

B A C H E L O R A R B E I T

Der Einfluss von Farbnormalisierung auf die  
Klassifizierung von Bildern durch künstliche  
neuronale Netze

Vorgelegt an der TH Köln  
Campus Gummersbach  
im Studiengang  
Medieninformatik

ausgearbeitet von:  
TORBEN KRAUSE  
(Matrikelnummer: 11106885)

**Erster Prüfer:** Professor Dr. Martin Eisemann  
**Zweiter Prüfer:** Professor Dr. Matthias Böhmer

Gummersbach, im Juli

**Vorwort**

**Danksagung**

**Abstract**

# Abbildungsverzeichnis

1	Szene mit unterschiedlicher Ausleuchtung . . . . .	6
2	Aufbau eines Digitalen Bildes mit zwei Helligkeitsabstufungen . . . . .	8
3	Histogramm eines Graustufen Bildes mit 16 Helligkeitsabstufungen [BB09, S. 42] . . . . .	11
4	Unterschiedlicher Kontrast und die Auswirkungen im Histogramm: niedriger Kontrast (links), normaler Kontast (Mitte), hoher Kontrast (rechts)[BB09, S. 45] . . . . .	12
5	Unterschiedlicher Dynamik und die Auswirkungen im Histogramm: hohe Dynamik (links), niedrige Dynamik mit 64 Graustufen (Mitte), extrem niedriger Kontrast mir 6 Graustufen (rechts)[BB09, S. 45] . . . . .	13
6	Aufbau eines Neurons [CSD18] . . . . .	15
7	Schematische Modellierung eines Perzeptrons nach Rosenblatt [Wik19] . .	16
8	Aufbau eines künstlichen neuronalen Netzes [Ang18] . . . . .	18
9	Aufbau eines Convolutional neural Network [Pen+16] . . . . .	19
10	Funktionsweise der Faltungsschicht [Mis19] . . . . .	21
11	Funktionsweise des Pooling Layers mit der Max Pooling variante [Wik18] .	22
12	Vorbereitung der Datensätzen mit dem LabelImg Programm [tzu19] . . .	23
13	Auswirkung des Gray-World Algorithmus . . . . .	32
14	Auswirkung der Histogramm Ausgleichung . . . . .	33
15	Das Zielbild wurde mithilfe des Referenzbildes ausgeglichen und angepasst. Auf der linken Seite ist das Ergebnis der Spezifikation . . . . .	34
16	Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Ohne Normalisierung . . . . .	38
17	Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch den Histogramm Ausgleich . . . . .	45
18	Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch den Gray-World Algorithmus . . . . .	46
19	Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch die Histogramm Spezifikation . . . . .	47
20	Helligkeitsverteilung und den Einfluss der Normalisierungs Algorithmen .	48

# Tabellenverzeichnis

1	Pixelbild in ein tabellarisches Histogramm überführt . . . . .	25
2	Mathematische Umsetzung der Histogrammausgleichung . . . . .	25
3	Tabellarisches Histogramm des Quell-Bildes (B1) . . . . .	26
4	Tabellarisches Histogramm des Referenz-Bildes (B2) . . . . .	26
5	Von zwei Ausgeglichenen Histogrammen zu einem Spezifizierten Histogramm	27
6	Tabellarisches Histogramm des Quell-Bildes (B1) auf Basis des Referenz-Bildes (B2) . . . . .	27
7	Modelle [LLC18] welche mit dem COCO Dataset Trainiert wurden [Con18]	28
8	Datensatz mit Nahrungsmitteln . . . . .	29
9	Pascal Visual Object Classes [Eve+]	29
10	Stanford Dog Dataset [Kho+11]	30
11	Durchschnittliche Genauigkeiten des Modells mit dem Nahrungsmittel Datensatz . . . . .	35
12	Durchschnittliche Genauigkeiten des Modells mit dem PascalVOC Datensatz	36
13	Durchschnittliche Genauigkeiten des Modells mit dem Stanford Dog Datensatz . . . . .	37
14	Klassen des Obst Datensatzes mit konstantem Hintergrund . . . . .	39
15	Genauigkeits-Berechnungen des Modells des Obst Datensatzes . . . . .	39
16	Laufzeiten der Normalisierungs-Algorithmen mit verschiedenen großen Bildern	42

