aktivierungsfunktion bewirkt, dass alle Werte der Ausgabeneuronen in den Bereich zwischen null und eins gebracht werden und all diese Ausgabewerte in Summe eins ergeben.[40]

3.1.3 Verlustfunktion

Die Verlustfunktion (auch Kostenfunktion genannt), repräsentiert die Differenz zwischen korrektem und vom Netzwerk prognostiziertem Ergebnis (=Kosten) und dient dem Computer zur Erkennung seiner Leistung und Zuverlässigkeit – vereinfacht gesagt, ob die getroffenen Vorhersagen korrekt oder falsch waren *[sh. Abb. 14]*.[41]

[39] Vgl. Mueller et al. (2020), S. 184; vgl. Oppermann. (2021, 2. August)
[40] Vgl. Oppermann. (2021, 2. August)
[41] Vgl. Mueller et al. (2020), S. 129

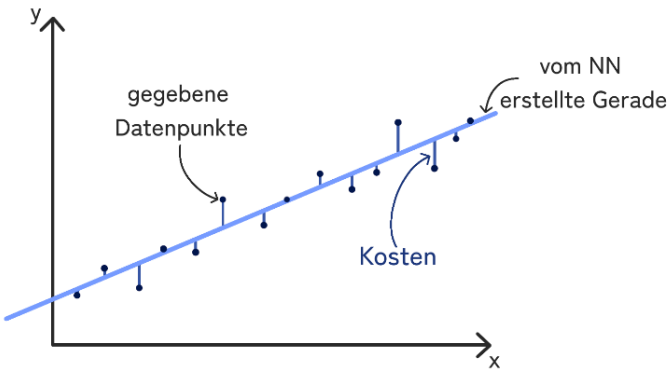


Abb. 14 - Verlustfunktion

Eine oft genutzte Verlustfunktion ist der mittlere quadratische Fehler. Ermittelt wird dabei die Summe der quadrierten Differenzen zwischen korrektem Wert und Vorhersage. Das Quadrieren trägt zur deutlicheren Sichtbarkeit von schwerwiegenden Fehlern bei und sorgt dafür, dass die Richtung der Abweichung (ob nach oben oder unten) einflusslos ist. Mit diesem berechneten Wert ist der Algorithmus folglich dazu in der Lage, Fehler zu minimieren und durch eine kleine Anpassung bei nahezu richtigen Ausgaben bzw. einer großen Anpassung bei gravierenden Fehlausgaben immer bessere Ergebnisse zu erzielen.[42]

3.1.4 Backpropagation

Backpropagation bzw. Rückwärtspropagierung ist der bekannteste Algorithmus zur Anpassung der Parameter in neuronalen Netzen und ist besonders bei tiefen neuronalen Netzen essenziell, da die Abhängigkeit und gegenseitige Beeinflussung der vielen Schichten untereinander, die Aktualisierung der Gewichte und Bias-Werte erschwert. Die Anpassung läuft hier in umgekehrter Richtung ab – der Fehler wird von der Ausgabeschicht aus zurückpropagiert, um die dafür verantwortlichen Neuronen ausfindig zu machen und schließlich den Fehler durch Justierungen der Parameter zu beheben.[43]

3.1.5 Gradientenabstieg

Das Gradientenabstiegsverfahren dient bei der Backpropagation dem Aufsuchen des Minimums der Verlustfunktion, um durch optimal angepasste Parameter den kleinstmöglichen Fehlerwert zu erzielen. Dafür werden die Ergebnisse der Ver-

[42] Vgl. Mueller et al. (2020), S. 129f; vgl. Kersting et al. (2019), S. 67, 156
[43] Vgl. Mueller et al. (2020), S. 168

lustfunktionen verschiedener möglicher Modelle mit jeweils abgeänderten Parametern ausgewertet und verglichen, bis die idealen Parameter ermittelt werden konnten.[44]

Eine einfache Fehlerkurve wie in *Abb. 15* zeigt, wie groß der berechnete Fehlerwert bei unterschiedlichen Parametereinstellungen ist. Um hierbei das Minimum zu finden, wird die erste Ableitung (zeigt die Steigung einer Funktion an einem gewissen Punkt) dieser Funktion genutzt – die „Suche“ beginnt an einem zufälligen Startpunkt und setzt sich in die Richtung des steilsten Abstiegs fort, bis das Minimum, an dem die erste Ableitung und damit die Steigung gleich Null ist, gefunden wird.[45]

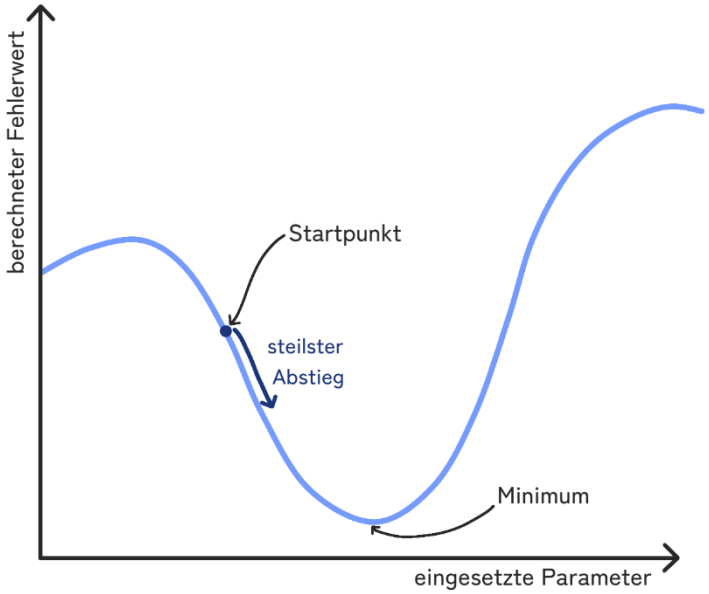


Abb. 15 - Gradientenabstieg_1

Hierbei erweist sich sofort das Problem, dass ein lokales, statt das globale Minimum gefunden wird, was wiederum bedeutet, dass das vom Computer gefundene Minimum dem naheliegendste Minimum und nicht unbedingt dem mit dem kleinstmöglich zu findenden Wert entspricht und somit nicht die optimalen Parameter widerspiegelt *[sh. Abb. 16]*.[46]

[44] Vgl. Kersting et al. (2019), S. 174, 179
[45] Vgl. Kersting et al. (2019), S. 177; Vgl. Mueller et al. (2020), S. 130f
[46] Vgl. Mueller et al. (2020), S. 130f; vgl. Kersting et al. (2019), S. 178; vgl. Alpaydın. (2008), S. 220

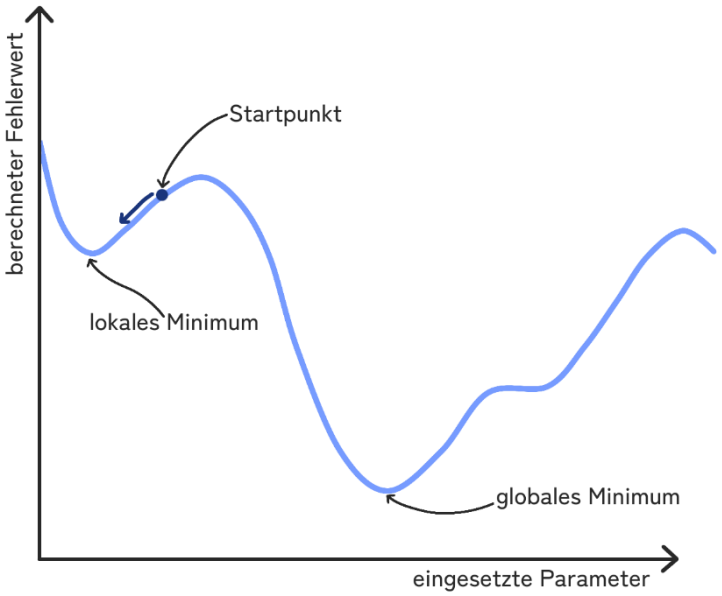


Abb. 16 - Gradientenabstieg_2

Um diese Schwäche zu beheben, können mehrere Versuche von jeweils unterschiedlichen Startpunkten aus durchgeführt werden, sodass nach Vergleich der gefundenen Minima der schlussendlich niedrigste Wert eruiert werden kann *[sh. Abb. 17]*.[47]

