

# **Bildähnlichkeitserkennung von Markenlogos mithilfe von Machine Learning**

Untertitel

**Masterarbeit**

Eingereicht in teilweiser Erfüllung der Anforderungen zur Erlangung des akademischen Grades:

**Master of Science in Engineering**

an der FH Campus Wien

Studienfach: Software Design and Engineering

**Autor:**

David Walser

**Matrikelnummer:**

01609388

**Betreuer:**

FH-Prof. DI Dr. Igor Miladinovic

**Datum:**

02.02.2022

Erklärung der Urheberschaft:

Ich erkläre hiermit diese Masterarbeit eigenständig verfasst zu haben. Ich habe keine anderen Quellen, als die in der Arbeit gelisteten verwendet, noch habe ich jegliche unerlaubte Hilfe in Anspruch genommen

Ich versichere diese Masterarbeit in keinerlei Form jemandem Anderen oder einer anderen Institution zur Verfügung gestellt zu haben, weder in Österreich noch im Ausland.

Weiters versichere ich, dass jegliche Kopie (gedruckt oder digital) identisch ist.

Datum:

Unterschrift:

# Abstract

(E.g. “This thesis investigates...”)

# Kurzfassung

(Z.B. "Diese Arbeit untersucht...")

# Abkürzungen

ÖPA	Österreichisches Patentamt
KNN	künstliches neuronales Netzwerk
CNN	convolutional neural network
ARP	Address Resolution Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
WLAN	Wireless Local Area Network

# Schlüsselbegriffe

GSM

Mobilfunk

Zugriffsverfahren

# Inhaltsverzeichnis

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Problembeschreibung . . . . .	1
1.2	Motivation und Ziel . . . . .	1
<b>2</b>	<b>Machine learning</b>	<b>2</b>
2.1	Supervised Machine Learning . . . . .	3
2.2	Unsupervised Machine Learning . . . . .	4
2.3	Semi-supervised Learning . . . . .	5
2.4	Reinforcement Learning . . . . .	6
2.5	Neuronale Netzwerke . . . . .	7
<b>3</b>	<b>Bildbezogenes Machine Learning</b>	<b>13</b>
3.1	Wie computer sehen . . . . .	13
3.2	Bilderkennung und Klassifizierung . . . . .	15
3.3	Algorithmen für Bildähnlichkeitserkennung . . . . .	16
<b>4</b>	<b>Prototyp zur Bildähnlichkeitserkennung von Markenlogos</b>	<b>23</b>
4.1	Daten . . . . .	23
4.2	Bildvorverarbeitung . . . . .	25
4.3	Algorithmus . . . . .	25
<b>5</b>	<b>Diskussion der Ergebnisse</b>	<b>27</b>
<b>6</b>	<b>Conclusio</b>	<b>28</b>
<b>7</b>	<b>Ausblick</b>	<b>29</b>
	<b>Bibliographie</b>	<b>30</b>
	<b>Abbildungen</b>	<b>35</b>
	<b>Tabellen</b>	<b>36</b>
	<b>Appendix</b>	<b>37</b>

# 1 Einführung

## 1.1 Problembeschreibung

Für uns Menschen ist es eine ziemlich einfache Aufgabe zu ermitteln ob ein Bild ähnlich zu einem anderen ist oder nicht. Wir erkennen alle Möglichen Merkmale eines Bildes, wie Farben, Texte oder Muster, ohne große Schwierigkeiten. Es stellt jedoch eine Herausforderung dar, wenn ein Bild mit 400.000 anderen Bildern verglichen und auf Ähnlichkeit geprüft werden soll.

Im Jahre 2020 wurden 6260 neue Marken beim österreichischen Patentamt angemeldet [1]. Die meisten dieser Marken werden in Kombination mit einem Bild, auch Logo genannt, registriert. Damit es bei einer Neuanschuldung nicht zu einer unmittelbaren Verwechslungsgefahr mit bereits bestehenden Marken kommt, bietet das österreichische Patentamt einen Dienst an, bei dem Daten zu einer Marke angegeben werden, und überprüft wird, ob es Ähnlichkeiten mit bereits angemeldeten Marken gibt [2]. Ein Hauptbestandteil dieser Ähnlichkeitsrecherche ist die Überprüfung von Ähnlichkeiten der Logos.

## 1.2 Motivation und Ziel

Machine Learning ist ein aufkommendes und zukunftsweisendes Thema. Laut einer Studie aus 2017 wurde ein Wachstum des machine learning Marktes von 1.03 Milliarden \$ in 2016 auf 8.81 Milliarden \$ im Jahre 2022 erwartet, was einer Wachstumsrate von 44.1% entspricht [3]. Eines der Hauptprobleme im Bereich Computer Vision ist es ähnliche Bilder in großen und unstrukturierten Kollektionen zu finden. Es kann ein sehr zeitaufwändiges Verfahren sein, da hier viele Bilder getestet werden müssen um eine Übereinstimmung zwischen Paaren zu finden [?]. Technologisch gesehen ist der mit dieser Masterarbeit verbundene Prototyp eine große Herausforderung, da zur Umsetzung allerneuste Technologien und Frameworks benötigt werden. Eine weitere Herausforderung wird es sein, wie genau die vom österreichischen Patentamt dankenswerter Weise zur Verfügung gestellten Daten zu analysieren und kategorisieren sind. Ziel dieser Masterarbeit ist es das Patentamt bei der Ähnlichkeitsrecherche zu unterstützen, in dem ein Prototyp entwickelt wird, welcher Ähnlichkeiten bei Bildern erkennt. Außerdem soll diese Masterarbeit einen Überblick über Machine Learning, mit Fokus auf Bildverarbeitung, enthalten. Daraus ergibt sich die folgende Forschungsfrage

Welche Unterschiede weisen verschiedene ML Algorithmen auf, im Bezug auf Erkennungsrate einer Ähnlichkeitsüberprüfung von Bildern?

zu beantworten.

tbd. more!



## 2 Machine learning

Für viele Firmen ist machine Learning bereits der meist benutzte Bereich aus dem Feld der künstlichen Intelligenz [4]. Machine Learning ist die Wissenschaft und Kunst Computern das Lernen anhand von Daten zu ermöglichen [5] und ist ein Anwendungsgebiet von künstlicher Intelligenz welches bereits seit vielen Jahren die Forschung und Wirtschaft unterstützt [6]. "Machinelles Lernen ist ein Oberbegriff für die "künstliche"Generierung von Wissen aus Erfahrung: Ein künstliches System lernt aus Beispielen und kann diese nach Beendigung der Lernphase verallgemeinern." [7] Muster und Gesetzmäßigkeiten werden aus den Trainingsdaten erkannt, woraus ein statistisches Modell erzeugt wird [7]. Dieser Prozess wird als Modelltraining bezeichnet und ist ein iterativer Prozess, welcher oft mehrfach durchlaufen wird, bis die Qualität des Ergebnis zufriedenstellend ist [6]. Aus dem Erlernen und Analysieren der historischen Daten kann ein Ergebnis für neue und unbekannte Daten prognostiziert werden, ohne explizit darauf programmiert zu sein [8].

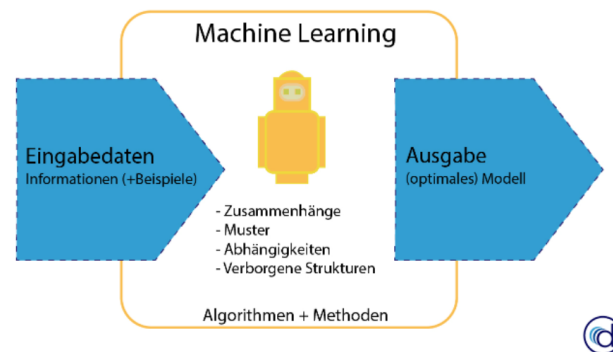


Abbildung 2.1: Machine Learning nimmt Eingabedaten mit Beispielen und lernt daraus, um für die Zukunft Prognosen zu machen [6]

Ein Alltag ohne dem Interagieren mit Machine Learning ist heutzutage kaum mehr wegzudenken. Bei der Benutzung von sozialen Medien, online Shopping oder Bankdiensten kommt Machine Learning zum Einsatz [9] und bereits seit den 1990er Jahren beeinflusst Machine Learning in Form des Spam Filters das Leben vieler [5]. Netflix bietet mithilfe von Machine Learning personalisierte Film und Serienempfehlungen an, und zusätzlich unterstützt Machine Learning bei der Optimierung der Produktion von Filmen und TV Shows [10]. Facebooks Algorithmus kann bereits mit 100 bis 150 Likes die Persönlichkeit einer Person genauer beschreiben als deren Familienmitglieder oder Freunde [11]. Die Grundlage für Machine Learning bilden Algorithmen, welche sich in folgende Arten aufteilen lassen [9]:

- supervised learning,
- unsupervised learning,
- semi-supervised learning und
- reinforcement learning.

## 2.1 Supervised Machine Learning

Supervised Machine Learning, im Deutschen übersetzt überwachtes Lernen, ist die am häufigsten angewendete Algorithmusart [9]. Ähnlich wie in der Schule wenn unter Aufsicht des Lehrers oder der Lehrerin geprüft wird, ob ein Problem richtig oder falsch gelöst wird, ist die Situation bei supervised Machine Learning Algorithmen. Dem Algorithmus wird ein gelabelter Datensatz zum Lernen zur Verfügung gestellt, somit weiß der Algorithmus für jeden Datensatz die richtige Lösung [12]. Ein gelabelter Datensatz kann z.B. ein Bild von einem Tier sein, wobei hier zusätzlich auch die Information über ein Feature mitgegeben wird, wie z.B. die Art des Tieres oder das Gewicht des Tieres (siehe Abbildung 2.2). Das Label ist die Information über ein Feature welche der Algorithmus später vorhersagen will [13]. Folgende

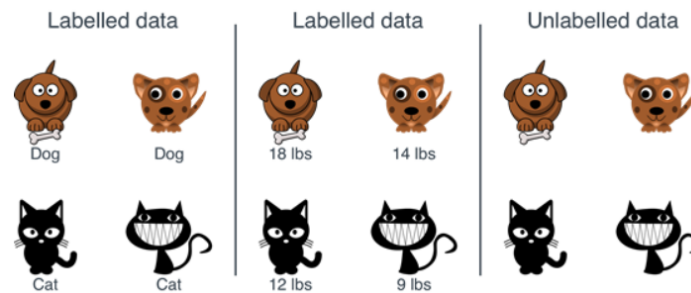


Abbildung 2.2: Labeled vs unlabeled Data [13]

Algorithmen sind Beispiele für supervised Learning [5]:

- k-Nearest Neighbors
- Naive Bayes
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees und Random Forests
- Neuronale Netzwerke (wobei diese auch unsupervised oder semisupervised sein können)

Klassifizierung und Regression sind klassische Anwendungsgebiete für supervised Learning [5].

### 2.1.1 Klassifizierung

Bei Klassifizierungsproblemen geht es darum einen Status vorherzusagen [13], wie z.B. zu welcher Klasse oder Gruppe die Inputdaten gehören [12]. Der Spam Filter ist ein gutes Beispiel für Klassifizierung, hierbei handelt es sich um zwei Klassen: Spam und nicht Spam. Der Algorithmus bekommt E-Mails zum Lernen, welche als Spam E-Mail oder normale E-Mail gelabelt sind, um somit für neue E-Mails herauszufinden, ob diese Spam E-Mails sind [5].

### 2.1.2 Regression

Regression ist ein Fachgebiet der Statistik und ist eine Methode um die Beziehung zwischen unabhängigen Variablen oder Merkmalen und einer abhängigen Variable oder einem Ergebnis zu verstehen. Sobald die Beziehung zwischen unabhängigen und abhängigen Variablen

geschätzt wurde, können die Ergebnisse vorhergesagt werden [14]. Bei Regressionsproblemen geht es um kontinuierliche Daten [12] und darum eine Zahl vorherzusagen, wie z.B. das Vorhersagen von Aktienkursen [13] oder von Grundstückspreisen [12].

### 2.2 Unsupervised Machine Learning

Bei unsupervised Machine Learning, im Deutschen übersetzt unüberwachtes Lernen, handelt es sich um den Ansatz mit Datensätzen zu lernen, welche kein Label besitzen [5]. Dies wird auch als selbst organisiertes Lernen bezeichnet [12], da der Algorithmus ohne einen Lehrer oder eine Lehrerin lernt [5]. Da es sich hierbei um ungelabelte Datensätze handelt, besitzen diese nicht die Zielinformation, welche vorhergesagt werden soll [13] und somit werden Zusammenhänge in den Daten [15], versteckte Muster oder Datengruppierungen von unsupervised Algorithmen erkannt, ohne dass menschliches Eingreifen erforderlich ist [14]. Die Fähigkeit,

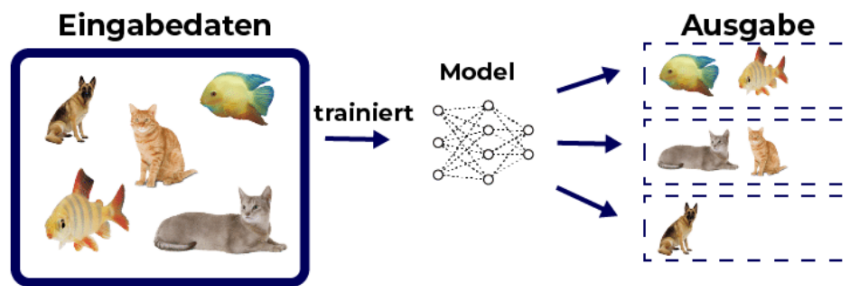


Abbildung 2.3: "Model trainiert ohne Zielvariable und findet eigenständig Muster und Zusammenhänge in den Daten"[15]

Ähnlichkeiten und Unterschiede in Informationen zu entdecken, macht sie zur idealen Lösung für die explorative Datenanalyse, Kundensegmentierung und Bilderkennung. Folgende Algorithmen sind Beispiele für unsupervised Learning [5]:

- K-Means
- DBSCAN
- Hierarchical Cluster Analysis (HCA)
- Principal Component Analysis (PCA)
- Kernel PCA
- Locally-Linear Embedding (LLE)
- Apriori
- Eclat

Clustering, Assoziationsanalyse und Dimensionalitätsreduktion sind die drei Hauptaufgaben von unsupervised Machine Learning [14].