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>5</b>
1.1 Problemstellung und Ziele . . . . .	5
1.2 Erläuterung der These . . . . .	5
1.3 Anforderungen . . . . .	6
1.4 Struktur der Arbeit . . . . .	7
<b>2 Theoretische Grundlagen</b>	<b>8</b>
2.1 Digitale Bilder . . . . .	8
2.1.1 Aufbau von Digitalbildern . . . . .	9
2.2 Histogramme . . . . .	11
2.2.1 Einfluss von Belichtungen . . . . .	13
2.3 Künstliche Intelligenz . . . . .	13
2.4 Neuronale Netze . . . . .	14
2.4.1 Funktionsweise künstlicher und Faltender neuronaler Netze . . . . .	20
2.4.2 Entstehung von Trainingsdaten . . . . .	22
2.5 Mean average Precision . . . . .	23
2.5.1 Arten der Farbnormalisierung . . . . .	24
<b>3 Methodik und Durchführung</b>	<b>28</b>
3.1 Verwendetes Modell . . . . .	28
3.2 Trainingsdaten . . . . .	28
3.3 Normalisierung-Algorithmen . . . . .	31
3.3.1 Gray-World-Algorithmus . . . . .	31
3.3.2 Histogramm Ausgleich . . . . .	32
3.3.3 Histogramm Spezifikation . . . . .	33
<b>4 Ergebnisse</b>	<b>34</b>
4.1 Frameworks und Entwicklungsumgebung . . . . .	34
4.2 Nahrungsmittel Datensatz . . . . .	35
4.3 PascalVOC Datensatz . . . . .	35
4.4 Stanford Dogs Dataset . . . . .	36
4.5 Zwischenstand . . . . .	37
4.6 Obst Datensatz . . . . .	38
4.7 Auswertung des Obst Datensatzes . . . . .	39
<b>5 Diskussion</b>	<b>41</b>
5.1 Laufzeittest . . . . .	41
<b>6 Fazit</b>	<b>42</b>

# 1 Einleitung

In keinem Zeitabschnitt waren digitale Technologien so stark im Fokus wie heutzutage. Gerade der Bereich selbstständig lernender Computer wird in der Forschung untersucht und vorangetrieben. Auch bekannt unter dem Namen *Deep Learning* oder künstliche Intelligenz (KI), ist mit diesen Bezeichnungen meist der Bereich der künstlichen neuronalen Netze gemeint. Solche Netze können durch eingeführte Datensätze oder Regeln lernen. Dadurch werden sie in dem trainierten Bereich *intelligent*. Durch genügend Beispiele in den Trainingsdaten können sie Hypothesen aufstellen. Die Faktoren, welche beim Training wichtig sind, stellen zum einen die Menge, wie auch die Qualität der Trainingsdaten dar, zum anderen die Zeit, in welcher das neuronale Netz trainiert wird. Durch steigende Rechenleistung kann das Verfahren beschleunigt werden, da normalerweise viel Zeit für das Training benötigt wird. In der Vergangenheit konnten die Forschungen durch zu langsame Hardware nicht effektiv weiter geführt werden. Mit der Leistung heutiger CPUs und GPUs ist dies ohne Probleme möglich.

## 1.1 Problemstellung und Ziele

Beim Erstellen eines neuronalen Netzes können durch schlechte Trainingsdaten Schwierigkeiten entstehen. Im Bereich der Objekterkennung können dadurch Probleme, wie fehlerhafte Zuordnungen auftreten. Datensätze bestehen üblicherweise aus mehreren Hunderttausend Bildern und bei Vortrainierten um die Tausend Bilder. Die richtige Ausleuchtung ist ein wichtiger Aspekt bei der Generierung der Trainingsbilder, da schon kleine Veränderungen der Lichtverhältnisse die Farben der Objekte verändern können. Ein und dasselbe Objekt kann dadurch in vielen verschiedenen Farbvariationen auftreten, wie man in Abbildung 1 erkennen kann. Durch weniger Licht wirken die Farben dunkler und unterscheiden sich sichtlich. Gerade bei einem Datensatz mit wenig Bildern, kann das zu fehlerhaften Prognosen führen. Häufig kann eine konstante Ausleuchtung nicht gewährleistet werden, weil die Bilder in der freien Umwelt erstellt werden und eine Ausleuchtung zu umständlich wäre. Das bedeutet das dieses Problem hauptsächlich in der Nachbereitung der Bilder angegangen werden kann.