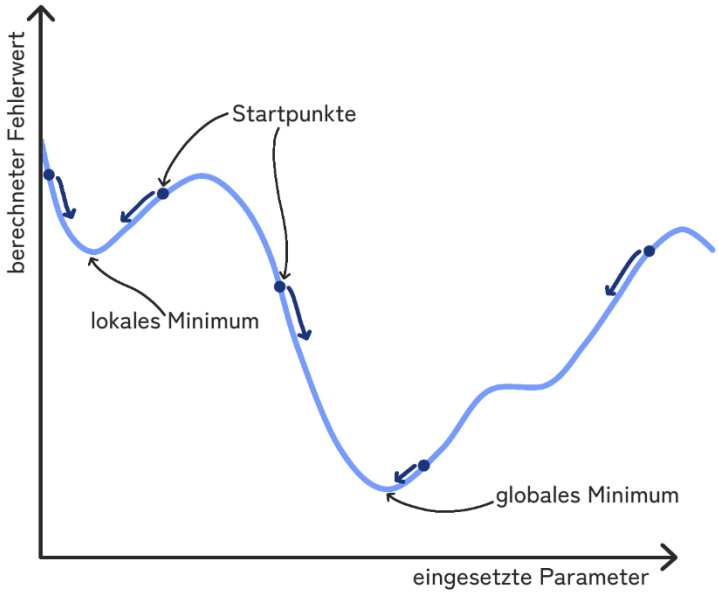


Abb. 17 - Gradientenabstieg_3

[47] Vgl. Mueller et al. (2020), S. 130f; vgl. Kersting et al. (2019), S. 178; vgl. Alpaydın. (2008), S. 220

Entscheidend für die Genauigkeit, Zuverlässigkeit und Stabilität des Netzes ist eine sinnvoll gewählte Lernrate (sh. Info „Lernrate).[48]

> *INFO - Lernrate:*
>
> Unter der Lernrate (auch Lernfaktor oder Schrittweite genannt) versteht man das Tempo, mit welchem die Parameter im neuronalen Netz geändert werden. Ist dieser Wert zu klein, dauert die Optimierung sehr lange, ist er zu groß, kann keine gute Lösung gefunden werden. Verwendet werden kann ein Wert zwischen ~0,00001 und 1,0, wobei er bei den meisten Anwendungen unter 0,2 angesetzt ist.
> Um die Schnelligkeit und Genauigkeit zu vereinen, kann auch eine adaptive Lernrate herangezogen werden, die sich bei fortschreitendem Lernprozess anpasst. Somit kann beispielsweise anfangs mit einer großen Lernrate Zeit eingespart und sich mit der Zeit, durch eine kleiner werdende Lernrate, dem genauen Minimum präzise angenähert werden. Andere Methoden sehen z.B. vor, dass sich die Lernrate abhängig von den Verbesserungen des Algorithmus variabel anpasst. Welche Lernrate die sinnvollste ist muss durch Ausprobieren evaluiert werden.[49]

3.2 Schwächen

Beim Training neuronaler Netze muss auf gewisse Schwächen Acht gegeben werden, damit ein Deep Learning-Modell auch richtig funktionieren und in neuen Umgebungen angewendet werden kann ohne fehlerhafte Ergebnisse zu liefern. In diesem Kapitel werden einige dieser Schwächen erläutert.

3.2.1 Unter- und Überanpassung

Eine Unteranpassung liegt an der Bereitstellung von zu wenig Daten und zu kurzem Training. Dadurch kann das Modell keine passende Annäherung finden, da gewisse nötige Details nicht erkannt werden können *[sh. Abb. 18 – Beispiel eines Regressionsproblems]*.[50]

[48] Vgl. Mueller et al. (2020), S. 186
[49] Vgl. Mueller et al. (2020), S. 169, 186; vgl. Alpaydın. (2008), S. 220, 270
[50] Vgl. Mueller et al. (2020), S. 51; vgl. Alpaydın. (2008), S. 36

Eine Überanpassung hingegen entsteht durch zu viele detailreiche Daten und zu langem Training, wodurch der Computer jedes kleinste Detail aller Daten schlicht auswendig lernt und so die Trainingsdaten immer korrekt auswerten kann. Jedoch versteht er dabei keine Zusammenhänge und Muster und ist nicht in der Lage Regeln abzuleiten, weswegen er mit neuen, noch unbekannten Daten nicht umgehen kann – so wird das neuronale Netz und dementsprechend die Ergebnisse unbrauchbar *[sh. Abb. 18]*.[51]

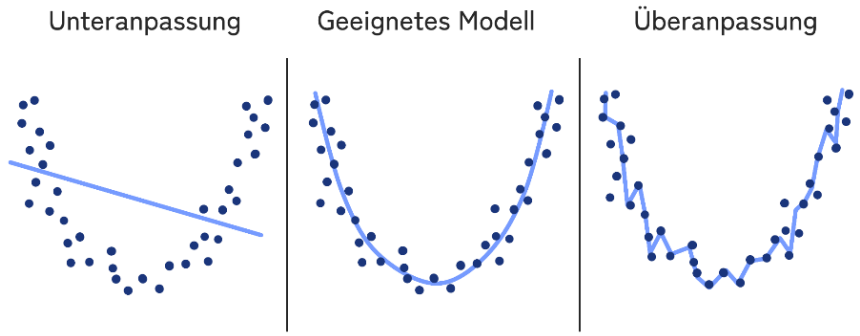


Abb. 18 - Unter-/ Überanpassung

Um ein Modell perfekt zu generalisieren (Generalisierung = Balanceakt zwischen möglichst guter Optimierung und dem Vermeiden von Unter-/ Überanpassung) muss das Training an dem Punkt gestoppt werden, an welchem keine Verbesserung der Zuverlässigkeit mehr auftritt oder sich das Modell in seinen Vorhersagen verschlechtert. Um diesen Punkt zu erkennen, werden Validierungsdaten (= zuvor abgespaltener Teil der Trainingsdaten) herangezogen. Nach Einsatz der Beispieldaten wird die Verlustfunktion mit diesen neuen Daten überprüft und je nachdem, ob eine Verbesserung oder Verschlechterung vorliegt, wird das Training fortgesetzt oder abgebrochen.[52]

Um zu entscheiden, wie lange und mit wie vielen Daten ein Modell trainiert werden soll, um eine Unter- oder Überanpassung zu vermeiden, ist meist eine menschliche Einschätzung von Nöten.[53] Auch hier gilt, dass verschiedene Modelle ausprobiert werden müssen, damit am Ende das Modell mit dem geringsten Fehler der Validierungsdaten gefunden werden kann.[54]

[51] Vgl. Mueller et al. (2020), S. 51, 170
[52] Vgl. Mueller et al. (2020), S. 130, 171
[53] Vgl. Mueller et al. (2020), S. 51
[54] Vgl. Alpaydın. (2008), S. 37

3.2.2 Verzerrung-Varianz-Dilemma

Verzerrung und Varianz sind beides Fehlerquellen, die versucht verhindert bzw. vermindert zu werden. Bei einer Verzerrung sind die Prognosen des Modells im Durchschnitt nicht korrekt, da das Modell aufgrund einer Unteranpassung zu simpel ist. Unter der Varianz ist grundsätzlich die Streuung zu verstehen. Diesem Problem liegt die Überanpassung zugrunde – das Modell passt sich zu stark den Trainingsdaten an und ist bei neuen Datensätzen unbrauchbar *[sh. Abb. 19]*.[55]

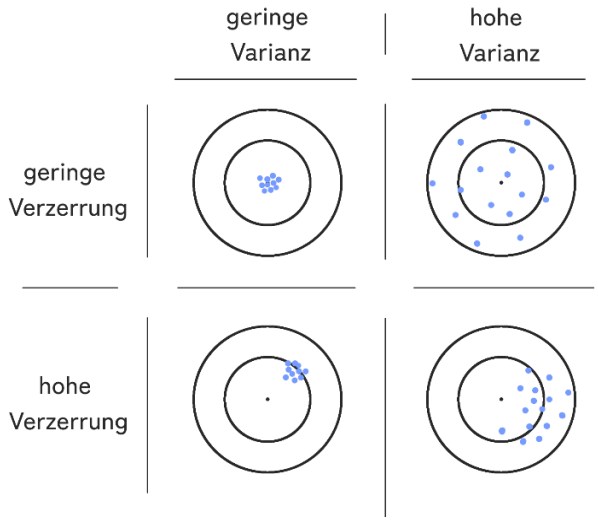


Abb. 19 - Verzerrung-Varianz-Dilemma

Da es sich hier um ein Dilemma handelt, gibt es keine optimale Lösung, sondern es müssen an gewissen Punkten Abstriche gemacht werden – bei hoher Verzerrung kann eine niedrige Varianz erreicht werden und andersrum. Das bestmögliche Ergebnis erhält man, indem der Gesamtfehler so niedrig wie möglich gehalten wird. Es muss somit ein Gleichgewicht zwischen Verzerrung und Varianz hergestellt werden, wobei die Komplexität nicht zu gering, aber auch nicht so hoch sein darf *[sh. Abb. 20]*.[56]

[55] Vgl. Kersting et al. (2019), S. 121-123
[56] Vgl. Kersting et al. (2019), S. 123