#### 2.2.1 Clustering

Beim Clustering geht es darum, die Population oder die Datenpunkte in eine Reihe von Gruppen aufzuteilen, so dass die Datenpunkte in denselben Gruppen anderen Datenpunkten in derselben Gruppe ähnlicher und den Datenpunkten in anderen Gruppen unähnlicher sind [16]. Es werden ungelabelte Daten auf der Grundlage ihrer Ähnlichkeiten oder Differenzen in verschiedene Gruppen aufgeteilt [14]. Anwendungsgebiete für Clustering sind beispielsweise

Bücher die je nach Titel und Inhalt in verschiedene Gruppen eingeteilt werden oder Pflanzen und Tiere welche in Spezies aufgeteilt werden (siehe Abbildung 2.3) [16].

### 2.2.2 Assoziationsanalyse

Die Assoziationsanalyse ist eine Methode um herauszufinden welche Muster (Beziehungen, Korrelationen, Strukturen, ...) es in den Datensätzen gibt [17]. Diese Methode wird häufig für Warenkorbanalysen verwendet, um ein besseres Verständnis über die Beziehung zwischen den Produkten zu bekommen. Beispiele davon sind z.B. auf Amazon die "Kunden, die diesen Artikel gekauft haben, kauften auch:" Anzeige oder die Spotify "Discover Weekly" Playlist. [14]

### 2.2.3 Dimensionalitätsreduktion

Während mehr Daten grundsätzlich zu genaueren Ergebnissen führen können, so kann dies auch die Leistung von Algorithmen für machine Learning beeinträchtigen (z.B. overfitting) und die Visualisierung von Datensätzen erschweren [14]. Bei der Dimensionalitätsreduktion geht es darum, die Variablen in den Daten auf die wesentlichen und zielführenden zu beschränken (z.B. werden redundante oder noise Features entfernt) [15], wobei die Integrität der Daten so weit wie möglich erhalten bleiben soll [14]. Es kann für das Bereinigen von Daten oder auch für das Hervorheben von Features verwendet werden, da die Daten dabei von hochdimensionalem Feature-Raum in niedrigdimensionalen Feature-Raum transformiert werden [18]. Ein Beispiel für Dimensionalitätsreduktion könnte die Reduktion von zweidimensionalen Eingabedaten in eindimensionale umzuwandeln.

## 2.3 Semi-supervised Learning

Um die Schwierigkeiten vom Erstellen von großen gelabelten Datensätzen entgegenzuwirken gibt es eine Methode bei der ein kleiner Anteil gelabelte Daten, der Großteil jedoch ungelabelte Daten sind. Diese Methode wird als semi-supervised Learning bezeichnet, welche die Benefits von unsupervised und supervised kombiniert [19]. Der Algorithmus wird initial mit den wenig gelabelten Daten trainiert und kann somit iterativ mehr und mehr an die ungelabelten Daten angewandt werden. Self-training und Co-training sind zwei Ansätze für semi-supervised learning. [20]

### 2.3.1 Self-training

Self-training ist eines der einfachsten Beispiele für semi-supervised learning und ist ein Prozedere einen supervised learning Ansatz in semi-supervised umzuwandeln [20]. Der straightforward Ansatz wird anhand folgendem Beispiel erklärt:

1. Die gelabelten Daten werden für das Trainieren des Modells herangezogen [21].
2. Dieses Modell wird dann für die Vorhersage von ungelabelten Daten verwendet [21].
3. Es werden Ergebnisse, welche dem zuvor bestimmten Kriterium entsprechen (z.B. >90% accuracy), mit pseudo-labels versehen und mit den bereits gelabelten Daten kombiniert [21].
4. Das Modell wird nun mit dem neuen Pool an gelabelten Daten trainiert [21].

5. Der ganze Prozess wird nun durchiteriert bis entweder alle Daten gelabelt sind oder die spezifizierte Maximalanzahl an Iterationen erreicht wird [21].

Die Performance von dem Self-training Ansatz variiert von Datensatz zu Datensatz und es gilt abzuwägen ob sie im Vergleich zu dem supervised Ansatz eine Verbesserung erzielt [20]. Ein Anwendungsfall für Self-training könnte z.B. das Erkennen von Bildrotationen sein. Es werden dabei z.B. Bilder in 90 Grad schritten rotiert, und durch Self-training wird herausgefunden um wieviel Grad es sich handelt.

### 2.3.2 Co-training

Co-training ist eine verbesserte Version von Self-Training, die zum Einsatz kommt, wenn nur wenig gelabelte Daten verfügbar sind [20]. Bei dieser Methode benötigt es zwei Sichten auf die Daten [22] und basierend darauf werden zwei individuelle Klassifizierer trainiert [20]. Die zwei Ansichten sind verschiedene Gruppen von Features, welche jeweils Informationen für jede Instanz beinhalten [20]. Dies bewirkt dass beide Ansichten unabhängig sind und sie alleinstehend die Klasse einer Instanz vorhersagen können [22]. Ein Anwendungsgebiet für Co-Training könnte die Klassifizierung von Webinhalten sein. Der Inhalt einer Webseite kann in zwei Ansichten aufgeteilt werden: eine mit Wörtern, und die andere mit Verweistexten der Links, welche zu dieser Seite führen [20]. Der Ablauf eines Co-Trainings sieht wie folgt aus:

1. Für jede Ansicht werden die separaten Models mit den gelabelten Daten trainiert [20].
2. Die ungelabelten Daten werden hinzugefügt und mit pseudo-labels versehen [20].
3. Die beiden Klassifizierer trainieren sich gegenseitig mit den pseudo-labels und es wird das höchste Konfidenzniveau angestrebt. Wenn z.B. der erste Klassifizierer das Label für einen Datensatz vorhersagt, während der andere einen Fehler macht, so werden die vom ersten Klassifizierer zugewiesenen pseudo-labels jene vom zweiten überschreiben und vice-versa. [20]
4. Zuletzt werden die beiden geupdaten Klassifizierer in zu einem kombiniert [20].

## 2.4 Reinforcement Learning

Reinforcement Learning, im Deutschen übersetzt Bestärkendes oder Verstärkendes Lernen, ist eine immer beliebter werdende Methode und befasst sich damit intelligente Lösungen für komplexe Steuerungsprobleme zu finden [23]. Das lernende System, welches in diesem Kontext *Agent* genannt wird, beobachtet das Umfeld und führt dann Aktionen aus, für die es entweder Belohnungen oder Bestrafungen bekommt [5]. Hierbei erlernt der Agent eigenständig welche Aktion für welche Situation am besten geeignet ist, um die Belohnungsfunktion zu maximieren [24]. Im Gegensatz zu supervised oder unsupervised learning werden beim Reinforcement Learning vorab keine Daten benötigt. Die Daten werden während den Trial-and-Error Trainingsdurchläufen der Simulationsumgebung generiert und gelabelt. Das Ziel von Reinforcement Learning ist es eine bestmögliche Policy zu erreichen [23]. Als Policy wird die Strategie bezeichnet, welche der Agent ausführt und gibt an welche Aktion als nächstes ausgeführt werden soll [24] um die Belohnung zu maximieren [23]. Reinforcement Learning anhand eines Beispiels: Die Katze (siehe Abbildung ??), welche in diesem Beispiel der Agent ist, befindet sich im Haus (Umgebung) und kann die zwei Zustände sitzen oder bewegen haben. Wenn die Katze nun eine Aktion ausführt, so kann sie ihren Zustand ändern, wofür sie eine Belohnung oder eine Strafe erhalten kann.

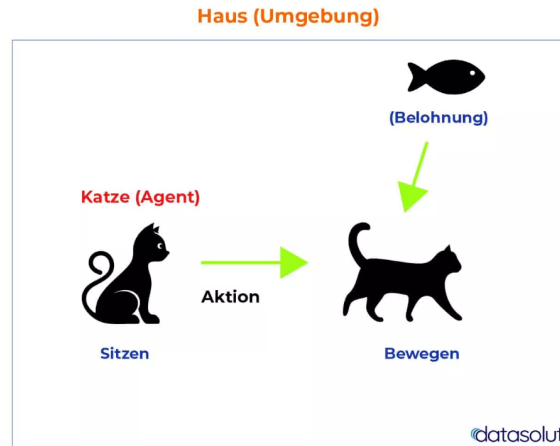


Abbildung 2.4: Beispiel von Reinforcement Learning [24]

## 2.5 Neuronale Netzwerke

Die Anfänge von künstlichen neuronalen Netzwerken machten 1943 der Neuropsychologe Warren MCCulloch und der Mathematiker Walter Pitts. Sie stellten ein vereinfachtes Computermodell vor, welches wie biologische Neuronen in Tiergehirnen zusammenarbeitet [5], um komplexe Aufgaben aus den Bereichen Statistik, Wirtschaft und Informatik lösen zu können [25]. Dies gilt als erste Architektur für künstliche neuronale Netzwerke und seither wurden viele weitere Architekturen entwickelt [5]. "Künstliche neuronale Netze, auch künstliche neuronale Netzwerke, kurz: KNN (englisch artificial neural network, ANN), sind Netze aus künstlichen Neuronen." [26] Der Ursprung von künstlichen Neuronen (veranschaulicht in Abbildung 2.5) sind die biologische Neuronen, welche in tierischen Gehirnen zu finden sind.

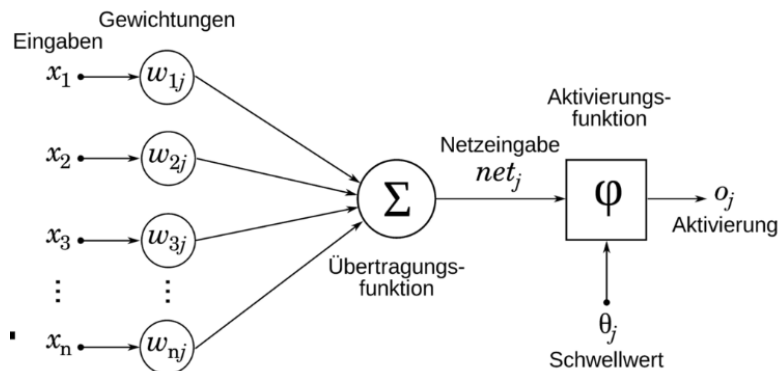


Abbildung 2.5: Aufbau eines Neurons [25]

Neuronen kommunizieren in dem sie kurze elektrische Impulse über Synapsen versenden, bzw. empfangen. Erhält ein Neuron eine ausreichende Anzahl an Signalen innerhalb weniger Millisekunden, so feuert dies eigene Signale weiter. Die einzelnen Neuronen sind in einem Netzwerk mit Milliarden von anderen Neuronen verbunden, in dem sie in fortlaufenden Layern eingegliedert sind (siehe Abbildung 2.6) [5].

Künstliche neuronale Netzwerke spiegeln das Verhalten des menschlichen Gehirns wieder und ermöglichen es Computerprogrammen selbständig Muster zu erkennen und allgemeine Probleme zu lösen [27]. Künstliche neuronale Netzwerke können verschiedene Datenquellen

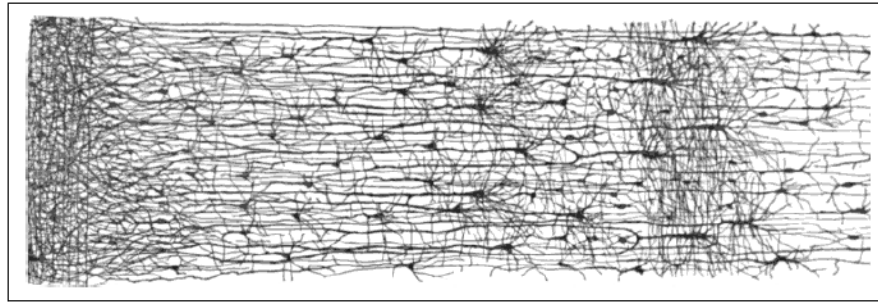


Abbildung 2.6: Multiple Layer in einem biologischen neuronalen Netzwerk [5]

wie Geräusche, Bilder, Texte, Tabellen oder Zeitreihen interpretieren, woraus sie Informationen extrahieren und Muster erkennen um später auf unbekannte Daten Vorhersagen treffen zu können. Grundsätzlich weisen künstliche neuronale Netzwerke die Strukturen gerichteter Graphen auf, können aber je nach Komplexität unterschiedlich aufgebaut sein [25].