## 1.2 Erläuterung der These

Für solche Probleme bietet die Bildverarbeitung einige Lösungsansätze, welche von der Theorie aus, eine konstante Farbgebung und dadurch eine Erhöhung der Genauigkeit bringen könnten. Ob es in der praktischen Ausführung die erwarteten Ergebnisse erzielt, soll getestet werden. Deswegen sollen in dieser Arbeit verschiedene Verfahren der Farb-



Abbildung 1: *Szene mit unterschiedlicher Ausleuchtung*

normalisierung von Bildern auf die Trainings- und Testdatensätze angewendet und auf verschiedene künstliche neuronale Netze trainiert werden. Beim Vergleich der Trainingsergebnisse soll herausgestellt werden, ob das Normalisieren der Daten einen positiven Einfluss auf die Genauigkeit hat und welche Unterschiede die Normalisierung-Verfahren aufweisen.

### 1.3 Anforderungen

Bei dem geplanten Vorhaben, welches diese Arbeit thematisiert, ist es wichtig Anforderungen an das neuronale Netz, dem Datensatz und den verwendeten Algorithmen zu formulieren, um sicher zu stellen, dass keine vermeidbaren Probleme auftreten. Zunächst werden Trainingsdaten benötigt, mit welchen die neuronalen Netze trainiert werden. Viele wissenschaftliche Arbeiten zeigen, dass eine große Menge an Daten benötigt wird, um ein gut funktionierendes neuronales Netz zu entwickeln. Im Schnitt werden 50.000 Trainingsbilder pro Klasse benötigt. Diese Menge an Daten kann im Anbetracht der verfügbaren Zeit nicht für drei verschiedene Datensätze generiert werden. Da die benötigten Ressourcen nicht zur Verfügung stehen, soll eine Möglichkeit gefunden werden, gute Ergebnisse mit einer geringeren Menge an Trainingsdaten zu realisieren. Um genügend Vergleichswerte untersuchen zu können, sollen mehrere Datensätze mit unterschiedlichen Klassen genutzt werden. Um sicher zu stellen, dass die Datensätze zum Vorhaben geeignet sind, werden diese für die Arbeit erstellt und ausgesucht. Da diese für die Arbeit geeignet sein müssen, ist es wichtig, eine hohe Qualität an Daten zu verwenden.

Die Trainingsdaten sollen unter folgenden Anforderungen erstellt werden:

1. Ein qualitativ hochwertiger Datensatz sollte nicht unordentlich sein, da das Reingen der Daten lange dauern kann. Das bedeutet, dass Klassen getrennt voneinander

gespeichert werden, und die Namensgebung für jede Objektklasse klar definiert ist. Bei Problemfällen können Daten entfernt oder hinzugefügt werden.

2. Der Datensatz sollte Bilder enthalten welche eine nicht zu hohe Auflösung besitzen, da sie den Trainingsfortschritt zurückhalten und mehr Ressourcen benötigen. Je größer das Bild ist, desto mehr Zeit wird für die Verarbeitung benötigt
3. Beim Generieren der Trainingsbilder sollte darauf geachtet werden saubere und gut ausgeleuchtete Aufnahmen zu machen. Aufnahmen der Objekte, sollten nicht zu stark verschwimmen und unscharf werden.
4. Das Ziel des Datensatzes sollte gut Definierte werden. Namen und Thema des Datensatzes sollten auf einen gewissen Bereich beschränkt werden. (zum Beispiel, keine Gesichter zusammen mit Pflanzen kombinieren)

Um die Auswirkungen der Normalisierungsverfahren vergleichen zu können werden verschiedene Verfahren auf die selben Trainings- und Testdaten angewendet. Durch die Normalisierungsverfahren soll die Farbvarianz verringert werden.