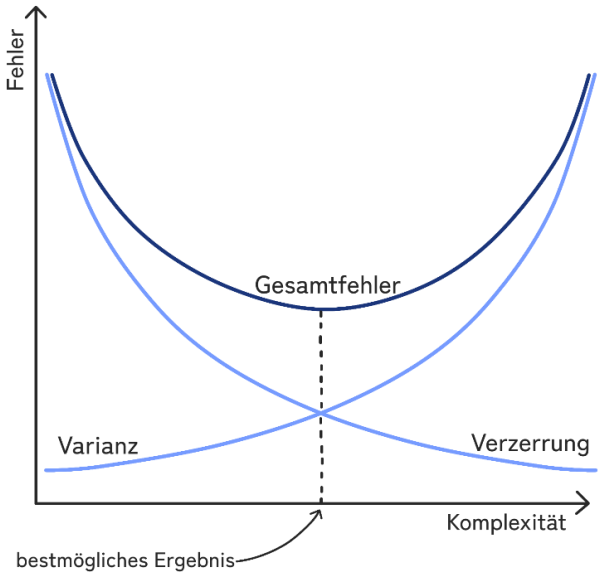


Abb. 20 - Verzerrung-Varianz-Funktionen

3.2.3 Rauschen

Ungewollte und unerwünschte Anomalien in Daten werden als Rauschen bezeichnet. Diese beschädigten Daten werden auch rauschende oder verrauschte Daten genannt und können zu fehlerhaften Schlussfolgerungen führen, da sie den Datensatz verfälschen und das Lernen von Zusammenhängen sowie das Erkennen von Mustern und Regeln erschweren.

Rauschen kann verschiedene Ursachen haben. Mögliche Beispiele sind Ungenauigkeiten beim Aufzeichnen aller Eingabedaten oder Fehler beim Einteilen der Daten.[57]

4 Verschiedene Konzepte zum Trainieren neuronaler Netze

In den folgenden Kapiteln werden die drei hauptsächlich genutzten Herangehensweisen zum Training neuronaler Netze und die Anwendungen dieser bzw. der jeweiligen Lernalgorithmen beschrieben.

[57] Vgl. Alpaydın. (2008), S. 28f

4.1 Überwachtes Lernen

Das entscheidende Merkmal beim überwachten Lernen (engl. supervised learning, SL) ist, dass nicht nur die Eingabedaten der Trainings- und Testbeispiele, sondern auch die korrekte Lösung bereits bekannt sind. Dafür müssen tausende Datenbeispiele zuvor per Hand zugeordnet und beschriftet werden, was bereits den größten Nachteil dieser Methode veranschaulicht.[58] Das neuronale Netz erhält somit alle Details der Daten, alle Merkmale und das gewollte Ergebnis und muss mithilfe dieser ganzen Informationen allgemeingültige Regeln suchen, mit welchen alle Daten korrekt bewertet werden können.[59]

Eingeteilt werden können überwachte Lernalgorithmen in Regressions- und Klassifikationsprobleme. Folgend wird ein Einblick in überwacht lernende Algorithmen gegeben:[60]

4.1.1 Regression

Mithilfe der Regression können spezifische Prognosen mit numerischen Werten erstellt werden. Ein mögliches Anwendungsbeispiel ist die Bestimmung des Gewichts eines bestimmten Tieres in Abhängigkeit seiner Größe. [61]

Lineare Regression:

Die lineare Regression stellt ein Verfahren dar, welches versucht eine passende Gerade für eine Vielzahl von Datenpunkten zu bestimmen, um so auch bei noch unbekannten Daten eine richtige Prognose liefern zu können. In *Abb. 21* sieht man ein Beispiel für eine einfache lineare Regression, dargestellt durch ein zweidimensionales Koordinatensystem. Bei mehreren Eingabemerkmalen entsteht statt einer Geraden, eine Ebene in einem mehrdimensionalen Raum – diese Art wird als multiple lineare Regression bezeichnet (ab einer vierten Dimension kann das Modell grafisch jedoch nicht mehr dargestellt werden). Die Werte auf der x-Achse sind die Eingabewerte und jene auf der y-Achse die Ausgabewerte. Anfangs werden also Datenpaare (x/y-Koordinaten) eingefügt, der Algorithmus findet anschließend mithilfe von den in Kapitel „grundsätzliche Vorgehensweise

[58] Vgl. Kersting et al. (2019), S. 26f, 47
[59] Vgl. Mueller et al. (2020), S. 50
[60] Vgl. Mueller et al. (2020), S. 48
[61] Vgl. Kersting et al. (2019), S. 45; Vgl. Alpaydın. (2008), S. 9

beim Training von neuronalen Netzen“ beschriebenen Methoden eine passende Gerade mit kleinstmöglicher Kostenfunktion und wird dann mit den – ebenfalls beschrifteten – Testdaten getestet. Wenn das Modell steht, kann jedem beliebigen x-Wert (Eingabewert), ein eindeutiger y-Wert (Ausgabewert) zugeordnet werden.[62]

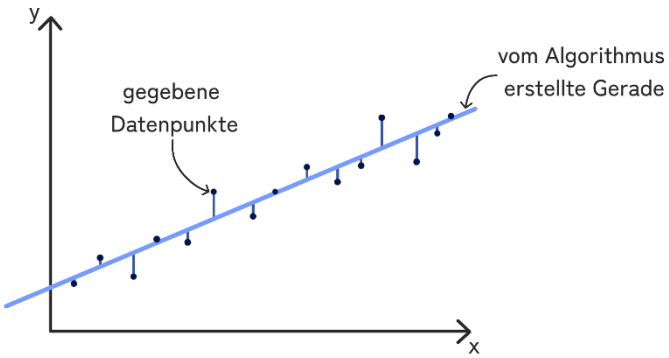


Abb. 21 - Lineare Regression

Die lineare Regression ist eine einfache Vorhersagemethode des maschinellen Lernens und oftmals aufgrund ihrer Simplizität für einige komplexere Anwendungen mit komplizierten Zusammenhängen nicht nutzbar. Ein etwas weiter ausgebautes und ausgeklügeltes, aber dennoch von ihrer Funktionsweise vergleichbares Verfahren, kann beim Deep Learning genutzt werden. Dabei wird eine nichtlineare Ebene berechnet, die bestmöglich durch die Mitte der Datenmenge verläuft.[63]

Nichtsdestotrotz ist die lineare Regression ein leistungsstarker Algorithmus, der aufgrund seines schnellen Trainings und einer unkomplizierten Implementierung bei simplen Aufgaben besonders praktikabel ist und in diesem Bereich seine Anwendung findet.[64] Grundlegende Voraussetzung für die Nutzbarkeit der linearen Regression ist immer ein linearer Zusammenhang zwischen den Daten. Bei nichtlinearen Zusammenhängen, wie beispielsweise dem menschlichen Wachstum, ist eine Anwendung der linearen Regression nicht aussagekräftig.[65]

[62] Vgl. Mueller et al. (2020), S. 133-135
[63] Vgl. Mueller et al. (2020), S. 133; Vgl. Kersting et al. (2019), S. 66
[64] Vgl. Mueller et al. (2020), S. 134
[65] Vgl. Mueller et al. (2020), S. 142

Polynomiale Regression:

Neben der linearen Regression mit Geraden oder Ebenen gibt es noch weitere Arten, die Polynome verschiedener Grade, je nach Datensatz, erstellen. Von der grundsätzlichen Funktionsweise unterscheiden sich diese Arten nicht. In *Abb. 22* sind ein paar mögliche Beispiele polynomialer Regression abgebildet.[66]

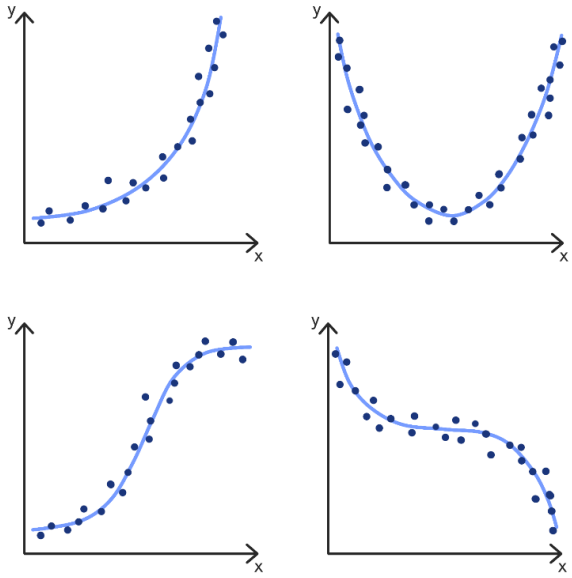


Abb. 22 - Polynomiale Regression

Logistische Regression:

Die logistische Regression gibt anders als die lineare Regression keine präzisen Ausgabewerte, sondern Wahrscheinlichkeiten aus. Somit kann bestimmt werden, mit welchen Wahrscheinlichkeiten bestimmte Eigenschaften beispielsweise gewissen Klassen (sh. Kapitel „Klassifikation“) zugeordnet werden können oder gewisse Ereignisse auftreten. Da Wahrscheinlichkeiten immer im Bereich zwischen 0,0 und 1,0 liegen, wird, wie in *Abb. 23* abgebildet, eine S-Kurve, derer Funktionswerte in genau diesem Bereich liegen, verwendet.[67]

[66] Vgl. Kersting et al. (2019), S. 42
[67] Vgl. Kersting et al. (2019), S. 108f

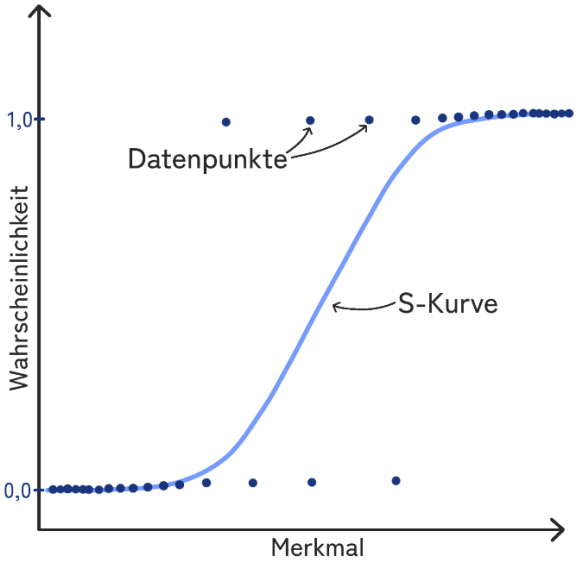


Abb. 23 - Logistische Regression

4.1.2 Klassifikation

Bei der Klassifikation werden die Daten in einzelne Gruppen bzw. sogenannte Klassen eingeordnet. Dafür muss der Algorithmus bestimmte Regeln aus vielen verschiedenen, verknüpften Eigenschaften bzw. Merkmalen finden, die es dem Computer ermöglichen jegliche Daten zu den passenden Klassen zuordnen zu können.[68] Ein typisches Anwendungsbeispiel der Klassifikation ist die Einteilung von Bildern in Katzen und Hunde.

Auch die Klassifikation kann in einem Koordinatensystem dargestellt werden. In *Abb. 24* ist eine Klassifikation zweier Objekte mit zwei Merkmalen in einem zweidimensionalen Koordinatensystem abgebildet. Der Klassifikationsalgorithmus hat dann die Aufgabe, eine Trennlinie zwischen den entstandenen „Datenwolken“ zu bilden. Um hierbei die optimale Trennlinie mit kleinstmöglicher Kostenfunktion zu eruieren, wird wie in „grundsätzliche Vorgehensweise beim Training von neuronalen Netzen“ beschrieben vorgegangen.

[68] Vgl. Kersting et al. (2019), S. 45-47

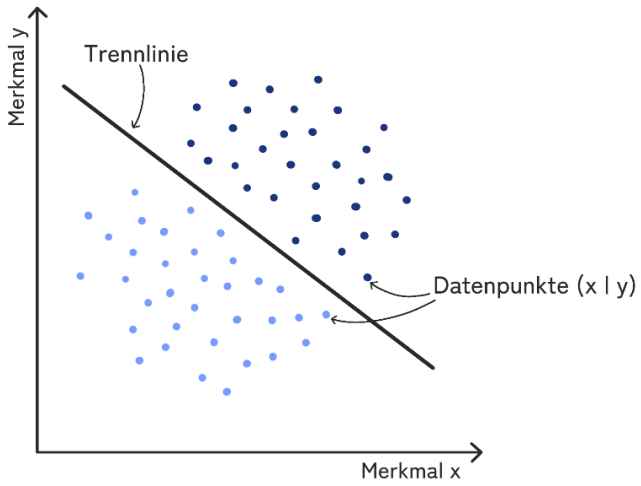


Abb. 24 - Klassifikation

In den meisten Fällen lassen sich die verschiedenen Datenwolken jedoch nicht durch eine Gerade trennen, sondern besitzen komplexere Verhältnisse zueinander, wodurch die Trennlinie jeglicher Polynomfunktion entsprechen kann *[sh. Abb. 25 (1)]*. Auch mehr als zwei Klassen lassen sich eingrenzen *[sh. Abb. 25 (2)]*. In praktischen Anwendungen wird jedoch schnell sichtbar, dass eine perfekte Abgrenzung bei gewissen Klassifikationsproblemen nicht immer erreicht werden kann bzw. das Finden einer Trennlinie oft nicht optimal möglich ist.[69]

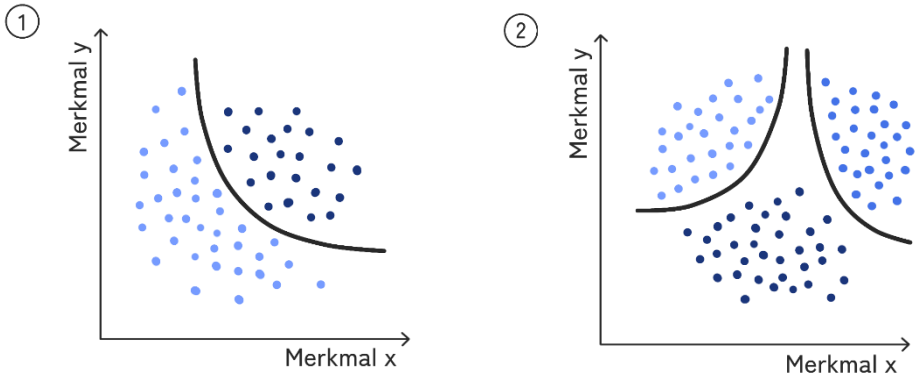


Abb. 25 - Klassifikationsbeispiele

Support Vector Machine:

Um Datenwolken geeignet zu trennen, können sogenannte Stützvektoren angewendet werden. Dafür wird eine „Support Vector Machine“ eingesetzt, was im Prinzip ein Algorithmus ist, der mittels Stützvektoren eine geeignete und sichere Trennlinie mit dem größtmöglichen Abstand zwischen den jeweiligen Datenwol-

[69] Vgl. Kersting et al. (2019), S. 48-51

ken findet. Als Stützvektoren werden die äußersten Datenpunkte bezeichnet, welche die Position der Trennlinie bestimmen – all jene Daten, die hinter diesen Stützvektoren liegen, haben somit keinen Einfluss auf das resultierende Ergebnis.[70]

Entscheidungsbaum:

Entscheidungsbäume sind eine andere, leicht nachvollziehbare Art der Darstellung zum Lösen von Klassifikationsproblemen. Der hierarchische Aufbau besteht aus einer sogenannten Wurzel und darauffolgenden Knotenpunkten, an welchen Entscheidungen getroffen werden, um am Ende des Baumes die korrekte Klassenzuordnung zu erzielen *[sh. Abb. 26]*. Ausschlaggebend für ein aussagekräftiges Ergebnis ist die Anordnung bzw. Reihenfolge und Anzahl der Abzweigungen und Entscheidungen.[71]

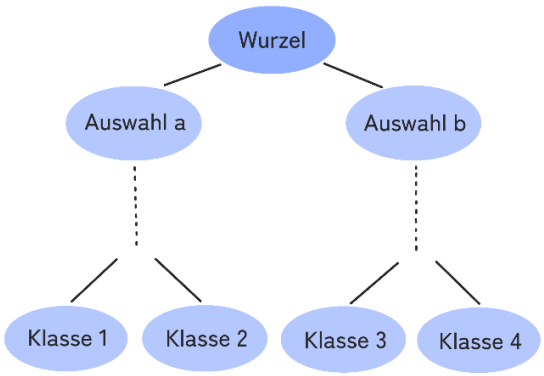


Abb. 26 - Entscheidungsbaum

4.2 Unüberwachtes Lernen

Beim unüberwachten Lernen (engl. unsupervised learning, UL) fällt das Beschriften der Eingabedaten weg. Der Algorithmus kennt also die jeweils korrekten Ausgaben nicht, sondern muss selbständig Muster, Zusammenhänge und Strukturen in den Datensätzen auffinden und so Regeln ausarbeiten. Unüberwachte Lernalgorithmen haben häufig die Aufgabe ähnliche Daten zu gruppieren (= Cluster-Bildung – sh. Kapitel „Clustering“) oder Anomalien in großen Datenmengen zu erkennen.[72]

[70] Vgl. Kersting et al. (2019), S. 96, 98-100
[71] Vgl. Kersting et al. (2019), S. 113-115
[72] Vgl. Mueller et al. (2020), S. 49