Angelehnt an die Struktur eines biologischen neuronalen Netzwerkes besteht ein künstliches neuronales Netzwerk ebenfalls aus Neuronen, welche in Layern (Schichten) angeordnet sind. Ein solches Netzwerk besteht aus einem Input-Layer, einem (oder mehreren) Hidden-Layer und einem Output-Layer (veranschaulicht in Abbildung 2.7), wobei jeder dieser Layer eine Vielzahl an Neuronen beinhaltet. Desto komplexer das zu lösende Problem ist, umso mehr Layer werden benötigt [28]. Der Input-Layer nimmt die eingegebenen Daten entgegen, verarbeitet und gewichtet diese, bevor sie an den Hidden-Layer weiter gegeben wird [25]. Der Hidden-Layer befindet sich zwischen Input und Output-Layer [25] und kann aus einem oder mehreren Layern bestehen und ist der Layer, in dem die meisten Berechnungen stattfinden [29]. Auch in diesem Layer finden wieder Gewichtungen statt. Sind mehrere Layer im Hidden-Layer vorhanden, so geschieht das Gewichten pro Layer [25]. Die Ausgabewerte des letzten Hidden-Layers werden dem Output-Layer als Eingabewerte übergeben [29]. Der Output-Layer ist die letzte Schicht und beinhaltet die Entscheidung oder Vorhersage des neuronalen Netzwerkes [25]. Eine angemessene Anzahl an Neuronen und eine passende Aktivie-

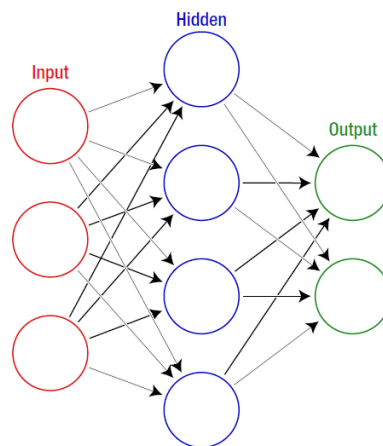


Abbildung 2.7: Einfache Veranschaulichung eines neuronalen Netzwerkes [30]

rungsfunktion sollte je nach Aufgabe gewählt werden. In den Neuronen werden Berechnungen getätigt [31] und jedes Neuron hat ein zugeordnetes Gewicht, wodurch sie unterschiedliche Wichtigkeiten erhalten [28]. Im biologischen Kontext können Gewichte mit Synapsen vergli-

chen werden [32]. Kombiniert mit einer Übertragungsfunktion entscheidet das Gewicht über den Input eines Neurons [28]. Bei den künstlichen neuronalen Netzwerken kombiniert ein Neuron die Eingangssignale mit zugeordneten Gewichten, welche die Bedeutung des Signals schwächen oder verstärken. Nach dem Aufsummieren wird das Ergebnis durch eine Aktivierungsfunktion gegeben um zu berechnen ob, und mit welchem Ausmaß das Ausgangssignal weitergesendet wird. Das Neuron wird als aktiviert bezeichnet, wenn es ein Ausgangssignal versendet [31].

### 2.5.1 Arten von neuronalen Netzwerken

Es gibt viele verschiedene Arten von künstlichen neuronalen Netzwerken und es werden nun ein paar davon etwas näher erläutert.

#### Preceptron

Preceptron ist ein neuronales Netzwerk mit nur einem Layer [33], welches im Bereich von supervised Learning Anwendung findet [34]. Es ist ein linearer oder binärer Klassifizierer und wird verwendet um Daten in zwei Gruppen zu klassifizieren [33]. Ein Preceptron funktioniert folgendermaßen:

1. Alle Eingangssignale werden mit ihren Gewichten multipliziert
2. Nach dem multiplizieren wird alles Aufsummiert
3. Abschließend wird eine Aktivierungsfunktion angewendet, welche den Wert auf den gewünschten Wert, wie z.B. (0,1) mapt. [33]

#### Feed forward neuronale Netzwerke

Ein Preceptron ist eine Form von Feed forward neuronalen Netzwerken, welche die einfachste Form von neuronalen Netzwerken sind [35]. Informationen werden nur in eine Richtung, von Input-Layer über Hidden-Layer zum Ausgangs-Layer verarbeitet. Abbildung 2.7 veranschaulicht ein feed forward neuronales Netzwerk und die funktionsweise ist gleich wie bei Preceptrons. Wenn es viele Layer zwischen dem Input und Output-Layer gibt, so wird es auch als deep feed forward neural network bezeichnet. [36]

#### Convolutional neuronale Netzwerke

Convolutional neural networks sind eines der beliebtesten deep neural networks [37]. Der Name kommt von der mathematischen Linearfunktion convolution, welche auf Matrixen angewandt wird [37]. CNNs sind effizient mit 2D und 3D Eingabedaten und werden unter anderem für Objektdetektion bei Bildern verwendet [25]. Die Architektur ist unterschiedlich zu klassischen neuronalen Netzwerken, mehr dazu in Kapitel 3.3.2.

#### Recurrent neuronale Netzwerke

Wenn es um sequentielle oder zeitliche Daten geht so können feed forward neuronale Netzwerke nicht verwendet werden. Es wird ein Mechanismus benötigt um historische Daten zu speichern und deshalb wurden Recurrent neuronale Netzwerke entwickelt. In klassischen feed forward neuronalen Netzwerken sind alle Datenpunkte unabhängig von einander, was mit sequentiellen Daten, welche von vorherigen Daten abhängig sind, nicht funktioniert [38].



Recurrent neuronale Netzwerke haben ein Konzept mit dem sie Informationen über vorherige Daten speichern können. Dieses Konzept wird Memory genannt. [39]

### 2.5.2 Training bei Neuronalen Netzwerken

Ein Vorteil von künstlichen neuronalen Netzwerken ist, dass beim Erstellen nicht alle Parameter spezifiziert werden müssen, da der Algorithmus anhand von Beispielen von selbst lernt [40]. Neuronale Netzwerke können grundsätzlich für jedes komplexe Problem verwendet werden, bei dem es sich um funktionale Beziehungen handelt. Es ist hierbei nicht notwendig die Art der Beziehung zu kennen, da das neuronale Netzwerk durch das Beobachten und iterativem anpassen der Parameter lernt [41].

Um zu verstehen wie gut ein solches Netzwerk für eine bestimmte Aufgabe performt kann die Verlustfunktion verwendet werden. Zu Beginn wird ein neuronales Netzwerk mit zufälligen Gewichten versehen, was in einem schlecht performenden neuronalem Netzwerk resultiert. Das Ziel ist es die hohe Verlustfunktion durch das Trainieren des neuronalen Netzwerkes in eine niedrige zu verwandeln [42]. Eine Methode für das fine-tuning von neuronalen Netzwerken ist Backpropagation. Dabei handelt es sich um eine Methode zur Anpassung der Gewichte eines neuronalen Netzwerkes, welche als Grundlage die Fehlerquote der vorangegangenen Iterationen (bei neuronalen Netzwerken Epochen genannt) hernimmt. Durch das richtige adaptieren der Gewichte kann die Fehlerquote verringert und die Genauigkeit des Models erhöht werden [43]. Gewichte verstärken oder schwächen die Signale der Neuronen und geben die Wichtigkeit eines bestimmten Bereiches der Daten an. Ein Neuron kann ein bestimmtes Muster erkennen und die Bedeutung dieses Muster für das Gesamtkonstrukt entscheidet sich je nach Gewicht [44].

### 2.5.3 Anwendungsbeispiel eines neuronalen Netzwerkes

Um ein besseres Verständnis für neuronale Netzwerke zu bekommen folgt nun ein Anwendungsbeispiel mit kleinen Codesnipets. Ziel ist es ein neuronales Netzwerk aufzubauen, welches ein Bild einer handgeschriebenen Zahl richtig klassifizieren kann. Hierfür wird der MNIST Datensatz <sup>1</sup> verwendet, welcher 60000 28 mal 28 Pixel große Graustufenbilder von handgeschriebenen Zahlen von 0 bis 9 beinhaltet. Um den MNIST Datensatz zu laden wird Tensorflow und Keras verwendet (siehe Listing 2.1).

---

```
from tensorflow.keras.datasets import mnist
# load dataset
(trainX, trainy), (testX, testy) = mnist.load_data()
```

---

Listing 2.1: MNIST Datensatz in Python laden

Die Bilder aus dem MNIST Datensatz sind in Abbildung 2.8 zu sehen. Die Architektur des neuronalen Netzwerkes hat 784 (28 mal 28) Neuronen im Input-Layer. Somit kann jeder Pixel eines Bildes an ein Neuron als Eingabedaten übergeben werden. Der Hidden-Layer beinhaltet 15 Neuronen, wobei die Anzahl hier variieren kann. Abgeleitet von den vorhandenen Klassen (0 bis 9) ergeben sich 10 Neuronen im Output-Layer. Um diese Architektur mithilfe von Keras und Tensorflow in Python nachzubilden können die Codezeilen aus dem Listing 2.2 verwendet werden.

---

```
from keras.layers import Dense
```

---

<sup>1</sup>MNIST Datensatz erhältlich unter <http://yann.lecun.com/exdb/mnist/>

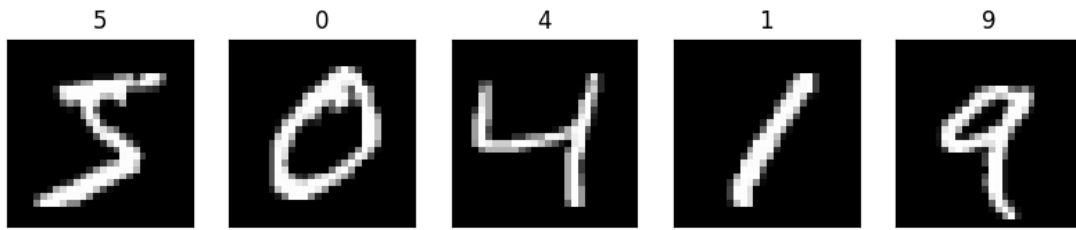


Abbildung 2.8: Beispielsbilder aus dem MNIST Datensatz

```
from keras.models import Sequential
image_size = 784 # 28*28 Pixel
num_classes = 10 # Zahlen von 0 bis 9
model = Sequential()
model.add(Dense(units=15, activation='sigmoid', input_shape=(
    image_size,)))
model.add(Dense(units=num_classes, activation='softmax'))
model.summary()
```

Listing 2.2: Architektur des Neuronalen Netzwerkes

Dense bedeutet dass es sich um ein vollständig verbundenes Netzwerk handelt, was bedeutet dass jedes Neuron des Input-Layers mit jedem Neuron aus dem nächsten Layer verbunden ist. Dies ist in Abbildung 2.9 zu sehen, welche die Beschriebene Architektur des neuronalen Netzwerkes veranschaulicht. Vor dem Compilieren und Trainieren des neuronalen Netzwerkes

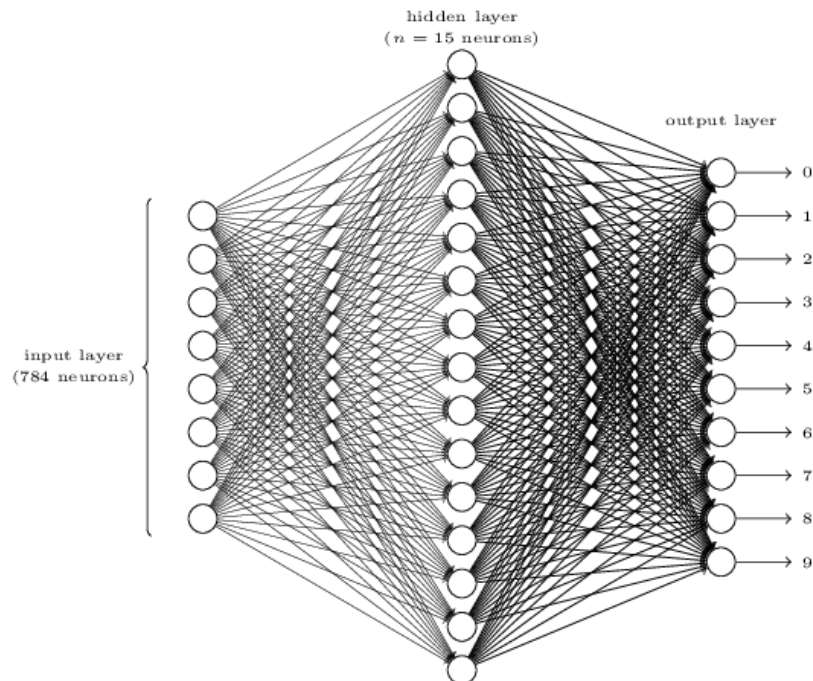


Abbildung 2.9: Architektur des Neuronalen Netzwerkes [45]

werden die Testdaten noch modifiziert, was im kompletten Codebeispiel im Anhang 1 zu finden ist.