## 1.4 Struktur der Arbeit

Für ein besseres Verständnis der einzelnen Entwicklungsschritte in den späteren Kapiteln werden in Kapitel 2.1 zuerst die Grundlagen der digitalen Bildverarbeitung beschrieben. Dabei soll zunächst vermittelt werden wie digitale Bilder entstehen und aus welchen Komponenten diese zusammengesetzt sind. Im Weiteren wird in Kapitel 2.4 auf die Funktionsweise künstlicher neuronaler Netze eingegangen. Hierbei werden die unterschiedlichen Schichten welche durchlaufen werden aufgeführt und beschrieben. Anschließend werden in Kapitel 3.3 die verwendeten Farbnormalisierungs-Verfahren erklärt und beschrieben. Der Ansatz der einzelnen Methoden wird erläutert, sowie die verwendeten Trainingsdaten und Modelle.

In der zweiten Hälfte der Arbeit wird es um die Durchführung und Ergebnisauswertung (Kapitel 4) des praktischen Teils der Arbeit gehen. In diesem werden die verwendeten Farbnormalisierungs-Verfahren zusätzlich in der technischen Umsetzung erläutert. Daraufhin werden die Ergebnisse und interpretiert. In einer kurzen Diskussion (Kapitel 5) werden diese behandelt. Nach der Diskussion wird nun das Ergebnis der Auswertung anhand der angeführten These evaluiert und ein Fazit (Kapitel 6) der Arbeit getroffen.

## 2 Theoretische Grundlagen

Das folgende Kapitel soll eine kurze und leicht verständliche Einführung sein, in dem die Theoretischen Grundlagen über den Aufbau von digitalen Bildern der einfluss von Belichtung sowie den verwendeten Bildbearbeitung Methoden gelegt. Anschließend folgt ein Überblick über die Grundlagen der künstlichen Intelligenz. Der letzte Abschnitt wendet sich den Schwerpunkten der künstlichen neuronalen Netze. Dabei werden hauptsächlich für die Arbeit relevante Grundlagen vermittelt.

### 2.1 Digitale Bilder

Damit der Ansatz der These, dass die Farbnormalisierung von Trainingsdaten das Trainingsverhalten und die spätere Genauigkeit künstlicher neuronaler Netze verbessern kann, nachvollzogen werden kann, soll in diesem Unterkapitel, auf die Aufnahme digitaler Bilder, mit deren Eigenschaften und Aufbau eingegangen werden. Außerdem sollen die Schwächen herausgearbeitet und mögliche Lösungsansätze erläutert werden. Zunächst folgt eine kurze Einführung.

Bildverarbeitung wird in der heutigen Zeit oft mit Bildbearbeitung assoziiert, also mit der Bearbeitung von Bildern mittels Bearbeitungssoftware. In der Bildverarbeitung geht es anders als in der Bearbeitung, um die Software selber, welche für die Konzeption und Erstellung von digitalen Bildern genutzt wird. In der Programmierung sind Bilder nichts weiter als Zahlen-Matrizen, wie in Abbildung 2 zu erkennen, welche man nach Belieben lesen und verändern kann, um seine geplanten Ziele zu erreichen. Grundsätzlich bestehen digitale Bilder aus rastern. Diese besitzen mehrere Werte, welche auch Pixel genannt werden. Die Werte entstehen je nach Menge der Farbabstufungen. Bei einem Graustufenbild sind das normalerweise 256 Abstufungen bei einem 8 bit Bild.