Eine Schwäche des unüberwachten Lernens ist jedoch, dass dem Algorithmus die richtigen Ergebnisse nicht bekannt sind. Dadurch kann es vorkommen, dass der Algorithmus die Daten an anderen Merkmalen als an den beabsichtigten unterscheidet und folglich eine vollkommen andere Aufgabenstellung als die gewollte löst.[73]

4.2.1 Clustering

Beim Clustering bzw. bei der Clusteranalyse werden unbeschriftete Daten in Gruppen (= Cluster) eingeteilt. Diese Gruppen sind nicht vorgegeben und somit im Vorhinein unbekannt. Sie werden so gebildet, dass ähnliche Daten einer Gruppe zugehören und sich möglichst klar von Daten anderer Gruppen unterscheiden *[sh. Abb. 27 – Beispiel von Datenpunkten mit zwei Merkmalen, die sinnvoll in zwei Gruppen eingeteilt werden können]*.[74]

k-Means-Clustering:

Das k-Means-Clustering ist ein Clustering-Verfahren, bei dem eine bestimmte, vorher bekannte Anzahl (k) von Gruppen gebildet werden. In dem Beispiel in *Abb. 27* ist k somit 2, da der Datensatz in 2 Cluster eingeteilt wird. Bei k = 3 würde eine erfolgreiche Clusteranalyse wie in *Abb. 28* abgebildet aussehen.[75]

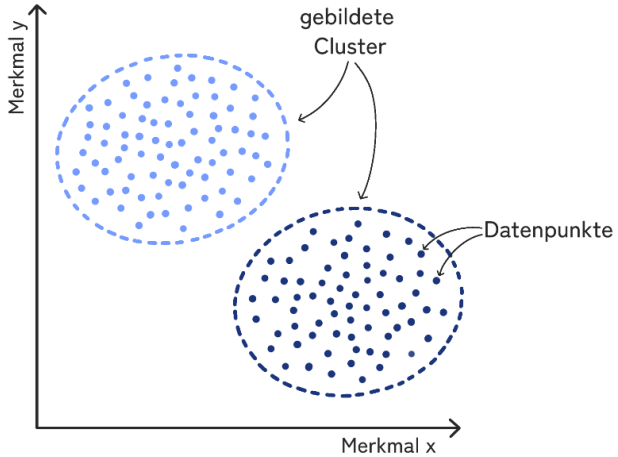


Abb. 27 - Clustering

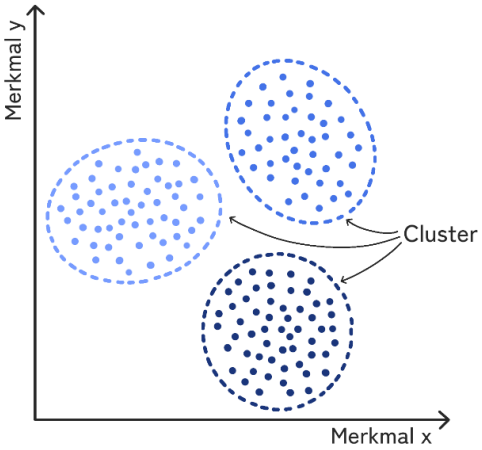


Abb. 28 - k-Means-Clustering

[73] Vgl. Kersting et al. (2019), S. 25
[74] Vgl. Kersting et al. (2019), S. 56
[75] Vgl. Kersting et al. (2019), S. 82f

Nachdem die Anzahl der Cluster definiert wurde, werden die jeweilgen Mittelpunkte gesucht. Um diese ausfindig zu machen, startet der Algorithmus vorerst mit k willkürlich gewählten Punkten als eventuelle Mittelpunkte *[sh. Abb. 29 (1) – beispielhafte Darstellung einer Clusteranalyse mit k = 2 in einem 2-dimensionalen Koordinatensystem]*. Anschließend wird der Abstand zu den einzelnen Mittelpunkten gemessen und jeder Datenpunkt dem Cluster mit dem nächsten Mittelpunkt zugeordnet *[sh. Abb. 29 (2)]*. Da nun eine vollkommen zufällige und fehlerhafte Einteilung vorliegt, werden die Mittelpunkte neu berechnet, wodurch einige Datenpunkte nun einem anderen Cluster als zuvor zugeordnet werden *[sh. Abb. 29 (3)]*. Bis alle Punkte sinnvoll in Cluster eingeteilt worden sind, müssen die Mittelpunkte immer wieder aktualisert und die Datenzuordnung immer wieder angepasst werden *[sh. Abb. 29 (4)]*. Am Ende dieses Prozesses ist eine Überprüfung der Clusteranalyse durch einen Menschen dennoch nötig, um abzuwägen, ob der Computer sinnvolle Cluster erstellt und sinnvolle Merkmale gewählt hat.[76]

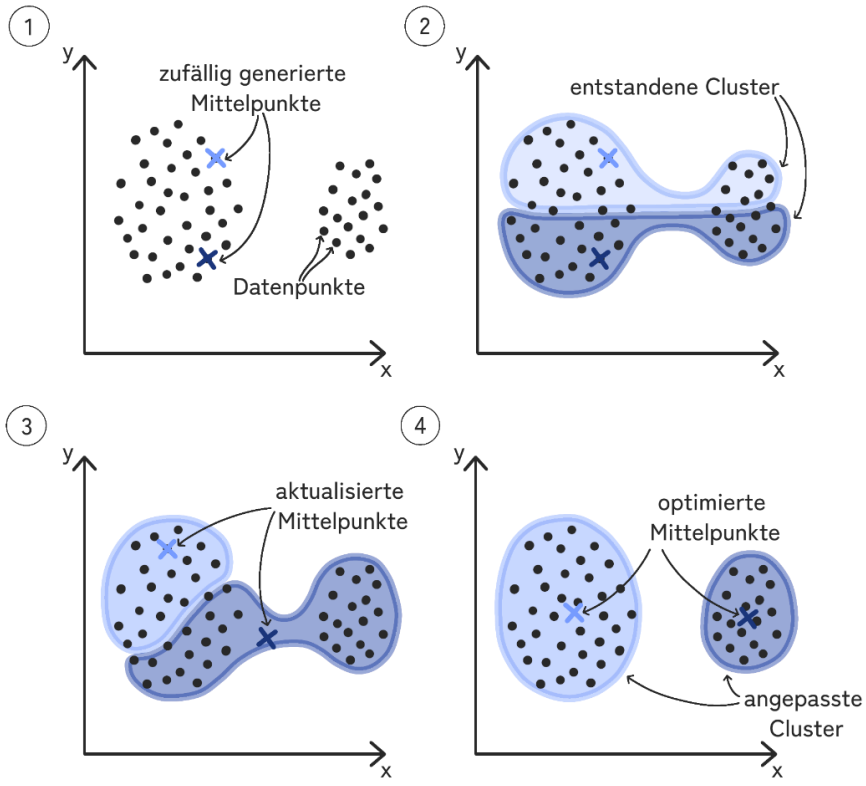


Abb. 29 - Clusteringverfahren

[76] Vgl. Kersting et al. (2019), S. 82-88

Bei Unternehmen finden Clusteranalysen beispielsweise im Bereich der Kunden- und Produktanalysen ihren Nutzen, um z.B. je nach Personengruppen spezifische Entscheidungen treffen und Werbungen anzeigen zu können. Auch bei der Bildkompression können Clusteranalysen durchgeführt werden, um Pixel mit ähnlichen Farben zu gruppieren und die nachfolgende Bilderkennung damit zu simplifizieren.[77]

4.3 Bestärkendes Lernen

Das bestärkende Lernen (engl. reinforcement learning, RL) arbeitet mit Feedback, also einer Belohnung, um positives, erwünschtes Verhalten zu verstärken und einer Bestrafung bei negativem, unerwünschtem Verhalten. Das Ziel des Algorithmus ist das positive Feedback zu maximieren und das negative hingegen zu minimieren.[78]

Vorteile dieser Art des Lernens sind, dass keine riesigen Datensätze sowie keine Beschriftung der Daten notwendig sind, aber trotzdem leistungsstarke Systeme erstellt werden können.[79]

Grundprinzip [sh. Abb. 30]:

Dem Akteur bzw. Agenten (z.B. ein physischer oder digitaler Roboter) wird der sogenannte Zustand von der Umwelt bzw. Umgebung übermittelt und dieser reagiert anschließend in Form einer Aktion auf diese Information. So interagiert er mit seiner Umwelt und ruft eine Änderung des Zustands hervor. Für diese Aktion bekommt der Agent im Anschluss Feedback/ eine Rückmeldung, wodurch dieser nun eigenständig eruieren kann, wie er die höchstmögliche Belohnung erhält (sh. Kapitel „Q-Learning“). Wichtig für das Erreichen des gewünschten Zielzustands sind gute Taktiken (sh. Info „Taktik“).[80]