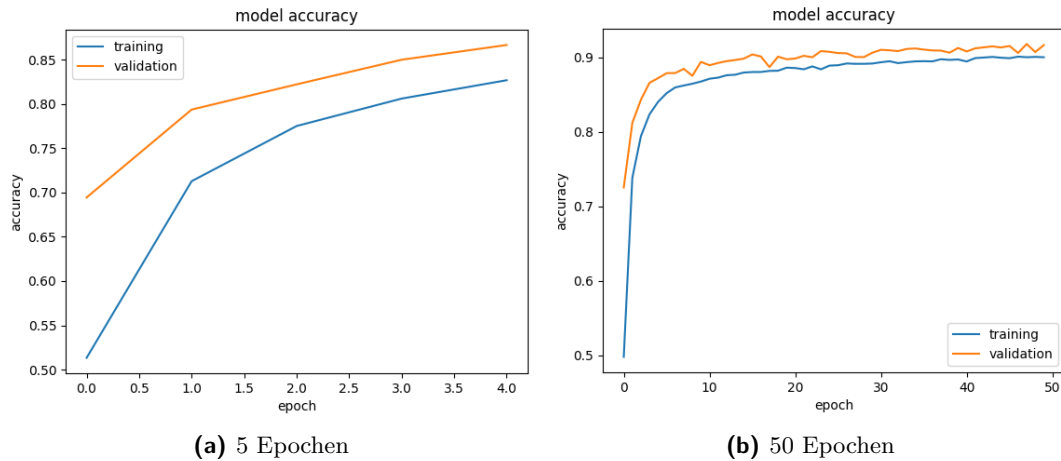


Abbildung 2.10: Vergleich 5 und 50 Epochen

---

```
model.compile(loss='categorical_crossentropy', optimizer='sgd',
              metrics=['acc'])
history = model.fit(x_train, y_train, batch_size=128, epochs=5,
                   verbose=False, validation_split=.1)
loss, accuracy = model.evaluate(x_test, y_test, verbose=False)
```

---

Listing 2.3: Model compilen und erlernen

Als Verlustfunktion wird hier die categorical crossentropy verwendet, da diese für Mehrklassenklassifizierung, bei der ein Datensatz genau einer Klasse angehört, gut geeignet ist [46]. Stochastic Gradient Descent gehören zu den einfachsten Optimierer, weshalb sie in diesem Beispiel für das Model gewählt wurden. Für das Trainieren des Algorithmus wurden zum Vergleich einmal 5 und einmal 50 Epochen ausgewählt. Wie in Abbildung 2.10 zu sehen ist erreicht das Model nach 5 Epochen eine Genauigkeit von etwas mehr als 80%, während es nach 50 Epochen eine Genauigkeit von fast 90% erreicht. Die Architektur und der Aufbau des neuronalen Netzwerkes könnte noch deutlich verändert und optimiert werden, jedoch dient dieses Beispiel zum Verständnis.

## 3 Bildbezogenes Machine Learning

Die Digitalisierung und Automatisierung von Abläufen nimmt auf der ganzen Welt rasant zu. Bilverarbeitung im Kontext von Computern und machine Learning bezieht sich darauf wichtige und bedeutende Informationen aus Bildern zu extrahieren. Da Bilder mittels unstrukturierten Daten repräsentiert werden, ist es weitaus schwieriger diese im Gegensatz zu strukturierten Daten wie Tabellen zu interpretieren. Je nach Anwendung gibt es unterschiedliche Methoden und Algorithmen, welche verwendet werden können. Der große Bereich von Computer Vision, welcher zudem stetig wächst, wird bereits in verschiedenen Bereichen wie Video Verarbeitung, Gesichtserkennung, Medizinisches Umfeld, Robotic Vision, Selbstfahrende Autos und vieles mehr eingesetzt [47]. Während Tesla bereits autonome und selbständig fahrende Autos verkauft, hat Apple Ende 2021 einen Techniker von Tesla angeheuert und plant 2024 selbstfahrende Autos auf den Markt zu bringen [48]. Bildverarbeitung und Bilderkennung spielen hierbei eine große Rolle, da Objekte wie Fahrspur, Ampel und andere Verkehrsteilnehmer richtig und zeitgerecht erkannt werden müssen [47].

Auch im medizinischen Bereich hat machine Learning einen großen Einfluss. Neuronale Netzwerke, insbesondere deep learning Methoden wie Convolutional neural Networks (siehe Kapitel 3.3.2), werden bereits für medizinische Bildanalyse wie Bildsegmentierung, Klassifizierung und Problemvorhersagungen verwendet und unterstützen dabei medizinische Experten [49]. Durch das Analysieren von Röntgenbildern der Brust können mithilfe von machine Learning COVID-19 Patienten, mit einer Genauigkeit von 96.78%, erkannt werden [50].

### 3.1 Wie computer sehen

Seit über 60 Jahren versuchen Wissenschaftler auf der ganzen Welt Maschinen beizubringen, wie sie bedeutungsvolle Informationen aus visuellen Daten extrahieren können [51]. Allgemein gilt Computer Vision als eines der komplexesten Gebiete im Bereich von künstlicher Intelligenz [52]. Bereits im Jahre 1959 beschäftigten sich die Neurophysiologen David Hubel und Torsten Wiesel mit dem Thema Computer Vision, welches ein Teilbereich von künstlicher Intelligenz ist, und den Zweck verfolgt Systemen das erkennen von bedeutsamen Informationen aus Bildern oder Videos zu extrahieren [53, 51]. Ihre Publikation "Receptive fields of single neurons in the cat's striate cortex"[54] gilt als eines der Papers, welches am meisten Einfluss auf die Thematik genommen hat [53, 51]. Dabei beschreiben sie ihre Experimente, bei denen sie Elektroden an den primären Bereich des visuellen Kortex von anesthierten Katzen angeschlossen haben, und diese dabei beobachtet haben, während den Versuchstieren verschiedene Bilder gezeigt wurden. Durch einen glücklichen Zufall sind sie zur Erkenntnis gekommen, dass es sowohl einfache, als auch komplexe Neuronen im primären visuellen Kortex gibt, und dass visuelle Verarbeitung immer mit simplen Strukturen wie Kanten und Rändern beginnt. Dieser Ansatz findet Anwendung bei convolutional neural networks und ist im Wesentlichen das Grundprinzip von deep Learning [51]. Bevor es tiefer in diese Materie geht, wird im folgenden Unterkapitel kurz definiert und veranschaulicht was ein Bild ist und wie ein Computer solch eines wahrnimmt.

### 3.1.1 Was ist ein (digitales) Bild?

Ein digitales Bild repräsentiert Bildinhalte mittels ganzer Zahlen [55]. Es handelt sich dabei um eine digitale Repräsentation von etwas, das erstellt und kopiert werden kann und in elektronischer Form gespeichert wird [56]. Grundsätzlich wird bei digitalen Bildern zwischen zwei verschiedenen Arten unterschieden:

- **Rastergrafiken:** die Informationen werden in gleichmäßiger Rasterung gespeichert [55]. Das Bild besteht aus einer endlichen Anzahl an digitalen Bildelementen, welche auch als Pixel bekannt sind [57] und welche in Reihen und Spalten angeordnet sind [58]. Die Anzahl der Pixel auf Reihe und Spalte ergeben die Auflösung eines Bildes. Ist ein Bild 200 Pixel breit und 400 Hoch, so hat es z.B. eine Auflösung von 200x400 Pixeln [58].
- **Vektorgrafik:** die Informationen werden durch geometrische Objekte wie Punkte, Flächen oder Linien, definiert [55].

Es ist auch möglich Raster und Vektorgrafiken zu kombinieren und daraus ein digitales Bild zu generieren [57]. Um digitale Bilder für Menschen sichtbar zu machen benötigt es geeignete Anzeigegeräte wie Projektoren oder Computerbildschirme [55]. Farbige Bilder werden oft nach dem RGB-Modell repräsentiert. RGB steht für **R**ed **G**reen und **B**lue und jeder Pixel beinhaltet einen Mix aus diesen Farben, welche in drei Zahlen repräsentiert werden. In einem Graustufenbild beinhaltet ein Pixel nur eine Zahl, welche die Intensität oder den Anteil an Licht repräsentiert [58]. Ein Computer kann ein Bild nicht wie wir Menschen sehen, für Maschinen sind digitale Bilder nur Zahlen [59]. Durch das Analysieren der Anordnung der Zahlen ist es einem Computer möglich ein Bild zu interpretieren. Ein ausreichend große Menge an Daten gilt als große Schwierigkeit wenn es darum geht Computern das erkennen und klassifizieren zu erlernen. [52]

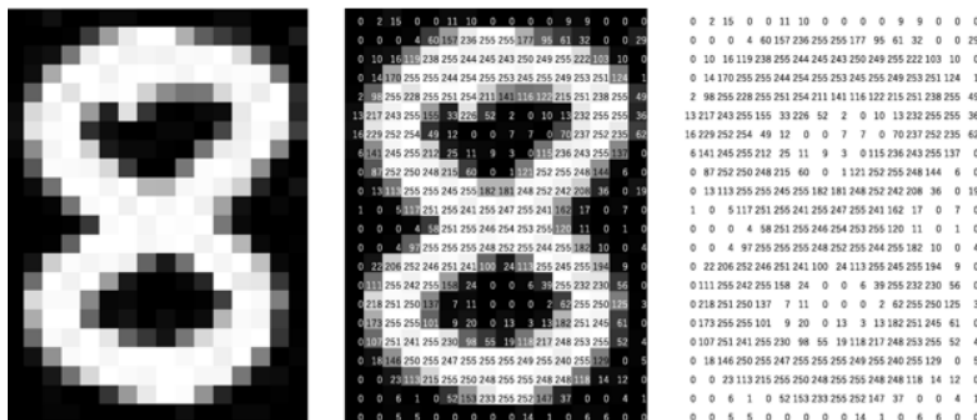


Abbildung 3.1: Repräsentation eines Bildes in Zahlen [59]

In Abbildung 3.1 ist ein Graustufenbild von der Zahl 8 aus dem MNIST Datensatz abgebildet. Hierbei ist ersichtlich wie ein Computer dieses Graustufenbild anhand von Zahlen zwischen 0 und 255, welche die Intensität repräsentieren, auffasst.

#### Dateiformate

Digitale Bilder können in verschiedenen Dateiformaten gespeichert werden. In Tabelle 3.1 sind einige davon beschrieben. Bei Bildern spielt Komprimierung eine Rolle, da so die Dateigröße

verringert werden kann. Es gibt zwei verschiedene Arten von Komprimierung, Lossless und Lossy. Bei der Lossless wird das Bild komprimiert ohne dabei einen Qualitätsverlust zu erleiden [60]. Das Bild wird komprimiert abgespeichert, sobald darauf zugegriffen wird, werden jedoch alle benötigten Informationen wiederhergestellt und das ursprüngliche Bild ausgegeben [30]. Eine Lossy Komprimierung bedeutet hingegen dass Teilinformationen verloren gehen [60]. Die Priorität bei dieser Komprimierung liegt dabei Speicherplatz zu sparen [30].

Dateiformat	Dateiendung	Komprimierung	Beschreibung
BITMAP	.bmp	keine	Bitmap wurde von Microsoft für Windows entwickelt. Es gibt bei diesem Dateiformat keine Komprimierung was jedoch in hohen Dateigrößen resultiert. [60]
JPEG	.jpg .jpeg	Lossy	JPEG steht für Joint Photographic Experts Group und sind im Internet sehr verbreitet und ein beliebtes Dateiformat für Digitalkameras [60].
GIF	.gif	Lossless	GIF bedeutet Graphic Interface Format und besitzen typischerweise sehr kleine Dateigrößen. Sie sind limitiert auf 256 Farben, können animiert werden und sind weit verbreitet im Web. [60]
PNG	.png	Lossless	PNG oder Portable Network Graphics wurde ursprünglich entwickelt um eine verbesserte Version des GIF Formates zu werden und dieses abzulösen. PNG Dateien können bis zu 16 Millionen Farben beinhalten. [60]

Tabelle 3.1: verschiedene Dateiformate

## 3.2 Bildererkennung und Klassifizierung

Um große Mengen an unstrukturierten Bilddaten effizient analysieren zu können, benötigt es fortschrittliche Techniken wie machine Learning [61]. Bildklassifizierung, welche oft auch als Bildererkennung bezeichnet wird [62], ist eines der Hauptprobleme im Bereich Computer Vision [63]. Die Aufgabe für Computer Vision hierbei ist es herauszufinden, was ein Bild beinhaltet und repräsentiert. Es wird als die Aufgabe bezeichnet, welche Gruppen von Pixeln oder Vektoren innerhalb eines Bildes nach bestimmten Regeln kennzeichnet und kategorisiert. Dabei wird die Bildklassifizierung in zwei Bereiche unterteilt, unsupervised und supervised. Bei der unsupervised Klassifizierung handelt es sich um eine vollautomatische Methode, bei der keine Trainingsdaten benötigt werden. Machine Learning Algorithmen werden verwendet um die Datensätze zu analysieren und zu clustern, indem sie ohne menschliches Eingreifen Muster erkennen. Beim supervised Ansatz wird mittels bereits klassifizierten Datensätzen trainiert, um so dann neue und unbekannte Datensätze zu klassifizieren. Applikationen zur Bildklassifizierung werden in Bereichen wie Medizinische Bildanalyse, Objekterkennung in Satellitenbildern, Ampelkontrollsystemen und vielen anderen Bereichen eingesetzt [61].

### 3.2.1 Herausforderungen bei maschineller Bilderkennung

In der folgenden Liste sind die Herausforderungen von maschineller Bilderkennung aufgezählt:

- **Viewpoint variation:** Ein Objekt kann aus verschiedenen Sichtpunkten und betrachtet werden.
- **Scale variation:** Ein Objekt der selben Klasse kann in unterschiedlichen Größen auftreten.
- **Deformation:** Ein Objekt kann verformt sein.
- **Occlusion:** Objekte können von anderen Objekten verdeckt sein, so dass nur wenige Pixel des Objektes sichtbar sind.
- **Illumination conditions:** Der Einfluss von Beleuchtung kann auf Pixelebene einen großen Einfluss haben.
- **Background clutter:** Ein Objekt kann sehr ähnlich zum Umfeld erscheinen und somit darin schwer erkennbar werden.
- **Intra-class variation:** Die Klasse eines Objektes kann sehr breit gefächert sein. Es gibt verschiedene Arten davon, welche ein eigenes Aussehen besitzen [64].