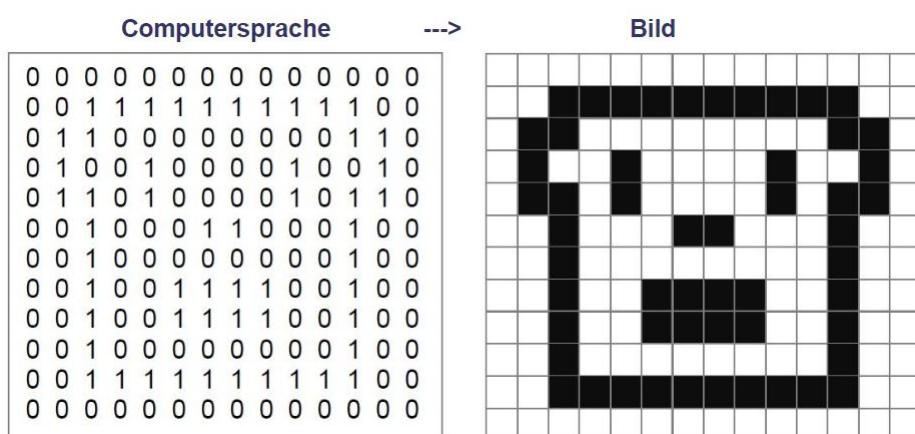


Abbildung 2: Aufbau eines Digitalen Bildes mit zwei Helligkeitsabstufungen

In diesen Pixeln werden Bildinformationen wie Farbe und Position festgehalten. Jeder Pixel besteht dabei oft aus drei Kanälen, den R Kanal für Rot, den G Kanal für Grün und den B Kanal für Blau. Diese bilden in bestimmten Kombinationen und Intensitäten ungefähr 16 Millionen verschiedene Farben. Der beschriebene Aufbau wird bei RGB-Bildern verwendet und auch bei Monitoren, Smartphones, etc. Neben dem RGB gibt es noch einige verschiedene Farbräume (CMY, CMYK, HSV, Lab, etc).

### 2.1.1 Aufbau von Digitalbildern

Üblicherweise wird bei Digitalbildern mit dem RGB-Farbraum gearbeitet. Aus diesem Grund, wird in dieser Arbeit hauptsächlich auf RGB-Bilder eingegangen. Ein Bild ist wie eine Matrix aufgebaut, dabei kann es sowohl quadratisch wie auch rechteckig Abmessungen haben. Jeder Punkt in der Matrix stellt einen Pixel dar und enthält jeweils einen Rot-, grün-, Blauwert welche zusammen eine beliebige Farbe darstellt. Um die Farbverteilung in einem Bild besser erkennen zu können, werden die Farbwerte in sogenannten Farb-Histogrammen zusammengefasst. Histogramme spielen für die Normalisierungsfunktionen eine Rolle, aus diesem Grund wird das Konzept hinter den Histogrammen und die spätere Manipulation, beschrieben. Für die Einfachheitshalber werden Graustufenbildern für die Erklärung verwendet. Anwendbar sind diese auch auf die Histogramme der einzelnen RGB-Kanäle.

#### RGB

Der RGB-Farbraum ist der geläufigste und bekannteste Farbraum dieser besteht wie in Abschnitt 2.1 kurz beschrieben aus drei Farbräumen (Rot-Grün-Blau). Dieses beschreibt, welche Art von Licht ausgestrahlt werden muss, um eine gewünschte Farbe zu erreichen. Die Kombination der einzelnen Farbräume ist eine additive Farbmischung. RGB ist eher ein Farbmodell, da es viele Farbräume gibt, welche sich davon ableiten (sRGB, AdobeRGB, etc.).

#### HSV

Anders als der RGB ist der HSV Farbraum aufgebaut, dennoch gibt es auch hier drei Werte. Das H (Hue) steht für den Farbton, welcher angenommen werden soll, S (Saturation) für die Sättigung der ausgewählten Farbe und V (Value) für den Hell-wert oder auch die Helligkeit der Farbe. Der HSV Farbraum besteht aus diesen Koordinaten. Dieser Farbraum ist gerade in der Kunst beliebt, da die Farbvorstellung besser funktioniert als bei additiven und subtraktiven Farbmischungen.

## **CMY/CMYK**

Anders als der RGB-Farbraum benutzt der CMY-Farbraum den subtraktiven anstelle der additiven Farbmischung. Hierbei sind die drei Grundfarben C für Cyan, M für Magenta und Y für Yellow. Bekannt ist diese Farbmischung gerade durch das Malen von Wasserfarben, oder für die Farbpigmente, welche in Druckern genutzt werden. Für den durch die Farbmischung kann der CMY kein richtiges Schwarz erreichen, deswegen wurde der Farbraum CMYK mit einem Parameter erweitert. Das K steht für Schwarz, damit auch schwarz richtig dargestellt werden kann und kein dunkles Grau entsteht.

## **Lab**

Der Lab-Farbraum ist ein Messraum, in welchem jede wahrnehmbare Farbe enthalten ist. Der Farbraum wurde auf Grundlage der Gegenfarbentheorie konstruiert. Eine wichtige Eigenschaft, ist die Geräteunabhängigkeit. Das bedeutet, die Farben werden unabhängig der Erzeugung und der Wiedergabe definiert. Die umfassende Definition kann in der DIN 6174 nachgelesen werden. Auf Grundlage der Gegenfarbentheorie liegen sich auf der a Achse die Farben Grün und Rot gegenüber. Die b Achse entspricht den Gegenfarben Blau und Gelb. Senkrecht durch die Ebenen befindet sich die L Achse welche von 0 (weiß) bis 100 (Schwarz) geht. Diese gibt die Helligkeit wieder und hat Zwischenschritte für die Graustufen.