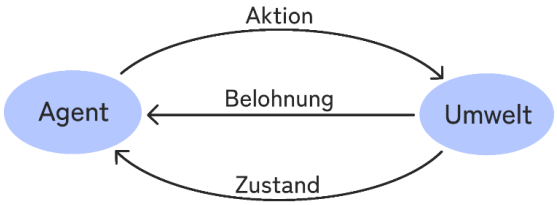


Abb. 30 - RL

[77] Vgl. Alpaydın. (2008), S. 11
[78] Vgl. Kersting et al. (2019), S. 203f; vgl. Mueller et al. (2020), S. 49
[79] Vgl. Mueller et al. (2020), S. 324
[80] Vgl. Mueller et al. (2020), S. 324f; vgl. Alpaydın. (2008), S. 13, 398

INFO - Taktik:

Entscheidend beim bestärkenden Lernen sind meist nicht einzelne Aktionen, sondern eine Reihe von korrekten, zielführenden Aktionen, welche die Taktik bilden. Der Algorithmus muss somit die Qualität der Taktiken einschätzen und so aus den zuvor getätigten Aktionen lernen können. Am Beispiel eines Brettspiels wie Schach ist dieses Prinzip leicht zu verstehen: Der Algorithmus hat genau dann eine gute Leistung erbracht, wenn das Spiel am Ende gewonnen ist – wie gut einzelne Züge dabei sind, ist bei einem erfolgreichem Spiel nicht entscheidend.[81]

Eingesetzt werden bestärkende Lernverfahren besonders in ungewissen Umgebungen wie beispielsweise bei selbstfahrenden Autos oder Robotern verschiedenster Aufgabenbereiche. Insbesondere Spiele können durch bestärkendes Lernen von Computern erfolgreich erlernt werden.[82]

4.3.1 Q-Learning

Der beim bestärkenden Lernen vorrangig genutzte Lernalgorithmus ist das Q-Learning und funktioniert nach dem in *Abb. 30* abgebildeten Schema. Das „Q“ steht hierbei für „Quality“ und gibt damit die Qualität für einen bestimmten Status an – also wie „gut“ oder „schlecht“ die jeweiligen Aktionen zu verschiedenen Zuständen sind, wodurch die bestmöglichen Aktionen ermittelt werden können. Diese Q-Werte werden in einer Q-Tabelle angeführt *[sh. Abb. 31]* und nach jeder Aktion aktualisiert.[83]

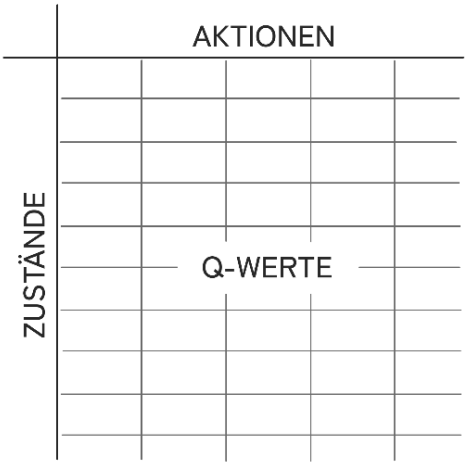


Abb. 31 - Q-Learning Tabelle (Schema)

[81] Vgl. Alpaydın. (2008), S. 12f
[82] Vgl. Mueller et al. (2020), S. 323f
[83] Vgl. *What is Q-Learning: The Best Guide To Understand Q-Learning.* (2022, 29. August)

Ein veranschaulichendes Beispiel für diese Art des Lernens ist ein Roboter, der sich in einer neuen Welt zurechtfinden muss *[sh. Abb. 32]*. Dabei gibt es an gewissen Orten des Feldes Belohnungen (z.B. Energie) und an anderen Bestrafungen (z.B. Kurzschluss). Die ausführbaren Aktionen sind in diesem Beispiel „hoch“, „rechts“, „runter“ und „links“. Jeder Aktion auf einem Feld kann somit ein Q-Wert abhängig von dem Feedback zugeordnet werden – je höher der Wert, desto größer die Belohnung. Die Aufgabe des Roboters ist es herauszufinden, wie er die meisten Belohnungen finden kann, ohne dabei bestraft zu werden.[84]

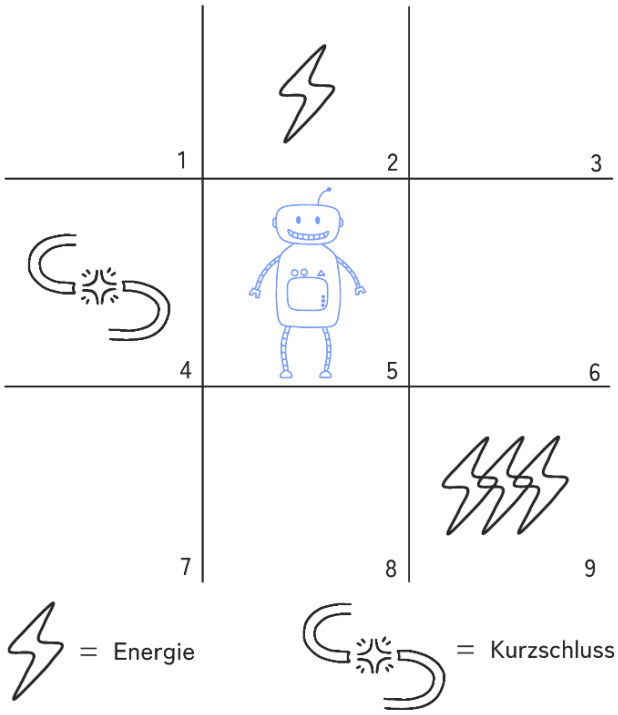


Abb. 32 - Q-Learning Beispielfeld

Zu Beginn sind die Q-Werte in der Q-Tabelle alle auf „0“ gesetzt, da der Roboter noch nichts von seiner Umwelt erkundet hat und somit keine Informationen über diese vorliegen *[sh. Abb. 33]*. Durch Erkunden der Umwelt, können diese Q-Werte mit der Zeit aktualisiert werden. Je mehr Werte in der Tabelle initialisiert sind, desto besser kann der Roboter seine Entscheidung über die nächstbeste Aktion fällen. Bei dieser Entscheidung muss der Roboter jedoch auch zwischen dem Erkunden und dem Ausschöpfen bereits bekannter Werte abwägen, um den effizientesten Weg zu ermitteln (sh. Info „Exploration-Exploitation-Dilemma“).[85]

[84] Vgl. deeplizard. (2018, 6. Oktober), 2:06-3:20
[85] Vgl. deeplizard. (2018, 6. Oktober), 3:45-5:16

	hoch	rechts	runter	links
Feld 1 (leer)	0	0	0	0
Feld 2 (Energie)	0	0	0	0
Feld 3 (leer)	0	0	0	0
Feld 4 (Kurzschluss)	0	0	0	0
Feld 5 (leer)	0	0	0	0
Feld 6 (leer)	0	0	0	0
Feld 7 (leer)	0	0	0	0
Feld 8 (leer)	0	0	0	0
Feld 9 (3 Energie)	0	0	0	0

Abb. 33 - Q-Learning Tabelle (Beispiel)

INFO - Exploration-Exploitation Dilemma:

Um dem Ziel der größtmöglichen Belohnung nachzugehen, wägt der Algorithmus zwischen dem Erkunden (exploration) und dem Ausschöpfen bzw. Ausnutzen (exploitation) ab.[86] Bei der Erkundung wird eine zufällige Handlung durchgeführt, während bei der Ausnutzung die aktuell beste Handlung durchgeführt wird.[87]

Sowohl die Erkundung als auch die Ausschöpfung sind wichtig für ein gutes Ergebnis. Legt der Algorithmus zu viel Wert auf die Erkundung, arbeitet dieser ineffizient, da er die bereits bekannten Informationen nicht nutzt. Liegt jedoch das Gegenteil vor und der Algorithmus legt zu viel Wert auf die Ausschöpfung, wird der erstbeste Weg ausgenutzt und ständig wiederholt, obwohl möglicherweise ein besserer Weg gefunden werden kann, der die Belohnung vervielfachen würde, was ebenfalls ineffizient ist.[88]

5 Programmierung eines neuronalen Netzes

Im Rahmen der vorwissenschaftlichen Arbeit wurde ein neuronales Netz zur Erkennung handschriftlicher Ziffern von mir programmiert. Dieses ist dazu in der Lage jegliche neuen handgeschriebenen Ziffern mit einer Wahrscheinlichkeit P von P > 95% korrekt zu prognostizieren.