Beim Modellieren einer Bildklassifizierung gilt es darauf zu schauen, dass es gegenüber dem Kreuzprodukt all dieser Herausforderungen invariant ist, während es gleichzeitig die Empfindlichkeit gegenüber jeder einzelnen Variation beibehält [64]. In Abbildung 3.2 sind die oben genannten Herausforderungen bildlich veranschaulicht.

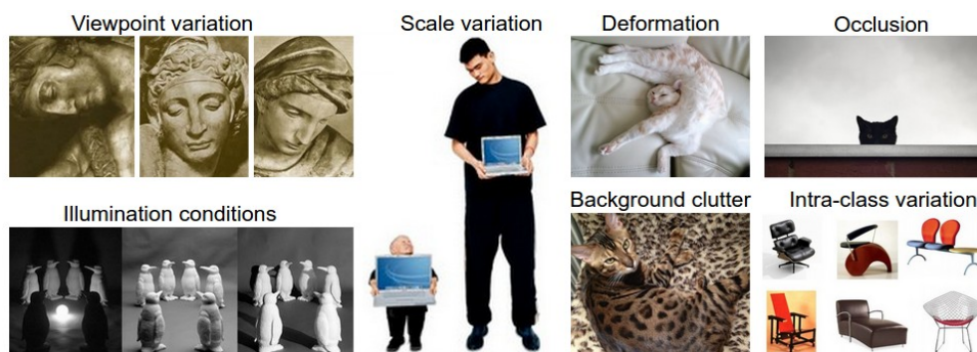


Abbildung 3.2: Beispiele für die Herausforderungen von maschineller Bilderkennung [64]

## 3.3 Algorithmen für Bildähnlichkeitserkennung

### 3.3.1 SIFT

Scale Invariant Feature Transform, kurz SIFT, wurde erstmals 2004 von David Lowe in seinem Paper "Distinctive Image Features from Scale-Invariant Keypoints"[65] veröffentlicht und ist eine Methode zur Überprüfung der Übereinstimmung von Bildern, welche auch zur Objekterkennung verwendet werden kann. Es handelt sich dabei um eine Methode die Merkmale aus Bildern extrahiert, welche für einen zuverlässigen Abgleich zwischen verschiedenen Ansichten eines Objekts oder einer Szene sorgen [65]. Die Extrahierung der Merkmale von Bildern erfolgt vor dem Abgleich unabhängig voneinander [66]. Mit effizienten Algorithmen

kann eine große Anzahl von Merkmalen aus typischen Bildern extrahiert werden. Diese Merkmale sind gegenüber Skalierung, Rotation, und Beleuchtungsänderung invariant und weisen eine Robustheit gegenüber Verzerrung und dem Hinzufügen von Rauschen auf. Mittels SIFT Deskriptoren werden starke Interessenspunkte abgeglichen, um so Ähnlichkeiten aufzuweisen [65].

#### Funktionsweise SIFT

Folgende Schritte werden benötigt um Merkmale aus Bildern zu extrahieren:

1. **Scale-space extrema detection:** in dieser Stufe, welche die erste Stufe von SIFT ist, werden alle Skalen und Koordinaten durchsucht [66]. Durch eine effiziente Implementierung der difference-of-Gaussian Funktion können potentiell interessante Punkte, welche invariant zu Skalierung und Orientierung sind, identifiziert werden [65].
2. **Keypoint localization:** Einige gefundene Punkte sind nicht gut genug und können mittels Keypoint localization eliminiert werden. Sobald ein Schlüsselpunkt mittels Vergleich eines Pixels mit dessen Nachbarn gefunden wurde, folgt eine detaillierte Anpassung an die nahe gelegenen Daten in Bezug auf Lage, Skalierung und dem Verhältnis der Hauptkrümmungen. [66] Auf Grundlage von Stabilität werden die Schlüsselpunkte ausgewählt [65].
3. **Orientation assignment:** Basierend auf lokalen Bildeigenschaften werden zu jedem Schlüsselpunkt konsistente Orientierungen zugewiesen, wodurch der Deskriptor des Schlüsselpunktes relativ zu dieser Ausrichtung dargestellt werden kann [66].
4. **Keypoint descriptor:** Im ausgewählten Maßstab werden lokale Bildgradienten in der Region um jeden Schlüsselpunkt gemessen. Diese werden in eine Darstellung transformiert, welche ein hohes Maß an Formverzerrung und Beleuchtungsänderung zulässt. [65] Es wird ein gewichtetes Richtungshistogramm im Umkreis des Schlüsselpunktes erstellt und die Orientierung wird dann Anhand des höchsten Schlüsselpunktes ermittelt [66].

#### 3.3.2 Convolutional Neuronal Networks

Convolutional neuronal networks, die im Deutschen übersetzt als "Gefaltete Neuronale Netzwerke" bezeichnet werden, sind eine besondere Form von künstlichen neuronalen Netzwerken, welche mehrere sogenannte Faltungsschichten beinhalten [67]. Seit den 1980er Jahren werden CNNs für visuelle Aufgaben verwendet [68]. In den letzten zehn Jahren gab es durch convolutional neuronale Netzwerke bahnbrechende Ergebnisse in einer Vielzahl von Bereichen der Mustererkennung, von Bildverarbeitung bis zur Spracherkennung. Einer der größten Vorteile von CNNs gegenüber regulären künstlichen neuronalen Netzwerken ist die reduzierte Anzahl der Parameter, wodurch es möglich ist größere Modelle, und somit komplexere Probleme zu lösen [37]. Während bei primitiveren Methoden Filter von Hand entwickelt werden, so sind CNNs durch ausreichendem Training in der Lage Filter selbst zu erlernen [69]. Ein weiterer wichtiger Aspekt von CNNs ist es abstrakte Merkmale zu erkennen. Die Merkmale werden pro Schicht erkannt, so ist es z.B. möglich dass in der ersten Schicht einfache Kanten und Ränder erkannt werden, während in der zweiten Schicht schon einfache Formen identifiziert werden [37]. Während sich die erste Schicht auf einfache Merkmale wie Farben und Kanten konzentriert, so sind tiefere Schichten in der Lage größere Elemente und Formen zu erkennen [70]. Das Netzwerk ist robust und gegenüber Verzerrungen oder optischen Veränderungen unempfindlich. Es kann Bilder verarbeiten, welche in unterschiedlichen Perspektiven und Lichtverhältnissen aufgenommen wurden, und kann trotzdem typische Merkmale eines



Bildes erkennen [67]. Vom Grundprinzip ist ein CNN ein vollvernetztes neuronales Feed-Forward-Netzwerk, was sich aus folgenden Schichten zusammensetzt:

- Eingangs-Schicht,
- Convolutional-Schicht,
- Pooling-Schicht,
- vollständig verbundenen neuronalen Netzwerk und
- Ausgangs-Schicht.

Während auf Convolutional-Schicht weitere Convolutional-Schichten oder Pooling-Schichten folgen können, ist das vollständig verbundene neuronale Netzwerk die letzte Schicht vor der Ausgabe-Schicht [70]. In Abbildung 3.3 ist ein beispielhafter Aufbau eines CNNs dargestellt, bei dem es darum geht Bilder zu klassifizieren.

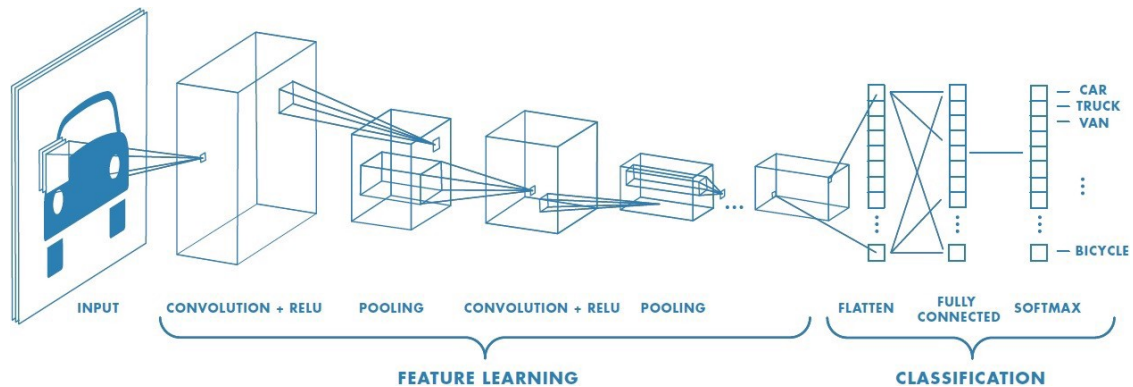


Abbildung 3.3: Aufbau von CNN [69]

#### Convolutional-Schicht

Die Convolutional-Schicht ist der zentrale Baustein eines CNN, in dem der Großteil der Berechnungen stattfindet [70]. Es ist die eigentliche Faltungsebene und ist fähig, aus den Daten Merkmale wie Kanten, Linien oder bestimmte Formen zu erkennen und extrahieren [67]. Hauptanwendungen von CNNs sind Anwendungen mit Bildern, Sprache oder Audio-Signalen [70].

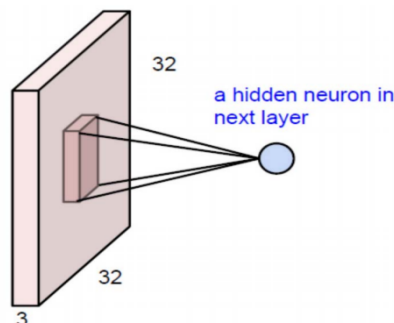


Abbildung 3.4: Beispiel einer Convolution [37]

Um den Vergleich zu einem vollständig vernetzten neuronalen Netzwerk, welches Pixel als Eingabe nimmt, folgt ein Beispiel. Angenommen es handelt sich z.B. um ein 32 Pixel brei-

tes und hohes Bild, so werden hierfür bereits  $32 \times 32 \times 3$  ( $=3072$ ) gewichtete Verbindungen benötigt. 32 für jeweils Breite und Höhe und dann noch jeweils 3 Kanäle für RGB. Wird ein einzelnes Neuron im Hidden-Layer hinzugefügt so werden weitere  $32 \times 32 \times 3$  gewichtete Verbindungen benötigt. Für eine aussagekräftige Bilderkennung werden zwei Neuronen im Hidden-Layer nicht ausreichen, und somit steigt die Anzahl der gewichteten Verbindungen pro Neuron weiter an. Um effizienter als normale vollständig vernetzte neuronale Netzwerke zu sein, betrachten CNNs anstelle des gesamten Bildes nur lokale Regionen davon. Wie in Abbildung 3.4 zusehen ist, bekommt der nächste Layer nur den dazugehörigen Teil von dem davorigen Layer, was in diesem Beispiel eine Verbindung zu einem  $5 \times 5$  Neuron sein könnte. Dadurch wird die Anzahl der benötigten gewichteten Verbindungen reduziert, da sich das CNN nur auf einen Teilbereich der Eingabedaten konzentriert. Um die Anzahl der gewichteten Verbindungen weiter zu verringern können lokale gewichtete Verbindungen für die gesamten Neuronen des nächsten Layers beibehalten werden [37]. Ein weiterer ähnlicher Ansatz ist ein

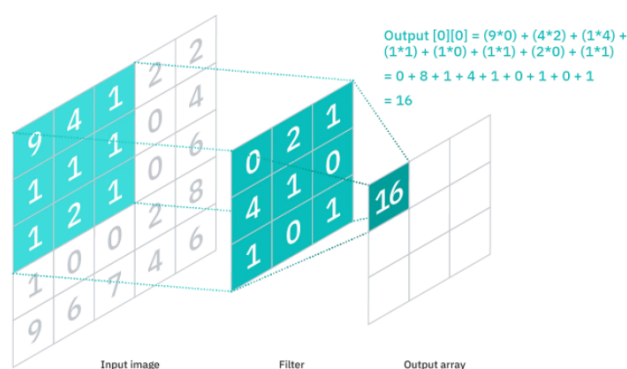


Abbildung 3.5: CNN Filter [70]

Einfügen eines Fensters (z.B.  $5 \times 5 \times 3$ ) [37], welches auch Kernel oder Filter genannt wird [69] und welches über die Eingangsneuronen geschoben wird, und somit die dazugehörige Ausgabe an den dazugehörigen Platz verbindet [37]. Dieser Filter oder auch Feature-Detektor ist ein zweidimensionales Array bestehend aus Gewichten, welche Teile des Bildes repräsentieren. Eine typische Größe dafür ist eine  $3 \times 3$  Matrix, diese kann jedoch variieren. Der Feature-Detektor wird dann auf einen Bereich der Eingabedaten angewandt, in dem das Kreuzprodukt zwischen den Eingabepixeln und dem Filter berechnet werden (siehe Abbildung 3.5). Dieser Bereich wird auch als Receptive Field bezeichnet. Das Ergebnis des Kreuzproduktes wird in ein Ausgabearray gespeichert und anschließend wechselt der Filter in Schritten (englisch: stride) und wiederholt den Vorgang bis der Filter über das ganze Bild angewandt wurde. Das endgültige Ergebnis aus der Reihe der Kreuzprodukte wird als Feature Map, Activation Map oder Convolved Feature bezeichnet [70].