[86] Vgl. Kersting et al. (2019), S. 207
[87] Vgl. Mueller et al. (2020), S. 332
[88] Vgl. deeplizard. (2018, 6. Oktober), 5:16-6:30

Dafür wird mit Programmcode ein neuronales Netz mit Eingabeschicht, zwei verborgenen Schichten und einer Ausgabeschicht erstellt und anschließend mit einem Datensatz trainiert. Die verborgenen Schichten bestehen jeweils aus 128 Neuronen und arbeiten mit der ReLU-Aktivierungsfunktion. Bei der Ausgabeschicht mit 10 Neuronen (Ziffern 0 bis 9) wurde die Softmax-Aktivierungsfunktion angewendet (sh. Kapitel „Aktivierungsfunktionen“). Auch eine Optimierungs- und Verlustfunktion wurden festgelegt, mit welchen das Modell trainiert werden soll. Anschließend wurde das Netz mit den Trainingsdaten des MNIST-Datensatzes (sh. Info „MNIST-Daten“) angepasst. Wiederholt wurde dieser Vorgang vier Mal, bevor das Modell mit den Testdaten getestet wurde, um so den Fehler bzw. die Genauigkeit des Netzes zu bestimmen *[sh. Abb. 34].*[89]

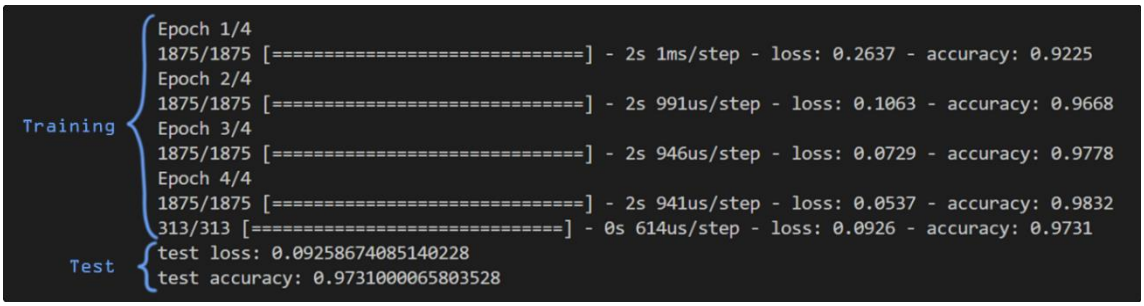


Abb. 34 - Programm: Training & Test

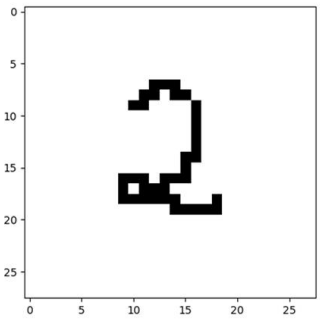
Nachdem das neuronale Netz trainiert wurde, können nun eigene Daten genutzt werden. Hierfür wurden zehn verschiedene Ziffern digital geschrieben und im Anschluss vom neuronalen Netz bestimmt. Dabei hat dieses neun von zehn Ziffern korrekt prognostizieren können, was einer Wahrscheinlichkeit P von P = 90% entspricht *[sh. Abb. 35 – Beispiel der handgeschriebenen Ziffern mit der jeweiligen Prognose der ersten fünf Ziffern]*.

> *INFO - MNIST-Daten:*
> Die MNIST-Daten sind tausende bereits beschriftete bzw. klassifizierte handgeschriebene Ziffern, die aufgeteilt sind in Trainings- und Testdaten und aus einer öffentlichen Bibliothek entnommen werden können.[90]

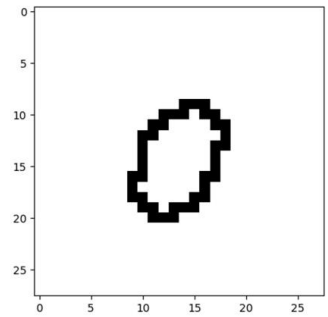
[89] Vgl. NeuralNine. (2021, 21. August); vgl. *Module: tf.* (2022, 26. Oktober); vgl. *Matplotlib.pyplot.* (o.D.)
[90] Vgl. NeuralNine. (2021, 21. August), 3:18-5:00

handgeschriebene Zahl:

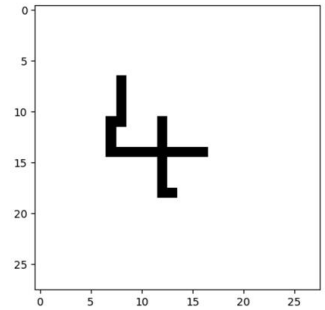
Ausgabe des neuronalen Netzes:



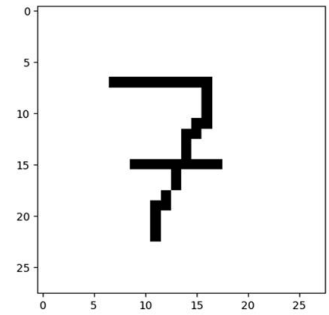
```
1/1 [==============================] - 0s 93ms/step
Die Ziffer ist wahrscheinlich eine 2
```



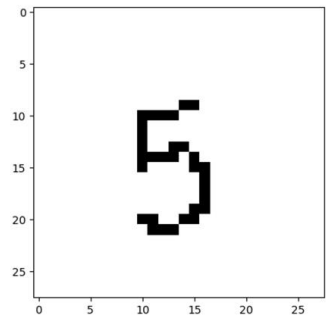
```
1/1 [==============================] - 0s 11ms/step
Die Ziffer ist wahrscheinlich eine 0
```



```
1/1 [==============================] - 0s 11ms/step
Die Ziffer ist wahrscheinlich eine 4
```



```
1/1 [==============================] - 0s 12ms/step
Die Ziffer ist wahrscheinlich eine 7
```



```
1/1 [==============================] - 0s 11ms/step
Die Ziffer ist wahrscheinlich eine 5
```

Abb. 35 - Programm: Zahl & Prognose

6 Anwendungen und ein Blick in die Zukunft

Wie in den vorherigen Kapiteln gesehen wird Maschinelles Lernen sowie Deep Learning heute bereits in vielen Bereichen angewendet. Einige weitere Beispiele, in welchen diese Technologien schon eingesetzt werden, sind folgende:

- Sprachbedienung bei Smartphones, Autos, Sprachassistenten, etc. für eine größere Benutzerfreundlichkeit[91]
- Verkehrsanalysen für bessere Vorhersagen von Navigationssystemen (z.B. für eine schnellere Bereitschaft der Feuerwehr)[92]
- Wetterprognosen sowie Vorhersagen über die Veränderung und den Einfluss des Klimawandels und Prognose von möglichen Naturkatastrophen[93]
- Personalisierte Werbungen und auf individuelle Interessen angepasste Anzeigen und Vorschläge im Internet (wird z.B. genutzt von „Amazon“ und ähnlichen Online-Shops, von Plattformen wie „Spotify“ oder „Netflix“ oder von Social-Media-Plattformen wie „YouTube“, „TikTok“ und Co.)[94]
- Roboter (z.B. Saug- oder Wischroboter), die sich in neuen Umgebungen zurechtfinden und Aufgaben durchführen müssen[95]
- Intelligente Rollatoren und weitere in der Medizin und Pflege gebrauchten Geräte, um die Arbeit der Pflegenden zu erleichtern[96]
- Feststellung und Verhinderung von Betrug bei beispielsweise Kreditkarten durch unnatürliche, auffällige Kaufmuster bzw. anderweitige Anomalien[97]
- Kundenservice mit Chatbots[98]
- Effizienzverbesserung und Ressourcenmanagement bei Maschinen durch eine optimal angepasste Arbeitsleistung und -geschwindigkeit[99]
- Von KI gemalte Kunstwerke, die beispielsweise einen gewissen Kunststil replizieren oder anhand von Textvorgaben ein Gemälde anfertigen kön-

[91] Vgl. Kersting et al. (2019), S. VII Vorwort
[92] Vgl. Kersting et al (2019), S. VII Vorwort
[93] Vgl. Mueller et al. (2020), S. 348
[94] Vgl. Kersting et al. (2019), S. VII Vorwort, 4f
[95] Vgl. Kersting et al. (2019), S. 225
[96] Vgl. Kersting et al. (2019), S. 230
[97] Vgl. Mueller et al. (2020), S. 34
[98] Vgl. Mueller et al. (2020), S. 35
[99] Vgl. Mueller et al. (2020), S. 34f

nen (wobei der Computer jedoch nicht kreativ ist, sondern lediglich die Pixel eines Bildes dargestellt als Zahlen analysiert und reproduziert)[100]

- Gesichtserkennung wie sie beispielsweise beim Smartphone zur Entsperrung oder bei der Polizei zum Identifizieren krimineller Personen genutzt wird[101]
- Unterstützung bei der medizinischen Diagnose von Krankheiten[102]