Strides geben an um wieviele Pixel sich der Filter weiter bewegt. Standardmäßig wird ein Stride von 1 verwendet, wie auch in Abbildung 3.6 dargestellt [69]. Ein Stride von mehr als 1 verringert das Ausgabearray, wird in der Praxis jedoch eher selten angewandt [71]. Während sich der Filter über das Bild bewegt, sind seine Gewichte fixiert, diese können sich jedoch während dem Training anpassen [70]. Einer der Nachteile des convolution Schrittes ist der Verlust von Informationen am Rande eines Bildes. Da Informationen nur aufgenommen werden, wenn der Filter sich darüber bewegt haben sie nie die Chance gesehen zu werden. Eine einfache Lösung dafür ist das sogenannte Zero-Padding [37]. Padding ist der Prozess, bei dem der Eingangsmatrix symmetrisch Nullen hinzugefügt werden [69]. Zero-padding wird in der Regel verwendet, wenn die Filter nicht zur Größe des Bildes passen, da mit Padding

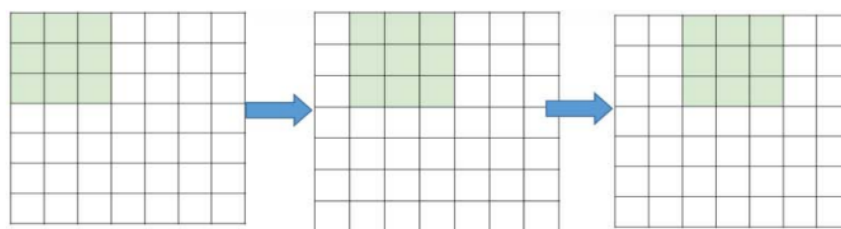


Abbildung 3.6: Stride [37]

die Größe der Ausgangsmatrix verändert werden kann [70]. Es gibt verschiedene Arten von Padding:

- Valid padding: wird auch als no padding bezeichnet. Hierbei findet kein Padding statt und Informationen können verloren gehen, wenn die Größe von Filter und Eingangsdaten nicht übereinstimmt [70].
- Same padding: durch Anwenden von same padding wird sichergestellt, dass die Größe der Ausgangsschicht gleich der Eingangsschicht ist [70].

Die Anzahl der Filter haben Einfluss auf die Tiefe der Ausgabe. Drei Filter würden beispielsweise drei verschiedene Feature Maps generieren und somit eine Tiefe von drei ergeben [70]. Nach jeder Faltungsoperation wendet ein CNN eine Nichtlinearitätsfunktion an [72] um Nichtlinearität in das Modell einzuführen [70]. Dies ist eine nichtlineare Gleichung, die es dem CNN ermöglicht, insgesamt kompliziertere Muster zu lernen. Eine beliebige Nichtlinearität ist Rectified Linear Unit Transformation (ReLU) [72]. Letztendlich wandelt die Faltungsschicht das Bild in numerische Werte um, so dass das neuronale Netz relevante Muster interpretieren und extrahieren kann. Diese Methode bietet den Vorteil, dass aufgrund der Filter das Erkennen von Merkmalen unabhängig von der Position im Bild stattfinden kann [37].

#### Pooling-Schicht

Die Pooling-Schicht wird auch Subsampling [67] oder Downsampling Schicht genannt und der Hauptgedanke von pooling ist down-sampling um Komplexität für nachfolgende Schichten zu reduzieren [37]. Es werden dabei überflüssige Informationen verworfen und die Datenmenge wird verringert, wodurch sich die Berechnungsgeschwindigkeit erhöht [67]. Im Bereich der Bildverarbeitung kann Pooling auch als Verringern der Auflösung angesehen werden [37]. Die Berechnungen wären deutlich aufwendiger, würde man die Pooling-Schicht weglassen, und somit die Convolution Schicht direkt mit dem vollständig verbundenen neuronalen Netzwerk verbinden [69]. Ähnlich wie bei der Convolution-Schicht wird bei der Pooling-Schicht ein Filter über die Eingabedaten geschoben, jedoch hat dieser Filter keine Gewichte. Stattdessen wird eine Aggregationsfunktion auf die Werte innerhalb des rezeptiven Feldes angewandt und somit die Ausgangsmatrix ausgefüllt [70]. Es gibt verschiedene Arten von Pooling:

- Max pooling: ist eines der weitverbreitetsten Pooling Methoden [37]. Während sich der Filter über den Eingabedatensatz bewegt, wird der Pixel mit dem größten Wert genommen und an das Outputarray gegeben [70]. Max Pooling wirkt auch als Rauschunterdrückung [69].
- Average pooling: Während sich der Filter über den Eingabedatensatz bewegt, wird ein Mittelwert innerhalb des rezeptiven Feldes ermittelt und an das Outputarray übergeben [70].

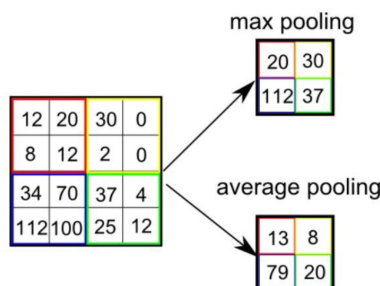


Abbildung 3.7: Max und Average Pooling [69]

### Vollständig verbundenes neuronales Netzwerk

Das vollständig verbundene neuronale Netzwerk folgt auf die Abfolgen von Convolutional- und Pooling Schicht und bildet den Abschluss des convolutional neuronalen Netzwerk [67]. Der Aufbau des vollständig verbundenen neuronalen Netzwerk ähnelt der Art und Weise, wie die Neuronen in einem herkömmlichen neuronalen Netz angeordnet sind. Jedes Neuron ist vollständig mit den Neuronen aus der vorherigen, und auch der nächsten Schicht verbunden [37]. Abhängig von den Klassen und Objekten, welche das neuronale Netzwerk bestimmen soll, werden die Anzahl der Neuronen bestimmt [67]. Der größte Nachteil der voll vernetzten Schicht ist es, dass es viele Parameter enthält, welche komplexe Berechnungen benötigen. Um die Anzahl der Neuronen und Verbindungen zu verringern kann die Dropout-Technik verwendet werden [37]. Über mehrere Epochen hinweg ist das Model in der Lage, zwischen dominierenden und bestimmten niederschweligen Merkmalen in Bildern zu unterscheiden und diese mit Hilfe von Softmax zu klassifizieren [69].

### 3.3.3 Siamese Networks

Im Bereich von Deep learning sind neuronale Netzwerke für fast jede Aufgabe geeignet, um jedoch gute Ergebnisse liefern zu können, sind diese auf ausreichend Daten angewiesen. Traditionell werden sie verwendet um verschiedene Klassen vorherzusagen, wenn jedoch eine neue Klasse hinzukommt, so muss das neuronale Netzwerk aktualisiert und erneut trainiert werden. Zusätzlich sind meist nur wenige Daten für bestimmte Probleme wie z.B. Gesichtserkennung oder Unterschriftenprüfung verfügbar. Um diese Art von Problemen zu lösen wurden Siamesische Netzwerke entwickelt, welche eine neue Architekturart von neuronalen Netzwerken sind [?]. Erstmals wurden diese Netzwerke 1990 von Bromley und LeCun zum Zweck der Unterschriftenverifikation entwickelt [?]. Da Siamesische Netzwerke nur wenige Daten benötigen, wurden diese über die letzten Jahre immer beliebter. Meist werden sie verwendet um Ähnlichkeiten in den Eingabedaten zu vergleichen [?].

#### Architektur

Ein siamesischen Netzwerk (auch twin network genannt) hat seinen Namen von siamesischen Zwillingen, welche Zwillinge sind, die physikalisch miteinander verbunden sind [?]. Es besteht aus zwei oder mehreren sogenannten Teil- oder Zwillingnetzwerken, welche unterschiedliche Eingaben akzeptieren und identisch sind [?, ?]. Identisch bedeutet in diesem Fall dass die Teilnetzwerke die gleiche Konfiguration mit denselben Parametern und Gewichten besitzen. Wenn Parameter aktualisiert werden, so werden diese gespiegelt in allen Teilnetzwerken ebenfalls aktualisiert [?]. Eine Beispielarchitektur für ein siamesischen Netzwerk ist Abbildung 3.8

### 3 Bildbezogenes Machine Learning

zu sehen. In diesem Fall werden zwei Bilder von handgeschriebenen Zahlen in zwei identische

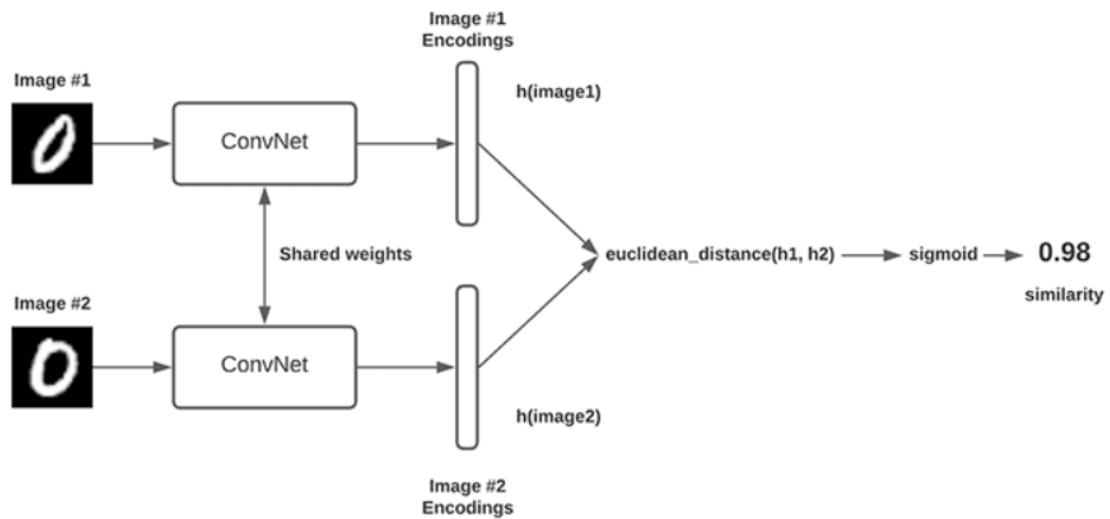


Abbildung 3.8: Beispielarchitektur Siamesisches Netzwerk [?]

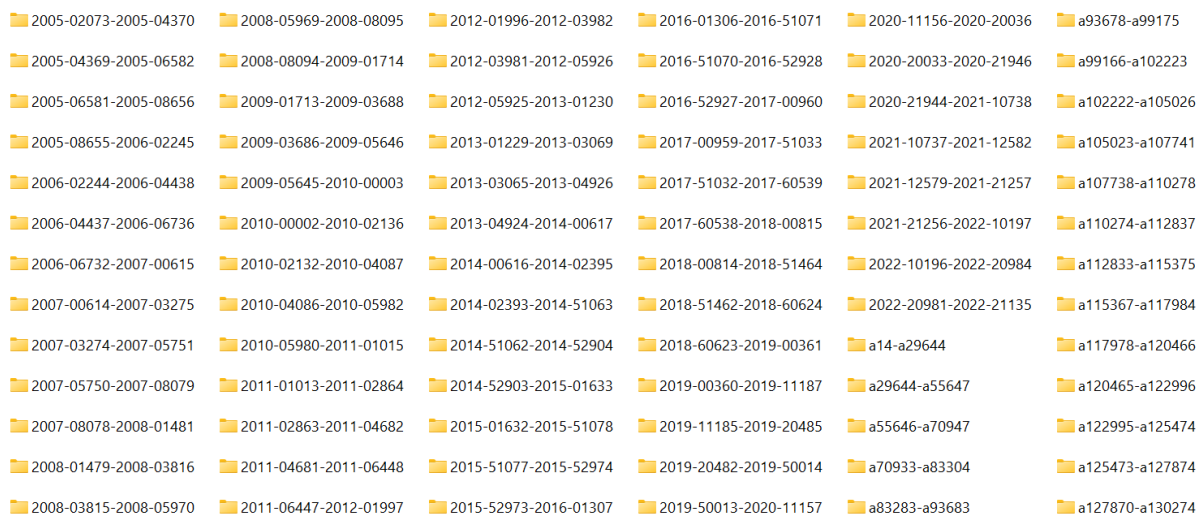
CNNs gegeben und anschließend wird die Euklidische Distanz berechnet. Am Ende wird in die Sigmoid Aktivierungsfunktion angewandt, um anhand eines Ergebnisses zwischen 0 und 1 die Ähnlichkeit auszudrücken.

## 4 Prototyp zur Bildähnlichkeitserkennung von Markenlogos

In diesem Kapitel wird der Versuch beschrieben, einen Prototypen zu entwickeln, welcher zwei Eingabedaten entgegen nimmt und deren Ähnlichkeit ermitteln kann. Es wird einerseits ein Versuch mittels **SIFT** und andererseits ein Versuch mit einem **Siamesischem Netzwerk** beschrieben.

### 4.1 Daten

Wie in der Einleitung bereits erwähnt beschäftigt sich das österreichische Patentamt viel mit Markenlogos und deren Ähnlichkeit. Da diese Masterarbeit in Zusammenarbeit mit dem österreichischen Patentamt entwickelt wird, haben sie mir dankenswerterweise ihre Logodaten zur Verfügung gestellt. Diese Daten enthalten 210630 Bilder, welche in 213 Ordner unterteilt sind und was insgesamt 13.4 GB an Daten ausmacht. Die Ordnerstruktur ist nach Anmeldungs-jahr der Markenlogos gegliedert (siehe Abbildung 4.1). Die Bilder in den jeweiligen Ordnern



2005-02073-2005-04370	2008-05969-2008-08095	2012-01996-2012-03982	2016-01306-2016-51071	2020-11156-2020-20036	a93678-a99175
2005-04369-2005-06582	2008-08094-2009-01714	2012-03981-2012-05926	2016-51070-2016-52928	2020-20033-2020-21946	a99166-a102223
2005-06581-2005-08656	2009-01713-2009-03688	2012-05925-2013-01230	2016-52927-2017-00960	2020-21944-2021-10738	a102222-a105026
2005-08655-2006-02245	2009-03686-2009-05646	2013-01229-2013-03069	2017-00959-2017-51033	2021-10737-2021-12582	a105023-a107741
2006-02244-2006-04438	2009-05645-2010-00003	2013-03065-2013-04926	2017-51032-2017-60539	2021-12579-2021-21257	a107738-a110278
2006-04437-2006-06736	2010-00002-2010-02136	2013-04924-2014-00617	2017-60538-2018-00815	2021-21256-2022-10197	a110274-a112837
2006-06732-2007-00615	2010-02132-2010-04087	2014-00616-2014-02395	2018-00814-2018-51464	2022-10196-2022-20984	a112833-a115375
2007-00614-2007-03275	2010-04086-2010-05982	2014-02393-2014-51063	2018-51462-2018-60624	2022-20981-2022-21135	a115367-a117984
2007-03274-2007-05751	2010-05980-2011-01015	2014-51062-2014-52904	2018-60623-2019-00361	a14-a29644	a117978-a120466
2007-05750-2007-08079	2011-01013-2011-02864	2014-52903-2015-01633	2019-00360-2019-11187	a29644-a55647	a120465-a122996
2007-08078-2008-01481	2011-02863-2011-04682	2015-01632-2015-51078	2019-11185-2019-20485	a55646-a70947	a122995-a125474
2008-01479-2008-03816	2011-04681-2011-06448	2015-51077-2015-52974	2019-20482-2019-50014	a70933-a83304	a125473-a127874
2008-03815-2008-05970	2011-06447-2012-01997	2015-52973-2016-01307	2019-50013-2020-11157	a83283-a93683	a127870-a130274

Abbildung 4.1: Ordnerstruktur der Markenlogos

sind mittels Anmeldungs-jahr und einer fortlaufenden Nummer beschrieben (siehe Abbildung 4.2).

#### 4 Prototyp zur Bildähnlichkeitserkennung von Markenlogos



Abbildung 4.2: Datenstruktur in den Ordnern

Um die Daten etwas besser verstehen zu können habe ich mir zwei Hilfsfunktionen geschrieben. Mittels folgender Funktion (Listing 4.1) wird das kleinste und größte Bild ermittelt.

```
def get_smallest_and_largest_image(filenamees):
    all_images = []
    for filename in filenamees:
        img = PILImage.open(filename, 'r')
        all_images.append([img.filename, img.size])
    sort = sorted(all_images, key=lambda x:x[1])
    return [sort[0],sort[-1]]
```

Listing 4.1: Hilfsfunktion zur Ermittlung des kleinsten und größten Bildes

Die Funktion nimmt eine Liste von Dateinamen entgegen und ermittelt zuerst von jedem Bild die Größe und anschließend wird die Liste dann nach der Größe sortiert. Somit befindet sich das kleinste Bild an erster und das größte an letzter Stelle, welche am Ende der Funktion zurückgegeben werden. Das Ergebnis ist in Abbildung 4.3 zu sehen. Das kleinste Bild hat 42

```
total count:
210630
smallest:
C:\Study\FH_Campus\MasterThesis\source\Images\1990-00821-1990-03474\1990-02289.gif
(42, 13)
largest:
C:\Study\FH_Campus\MasterThesis\source\Images\2017-60538-2018-00815\2017-61246.gif
(23080, 4724)
```

Abbildung 4.3: Ergebnis Ermittlung kleinstes und größtes Bild

mal 13 Pixel und wie in Abbildung 4.4 zu sehen ist, kann man hier die Pixel einzeln abzählen. Das größte Bild ist 23080 mal 4724 Pixel groß, was für Latex leider zu groß war und ich es runterkomprimieren musste. Das komprimierte Bild ist in Abbildung 4.5 veranschaulicht. Der Datensatz weist eine hohe Diversität bei den Merkmalen Größe, Auflösung, Ausrichtung und Farben auf wie auch in Abbildung 4.2 zu sehen ist. Es sind kleine und große, horizontale



Abbildung 4.4: Kleinstes Bild im Datensatz



Abbildung 4.5: Größtes Bild im Datensatz

und vertikale, farbig und schwarz weiß Bilder vorhanden. Ebenfalls gibt es Markenlogos die ausschließlich aus Text bestehen, ein Logo in Bildform und Text beinhalten oder nur ein Logo in Bildform enthalten.

## 4.2 Bildvorverarbeitung

### 4.2.1 Sift

Da SIFT gegenüber Skalierung und Rotation invariant ist wird hierbei keine Bildvorverarbeitung benötigt (siehe Kapitel 3.3.1).

### 4.2.2 Siamese Network

Aufgrund der Diversität zwischen den Bildern im Datensatz benötigt es für das Siamesische Netzwerk eine Bildvorverarbeitung. Hierfür wurde eine Hilfsfunktion (siehe Listing 4.2) erstellt, welche den Mittelwert der Breite und Höhe über den kompletten Datensatz berechnet.

---

```
def get_average_imagesize(filenamees):  
    widths = []  
    heights = []  
    for filename in filenamees:  
        size = PILImage.open(filename, 'r').size  
        widths.append(size[0])  
        heights.append(size[1])  
    average_width = sum(widths)/len(widths)  
    average_height = sum(heights)/len(heights)  
    return [average_width, average_height]
```

---

Listing 4.2: Hilfsfunktion zur Ermittlung der durchschnittlichen Größe im Datensatz

Das Ergebnis beträgt gerundet 831 Pixel breit und 505 Pixel hoch ist in Abbildung 4.6 zu sehen.



```
average image width: 830.668931301334  
average image height: 504.9858139866116
```

Abbildung 4.6: Durchschnittliche Größe eines Bildes im Datensatz

### 4.3 Algorithmus

## 5 Diskussion der Ergebnisse

Vergleich SIFT vs. Siamese Network skizze: SIFT: benötigt keine Trainingsdaten + keine Bildvorverarbeitung, ist schnell, schlechte generalisierung, simple implementierung Siamese Network (mit CNN): benötigt Trainingsdaten um zu lernen, benötigt Bildvorverarbeitung, viele Parameter zu setzen

Algorithmus	Vorteile	Nachteile
SIFT	tbd. VORTEILE	tbd. NACHTEILE
Siamese Network	tbd. VORTEILE	tbd. NACHTEILE

Tabelle 5.1: Vergleich SIFT vs. Siamese Network

## 6 Conclusio

## **7 Ausblick**

# Literaturverzeichnis

- [1] Österreichisches Patentamt, “Statistische Übersicht über geschäftsumfang und geschäftstätigkeit in patentangelegenheiten gebrauchsmusterangelegenheiten markenangelegenheiten musterangelegenheiten,” Website, 2020, online erhältlich unter [https://www.patentamt.at/fileadmin/root\\_oepa/Dateien/Allgemein/Statistiken/Stat2020\\_v1\\_1.pdf](https://www.patentamt.at/fileadmin/root_oepa/Dateien/Allgemein/Statistiken/Stat2020_v1_1.pdf); abgerufen am 27. Februar 2022. 1
- [2] —, “MarkenÄhnlichkeitsrecherche,” Website, 2022, online erhältlich unter <https://www.patentamt.at/markenaehnlichkeitsrecherche/>; abgerufen am 27. Februar 2022. 1
- [3] Research and Markets, “Machine learning market by vertical bfsi, healthcare and life sciences, retail, telecommunication, government and defense, manufacturing, energy and utilities, deployment mode, service, organization size, and region - global forecast to 2022,” Website, 2017, online erhältlich unter <https://www.researchandmarkets.com/reports/4395173/machine-learning-market-by-vertical-bfsi/>; abgerufen am 18. März 2022. 1
- [4] V. Kurama, “ML-based image processing,” Website, 2021, online erhältlich unter <https://nanonets.com/blog/machine-learning-image-processing/>; abgerufen am 13. Mai 2022. 2
- [5] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*, 2nd ed. O’Reilly, 2019. 2, 3, 4, 6, 7, 8, 35
- [6] L. Wuttke, “Machine learning: Definition, algorithmen, methoden und beispiele,” Website, 2022, online erhältlich unter <https://datasolut.com/was-ist-machine-learning/>; abgerufen am 18. März 2022. 2, 35
- [7] Wikipedia, “Maschinelles lernen,” Website, 2022, online erhältlich unter [https://de.wikipedia.org/wiki/Maschinelles\\_Lernen](https://de.wikipedia.org/wiki/Maschinelles_Lernen); abgerufen am 04. April 2022. 2
- [8] E. Burns, “machine learning,” Website, 2021, online erhältlich unter <https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML>; abgerufen am 04. April 2022. 2
- [9] Oracle, “Was ist machine learning?” Website, 2022, online erhältlich unter <https://www.oracle.com/at/data-science/machine-learning/what-is-machine-learning/>; abgerufen am 26. März 2022. 2, 3
- [10] Netflix, “Machine learning learning how to entertain the world,” Website, 2022, online erhältlich unter <https://research.netflix.com/research-area/machine-learning>; abgerufen am 26. März 2022. 2
- [11] H. Hodson, “What you ‘like’ on facebook gives away your personality,” Website, 2015, online erhältlich unter <https://www.newscientist.com/article/dn26781-what-you-like-on-facebook-gives-away-your-personality/>; abgerufen am 26. März 2022. 2
- [12] Intellipaat, “Supervised learning vs unsupervised learning vs reinforcement learning,” Website, 2022, online erhältlich unter <https://intellipaat.com/blog/>

- supervised-learning-vs-unsupervised-learning-vs-reinforcement-learning/; abgerufen am 09. April 2022. 3, 4
- [13] L. G. Serrano, *Grokking Machine Learning*, 4th ed. Manning, 2021. [Online]. Available: <https://livebook.manning.com/book/grokking-machine-learning/chapter-1/> 3, 4, 35
- [14] IBM, “Unsupervised learning,” Website, 2022, online erhältlich unter <https://www.ibm.com/cloud/learn/unsupervised-learning>; abgerufen am 02. Mai 2022. 4, 5
- [15] L. Wuttke, “Was ist unsupervised learning (unüberwachtes lernen)?” Website, 2020, online erhältlich unter <https://datasolut.com/wiki/unsupervised-learning/>; abgerufen am 02. Mai 2022. 4, 5, 35
- [16] S. Priy, “Clustering in machine learning,” Website, 2021, online erhältlich unter <https://www.geeksforgeeks.org/clustering-in-machine-learning/>; abgerufen am 02. Mai 2022. 4, 5
- [17] Baysan, “What is association rule learning? an applied example in python: Basket analysis and product offering,” Website, 2021, online erhältlich unter <https://medium.com/codex/what-is-association-rule-learning-abd4a76144d8>; abgerufen am 02. Mai 2022. 5
- [18] N. Barla, “Dimensionality reduction for machine learning,” Website, 2021, online erhältlich unter <https://neptune.ai/blog/dimensionality-reduction>; abgerufen am 02. Mai 2022. 5
- [19] DataRobot, “Semi-supervised learning,” Website, 2021, online erhältlich unter <https://www.datarobot.com/blog/semi-supervised-learning/>; abgerufen am 03. Mai 2022. 5
- [20] Alexsoft, “Semi-supervised learning, explained with examples,” Website, 2022, online erhältlich unter <https://www.altexsoft.com/blog/semi-supervised-learning/>; abgerufen am 03. Mai 2022. 5, 6
- [21] S. Dobilas, “Self-training classifier: How to make any algorithm behave like a semi-supervised one,” Website, 2021, online erhältlich unter <https://towardsdatascience.com/self-training-classifier-how-to-make-any-algorithm-behave-like-a-semi-supervised-one-2958e7b54ab7>; abgerufen am 03. Mai 2022. 5, 6
- [22] Wikipedia, “Co-training,” Website, 2021, online erhältlich unter <https://en.wikipedia.org/wiki/Co-training>; abgerufen am 04. Mai 2022. 6
- [23] C. Lemke, “So funktioniert reinforcement learning,” Website, 2019, online erhältlich unter <https://www.alexanderthamm.com/de/blog/einfach-erklart-so-funktioniert-reinforcement-learning/>; abgerufen am 04. Mai 2022. 6
- [24] L. Wuttke, “So funktioniert reinforcement learning,” Website, 2019, online erhältlich unter <https://datasolut.com/reinforcement-learning/>; abgerufen am 04. Mai 2022. 6, 7, 35
- [25] —, “Künstliche neuronale netzwerke: Definition, einföhrung, arten und funktion,” Website, 2022, online erhältlich unter <https://datasolut.com/neuronale-netzwerke-einfuehrung/>; abgerufen am 10. Mai 2022. 7, 8, 9, 35
- [26] Wikipedia, “Künstliches neuronales netz,” Website, 2022, online erhältlich unter [https://de.wikipedia.org/wiki/Künstliches\\_neuronales\\_Netz](https://de.wikipedia.org/wiki/Künstliches_neuronales_Netz); abgerufen am 05. Mai 2022. 7
- [27] IBM, “What are neural networks?” Website, 2020, online erhältlich unter <https://www.ibm.com/cloud/learn/neural-networks>; abgerufen am 05. Mai 2022. 7
- [28] D. G. Inonos, “Neural networks: Was können künstliche neuronale netze?” Website, 2020, online erhältlich unter <https://www.ionos.at/digitalguide/online-marketing/suchmaschinenmarketing/was-ist-ein-neural-network/>; abgerufen am 05. Mai 2022. 8, 9