In Zukunft wird es immer mehr Einsatzbereiche für KI geben. Ein ausschlaggebender Punkt, der besonders beim Deep Learning essenziell ist, sind die Menge und Qualität der zur Verfügung stehenden Daten, mit welchen Machine-Learning-Modelle lernen können. Da durch die fortschreitende Digitalisierung die Menge an Daten immer weiter und rasanter wächst und Daten immer einfacher gesammelt werden können, können auch immer mehr und schneller neue Algorithmen für neue Problemstellungen trainiert und komplexere Aufgaben gelöst werden.[103]

Eine verbesserte, sichere und weiter ausgebaute Verwendung von selbstfahrenden Autos, eine zuverlässige Echtzeit-Sprachübersetzung bei Telefonaten oder ein Ausbau von komplexen, in der Medizin und Pflege einsetzbaren, smarten Robotern können in naher Zukunft durch Maschinelle Lernalgorithmen realisierbar sein und alltäglichen Nutzen finden.[104]

All die genannten Anwendungen gehören zur Kategorie der „schwachen KI“ und sind nicht vergleichbar mit einer „starken KI“ (sh. Info „Schwache vs. Starke KI“).[105]

> *INFO - Schwache vs. Starke KI:*
> Schwache Künstliche Intelligenzen sind auf spezifische, einzelne Problemstellungen trainiert. Die Maschine führt genau die Aufgabe aus, auf die sie programmiert wurde bzw. die der Lernalgorithmus erlernt hat. Sie haben keinen eigenen Willen, eigene Gedanken o.Ä. Starke Künstliche Intelligenzen hingegen haben

[100] Vgl. Mueller et al. (2020), S. 299f
[101] Vgl. Alpaydın. (2008), S. 2
[102] Vgl. Alpaydın. (2008), S. 7
[103] Vgl. Kersting et al. (2019), S. 143
[104] Vgl. Alpaydın. (2008), S. XIV Vorwort
[105] Vgl. Gondlach. (2021, 15. April)

ein Bewusstsein und sind in der Lage unbegrenzt viele Aufgaben zu lösen. Solche intelligenten Maschinen, wie sie in manchen Filmen dargestellt werden, gibt es jedoch heutzutage noch nicht und werden in absehbarer Zukunft auch nicht entwickelt.[106]

[106] Vgl. Gondlach. (2021, 15. April)

7 Resümee

Vielen ist gar nicht bewusst, was hinter einem Sprachassistenten wie „Alexa“, der Bild- und Gesichtserkennung des Smartphones, den Empfehlungssystemen von „Netflix“, „Amazon“, „Spotify“ und Co. oder selbstfahrenden Autos steckt. Die Lösung hinter diesem Mysterium ist die in dieser Arbeit vorgestellte und nähergebrachte Technologie des Deep Learnings – eine Technik, die auf dem Prinzip von künstlichen neuronalen Netzen basiert und es ermöglicht, komplexe Probleme durch maschinelles Lernen zu lösen.

Dabei wurde in dieser vorwissenschaftlichen Arbeit das grundlegende Vorgehen beim Erstellen eines Deep Learning-Modells vorgestellt, die verschiedenen Konzepte beim Training eines solchen untersucht und erläutert sowie eine praktische Anwendung anhand eines eigens programmierten neuronalen Netzes aufgezeigt. Insgesamt zeigen die Ergebnisse dieser Arbeit, dass Deep Learning eine vielversprechende Technologie ist, die in vielen Bereichen zur Lösung komplexer Probleme beitragen kann. Dabei ist wichtig, dass sich bewusst mit der Funktionsweise von Deep Learning und den Vorgehensweisen beim Training von neuronalen Netzen auseinandergesetzt werden muss, um die volle Leistungsfähigkeit von Deep Learning ausschöpfen zu können und nutzvolle Ergebnisse zu erzielen. Die Wahl der genutzten Techniken, Funktionen, Parametern, etc. ist dabei vor allem abhängig von dem Einsatzgebiet des Deep Learning-Modells und weiteren Gegebenheiten, wie beispielsweise dem zur Verfügung stehenden Datensatz.

Abschließend lässt sich sagen, dass Deep Learning eine vielversprechende Technologie ist, die bereits in vielen Bereichen unseres täglichen Lebens zum Einsatz kommt und ständig weiterentwickelt wird. Es gibt zahlreiche Möglichkeiten, wie Deep Learning eingesetzt werden kann und es besteht großes Potenzial, dass Deep Learning in Zukunft noch weiter an Bedeutung gewinnen wird. Eine Antwort darauf, was genau in Zukunft damit möglich sein wird, welche Anwendungen es ermöglichen wird, wie sich das Leben verändern wird und auch in welchen Bereichen mehr Vorsicht geboten sein wird kann per se nicht gegeben werden – es bleibt aber spannend zu sehen, was die Zukunft im Bereich der Künstlichen Intelligenz zu bieten hat.

Literaturverzeichnis

Kersting, K., Lampert, C. & Rothkopf, C. (Hrsg.). (2019). *Wie Maschinen lernen: künstliche Intelligenz verständlich erklärt.* Wiesbaden: Springer.

Mueller, J. P. & Massaron, L. (2020). *Deep Learning kompakt für Dummies* (S. Linke, Übers.). Weinheim: WILEY-VCH Verlag. (Originalwerk veröffentlicht 2019)

Kaffka, T. (2017). *Neuronale Netze: Grundlagen; mit Beispielprogrammen in Java.* Frechen: Mitp.

Alpaydın, E. (2008). *Maschinelles Lernen* (S. Linke, Übers.). München: Oldenbourg Verlag. (Originalwerk veröffentlicht 2004)

Goertzel, B. & Wang, P. (2007). Advances in Artificial General Intelligence. Amsterdam: IOS Press.

Matrix. (o.D.). Abgerufen am 27. Dezember 2022, von https://simpleclub.com/lessons/mathematik-matrix

Stadler, M. L. (2022, 6. September). *Künstliche Intelligenz.* mindsquare AG. Abgerufen am 27. Dezember 2022, von https://mindsquare.de/knowhow/kuenstliche-intelligenz/

Wuttke, L. (2021, 19. Oktober). *Training-, Validierung- und Testdatensatz.* Köln: datasolut. Abgerufen am 27. Dezember 2022, von https://datasolut.com/wiki/trainingsdaten-und-testdaten-machine-learning/

Oppermann, A. (2021, 31. August). *Neuronale Netze und ihre Anwendungen*. Abgerufen am 27. Dezember 2022, von https://artemoppermann.com/de/neuronale-netze-und-ihre-anwendungen/

Brunton, S. (2019, 6. Juni). *Neural Network Architectures & Deep Learning. [Video].* Abgerufen am 27. Dezember 2022, von https://www.youtube.com/watch?v=oJNHXPs0XDk

Simplilearn. (2019, 19. Juni). *Neural Network In 5 Minutes | What Is A Neural Network? | How Neural Networks Work | Simplilearn. [Video].* Abgerufen am 27. Dezember 2022, von https://www.youtube.com/watch?v=bfmFfD2RIcg

Oppermann, A. (2021, 2. August). *Aktivierungsfunktionen in neuronalen Netzen: Sigmoid, tanh, ReLU.* Abgerufen am 27. Dezember 2022, von https://artemoppermann.com/de/aktivierungsfunktionen/

What is Q-Learning: The Best Guide To Understand Q-Learning. (2022, 29. August). Abgerufen am 27. Dezember 2022, von https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning

deeplizard. (2018, 6. Oktober). *Q-Learning Explained – A Reinforcement Learning Technique. [Video].* Abgerufen am 27. Dezember 2022, von https://www.youtube.com/watch?v=qhRNvCVVJaA

NeuralNine. (2021, 21. August). *Neural Network Python Project – Handwritten Digit Recognition. [Video].* Abgerufen am 27. Dezember 2022, von https://www.youtube.com/watch?v=bte8Er0QhDg

Module: tf. (2022, 26. Oktober). Abgerufen am 27. Dezember 2022, von https://www.tensorflow.org/api_docs/python/tf

Matplotlib.pyplot. (o.D.) Abgerufen am 27. Dezember 2022, von https://matplotlib.org/stable/api/pyplot_summary.html

Gondlach, K. (2021, 15. April). *Wie wird künstliche Intelligenz die Zukunft verändern.* Abgerufen am 27. Dezember 2022, von https://www.kaigondlach.de/artikel/wie-wird-kuenstliche-intelligenz-die-zukunft-veraendern/

Abbildungsverzeichnis

Eidesstattliche Erklärung

Ich, Sarah Götz, erkläre hiermit eidesstattlich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die vorliegende Arbeit wurde bisher in gleicher oder ähnlicher Form noch nicht eingereicht.

13.02.2023, Sarah Götz

Datum, Unterschrift