- [29] I. D. Baruah, “Understanding the basics of neural networks (for beginners),” Website, 2021, online erhältlich unter <https://medium.com/geekculture/understanding-the-basics-of-neural-networks-for-beginners-9c26630d08>; abgerufen am 07. Mai 2022. 8
- [30] H. Singh, *Practical Machine Learning and Image Processing*, 2019. 8, 15, 35
- [31] C. Nicholson, “A beginner’s guide to neural networks and deep learning,” Website, 2022, online erhältlich unter <https://wiki.pathmind.com/neural-network>; abgerufen am 05. Mai 2022. 8, 9
- [32] E. Guresen and G. Kayakutlu, “Definition of artificial neural networks with comparison to other networks,” *Procedia Computer Science*, vol. 3, pp. 426–433, 2011, world Conference on Information Technology. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050910004461> 9
- [33] S. Sharma, “What the hell is perceptron?” Website, 2017, online erhältlich unter <https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53>; abgerufen am 11. Mai 2022. 9
- [34] Simplilearn, “What is perceptron: A beginners guide for perceptron,” Website, 2022, online erhältlich unter <https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron>; abgerufen am 11. Mai 2022. 9
- [35] A. Thamm, “Feedforward neural network,” Website, 2022, online erhältlich unter <https://www.alexanderthamm.com/de/data-science-glossar/feedforward-neural-network/>; abgerufen am 11. Mai 2022. 9
- [36] DeepAi, “Feed forward neural network,” Website, 2022, online erhältlich unter <https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network>; abgerufen am 11. Mai 2022. 9
- [37] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. 9, 17, 18, 19, 20, 21, 35
- [38] M. Saeed, “An introduction to recurrent neural networks and the math that powers them,” Website, 2021, online erhältlich unter <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/>; abgerufen am 11. Mai 2022. 9
- [39] IBM, “Recurrent neural networks,” Website, 2022, online erhältlich unter <https://www.ibm.com/cloud/learn/recurrent-neural-networks>; abgerufen am 11. Mai 2022. 10
- [40] S. K. Sarvepalli, “Deep learning in neural networks: The science behind an artificial brain,” 10 2015. 10
- [41] S. F. Frauke Günther, “neuralnet: Training of neural networks,” 6 2010. 10
- [42] V. Bushaev, “How do we ‘train’ neural networks ?” Website, 2017, online erhältlich unter <https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73>; abgerufen am 07. Mai 2022. 10
- [43] D. Johnson, “Back propagation neural network: What is backpropagation algorithm in machine learning?” Website, 2022, online erhältlich unter <https://www.guru99.com/backpropagation-neural-network.html>; abgerufen am 10. Mai 2022. 10
- [44] J.-L. Queguiner, “What does training neural networks mean?” Website, 2020, online erhältlich unter <https://blog.ovhcloud.com/what-does-training-neural-networks-mean/>; abgerufen am 10. Mai 2022. 10
- [45] W. Rowe, “What is a neural network? an introduction with examples,” Website, 2020,

- online erhältlich unter <https://www.bmc.com/blogs/neural-network-introduction/>; abgerufen am 11. Mai 2022. 11, 35
- [46] Peltarion, “Categorical crossentropy,” Website, 2022, online erhältlich unter <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy>; abgerufen am 12. Mai 2022. 12
- [47] S. Fernandes, D. Duseja, and R. Muthalagu, “Application of image processing techniques for autonomous cars,” *Proceedings of Engineering and Technology Innovation*, vol. 17, pp. 01–12, Jan. 2021. [Online]. Available: <https://ojs.imeti.org/index.php/PETI/article/view/6074> 13
- [48] Reuters, “Apple hires former tesla engineer to boost self-driving car effort,” Website, 2021, online erhältlich unter <https://www.reuters.com/technology/apple-hires-former-tesla-engineer-boost-car-effort-bloomberg-news-2021-11-05/>; abgerufen am 13. Mai 2022. 13
- [49] Q. Ke, J. Zhang, W. Wei, D. Połap, M. Woźniak, L. Kośmider, and R. Damaševičius, “A neuro-heuristic approach for recognition of lung diseases from x-ray images,” *Expert Systems with Applications*, vol. 126, pp. 218–232, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0957417419300776> 13
- [50] M. A. Elaziz, K. M. HosnyID, A. Salah, M. M. Darwish, S. Lu, and A. T. Sahlol, “New machine learning method for image-based diagnosis of covid-19,” May. 2020. 13
- [51] R. Demush, “A brief history of computer vision (and convolutional neural networks),” Website, 2019, online erhältlich unter <https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3>; abgerufen am 13. Mai 2022. 13
- [52] Temera, “Computer vision (ai) recognising objects in images,” Website, 2022, online erhältlich unter <https://temera.it/en/news/blog/computer-vision.html>; abgerufen am 16. Mai 2022. 13, 14
- [53] IBM, “What is computer vision?” Website, 2022, online erhältlich unter <https://www.ibm.com/topics/computer-vision>; abgerufen am 13. Mai 2022. 13
- [54] H. David and W. Torsten, “Receptive fields of single neurones in the cat’s striate cortex,” vol. 148, May. 1959. 13
- [55] Wikipedia, “Digitales bild,” Website, 2022, online erhältlich unter [https://de.wikipedia.org/wiki/Digitales\\_Bild](https://de.wikipedia.org/wiki/Digitales_Bild); abgerufen am 14. Mai 2022. 14
- [56] T. Target, “Definition image,” Website, 2016, online erhältlich unter <https://www.techtarget.com/whatis/definition/image>; abgerufen am 14. Mai 2022. 14
- [57] Wikipedia, “Digital image,” Website, 2022, online erhältlich unter [https://en.wikipedia.org/wiki/Digital\\_image](https://en.wikipedia.org/wiki/Digital_image); abgerufen am 14. Mai 2022. 14
- [58] A. Cepalia, “How computers see images,” Website, 2022, online erhältlich unter <https://realpython.com/lessons/how-computers-see-images/>; abgerufen am 14. Mai 2022. 14
- [59] S. Kadam, “Cnn series part 1: How do computers see images?” Website, 2020, online erhältlich unter <https://medium.com/analytics-vidhya/cnn-series-part-1-how-do-computers-see-images-32462a0b33ca>; abgerufen am 16. Mai 2022. 14, 35
- [60] U. of Michigan Library, “Native file formats,” Website, 2021, online erhältlich unter <https://guides.lib.umich.edu/c.php?g=282942&p=1885348>; abgerufen am 16. Mai 2022. 15



- [61] G. Boesch, “A complete guide to image classification in 2022,” Website, 2022, online erhältlich unter <https://viso.ai/computer-vision/image-classification/>; abgerufen am 17. Mai 2022. 15
- [62] H. Bandyopadhyay, “Image classification explained [+v7 tutorial],” Website, 5 2022, online erhältlich unter <https://www.v7labs.com/blog/image-classification-guide>; abgerufen am 17. Mai 2022. 15
- [63] P. Gavali and J. S. Banu, “Chapter 6 - deep convolutional neural network for image classification on cuda platform,” in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, A. K. Sangaiah, Ed. Academic Press, 2019, pp. 99–122. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128167182000130> 15
- [64] S. University, “Cs231n: Deep learning for computer vision,” 2022. [Online]. Available: <https://cs231n.github.io/classification/> 16, 35
- [65] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” 2004. 16, 17
- [66] H. R. Kher and V. K. Thakar, “Scale invariant feature transform based image matching and registration,” in *2014 Fifth International Conference on Signal and Image Processing*, 2014, pp. 50–55. 16, 17
- [67] S. Luber and N. Litzel, “Was ist ein convolutional neural network?” Website, 2019, online erhältlich unter <https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/>; abgerufen am 23. Mai 2022. 17, 18, 20, 21
- [68] W. Rawat and Z. Wang, “Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review,” *Neural Computation*, vol. 29, no. 9, pp. 2352–2449, 09 2017. [Online]. Available: [https://doi.org/10.1162/neco\\_a\\_00990](https://doi.org/10.1162/neco_a_00990) 17
- [69] S. Saha, “A comprehensive guide to convolutional neural networks — the eli5 way,” Website, 2018, online erhältlich unter <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>; abgerufen am 23. Mai 2022. 17, 18, 19, 20, 21, 35
- [70] IBM, “Convolutional neural networks,” Website, 2020, online erhältlich unter <https://www.ibm.com/cloud/learn/convolutional-neural-networks>; abgerufen am 25. Mai 2022. 17, 18, 19, 20, 35
- [71] S. University, “Cs231n: Convolutional neural networks for visual recognition,” 2022. [Online]. Available: <https://cs231n.github.io/convolutional-networks/> 19
- [72] R. Draelos, “How computers see: Intro to convolutional neural networks,” Website, 2019, online erhältlich unter <https://glassboxmedicine.com/2019/05/05/how-computers-see-intro-to-convolutional-neural-networks/>; abgerufen am 26. Mai 2022. 20

# Abbildungsverzeichnis

2.1	Machine Learning nimmt Eingabedaten mit Beispielen und lernt daraus, um für die Zukunft Prognosen zu machen [6] . . . . .	2
2.2	Labeled vs unlabeled Data [13] . . . . .	3
2.3	"Model trainiert ohne Zielvariable und findet eigenständig Muster und Zusammenhänge in den Daten"[15] . . . . .	4
2.4	Beispiel von Reinforcement Learning [24] . . . . .	7
2.5	Aufbau eines Neurons [25] . . . . .	7
2.6	Multiple Layer in einem biologischen neuronalen Netzwerk [5] . . . . .	8
2.7	Einfache Veranschaulichung eines neuronalen Netzwerkes [30] . . . . .	8
2.8	Beispielbilder aus dem MNIST Datensatz . . . . .	11
2.9	Architektur des Neuronalen Netzwerkes [45] . . . . .	11
2.10	Vergleich 5 und 50 Epochen . . . . .	12
3.1	Repräsentation eines Bildes in Zahlen [59] . . . . .	14
3.2	Beispiele für die Herausforderungen von maschineller Bilderkennung [64] . . .	16
3.3	Aufbau von CNN [69] . . . . .	18
3.4	Beispiel einer Convolution [37] . . . . .	18
3.5	CNN Filter [70] . . . . .	19
3.6	Stride [37] . . . . .	20
3.7	Max und Average Pooling [69] . . . . .	21
3.8	Beispielarchitektur Siamesisches Netzwerk [?] . . . . .	22
4.1	Ordnerstruktur der Markenlogos . . . . .	23
4.2	Datenstruktur in den Ordnern . . . . .	24
4.3	Ergebnis Ermittlung kleinstes und größtes Bild . . . . .	24
4.4	Kleinstes Bild im Datensatz . . . . .	25
4.5	Größtes Bild im Datensatz . . . . .	25
4.6	Durchschnittliche Größe eines Bildes im Datensatz . . . . .	26

# Tabellenverzeichnis

3.1	verschiedene Dateiformate . . . . .	15
5.1	Vergleich SIFT vs. Simaese Network . . . . .	27

# Appendix

---

```
import keras
from tensorflow.keras.datasets import mnist
from matplotlib import pyplot as plt
from keras.layers import Dense
from keras.models import Sequential

# load dataset
(pic_train, label_train), (pic_test, label_test) = mnist.load_data
()
# summarize loaded dataset
print('Train: X=%s, y=%s' % (pic_train.shape, label_train.shape))
print('Test: X=%s, y=%s' % (pic_test.shape, label_test.shape))
# plot first few images
fig, axes = plt.subplots(ncols=5, sharex=False,
    sharey=True, figsize=(10, 4))
for i in range(5):
    axes[i].set_title(label_train[i])
    axes[i].imshow(pic_train[i], cmap='gray')
    axes[i].get_xaxis().set_visible(False)
    axes[i].get_yaxis().set_visible(False)
plt.show()

num_classes = 10 # Zahlen von 0 bis 9
image_size = 28*28 # Pixels

# Gives a new shape to an array without changing its data.
x_train = pic_train.reshape(pic_train.shape[0], image_size)
x_test = pic_test.reshape(pic_test.shape[0], image_size)

# Converts a class vector (integers) to binary class matrix.
y_train = keras.utils.np_utils.to_categorical(label_train,
    num_classes)
y_test = keras.utils.np_utils.to_categorical(label_test,
    num_classes)

model = Sequential()

model.add(Dense(units=15, activation='sigmoid', input_shape=(
    image_size,)))
model.add(Dense(units=num_classes, activation='softmax'))
model.summary()
```

```
model.compile(loss='categorical_crossentropy', optimizer='sgd',
              metrics=['acc'])
history = model.fit(x_train, y_train, batch_size=128, epochs=5,
                   verbose=False, validation_split=.1)
loss, accuracy = model.evaluate(x_test, y_test, verbose=False)

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['training', 'validation'], loc='best')
plt.show()

print(f'Test loss: {loss:.3}')
print(f'Test accuracy: {accuracy:.3}')
```

---

Listing 1: Vollständiger Code für ein neuronales Netzwerk das Handschriftliche Zahlen erkennt