## 2.2 Histogramme

Histogramme beschreiben eine Häufigkeitsverteilung [BB09, 42ff.]. Speziell bei Bildern zeigen Histogramme die Häufigkeit der auftretenden Intensität-werte. Am einfachsten lässt sich das mit Graustufenbildern nachvollziehen. Ein Beispiel dafür zeigt die (Abbildung 3) für ein Graustufenbild  $I$  mit möglichen Intensität-werten im Bereich  $I(n, V) \in [0, K - 1]$  enthält das Histogramm  $H$  genau  $K$  Einträge. Die Menge der Einträge bei einem 8 Bit Graustufenbild sind  $2^8$  Farbabstufungen ( $K = 2^8 = 256$ ). Der Wert des Histogramms an der Stelle  $i$  ist  $h(i)$  die Anzahl der Pixel von  $I$  mit einem Intensitätswert  $i$ , für alle ( $0 < i < K$ ). Mathematisch ausgedrückt ergibt sich  $h(i) = \text{card}(u, v) | I(u, v) = i$  daraus lässt sich entnehmen, dass  $h(0)$  die Anzahl der Pixel mit dem Wert 0,  $h(1)$  die Anzahl des Wertes 1 usw.,  $h(255)$  ist schließlich der letzte Wert und gibt die Menge aller weißen Pixel in dem Bild an ( $K - 1$ ). Eine Histogramm Berechnung stellt somit einen eindimensionalen Vektor  $h$  der Länge  $K$  da.

$\text{card}$  beinhaltet die Anzahl der Elemente einer Menge (*Kardinalität*)

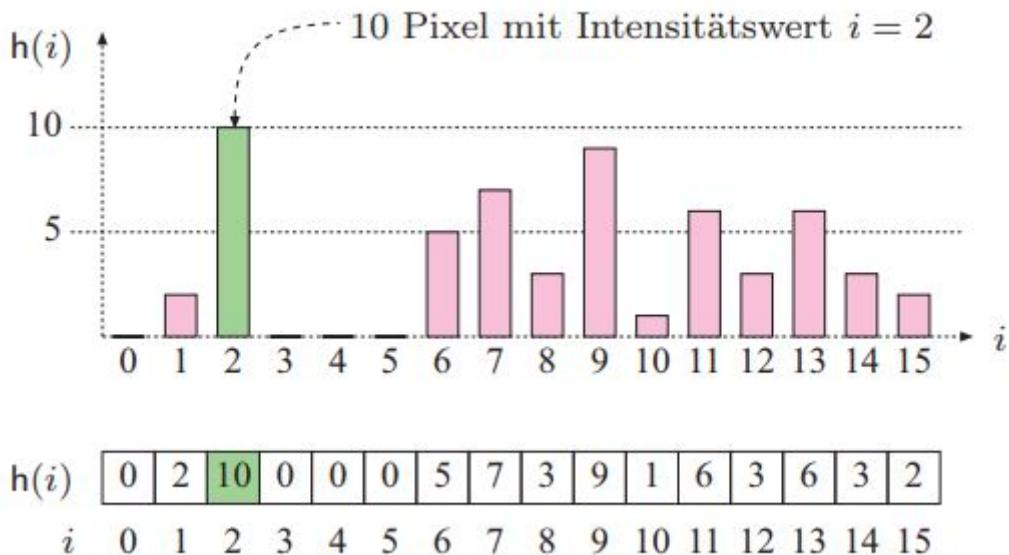


Abbildung 3: *Histogramm eines Graustufen Bildes mit 16 Helligkeitsabstufungen [BB09, S. 42]*

Die Abb.3 zeigt ein mögliches Histogramm mit einer Farbabstufung von 16 werten. Ein Histogramm zeigt die wichtigsten Eigenschaften eines Bildes, wie beispielsweise die Dynamik oder den Kontrast eines Bildes. Außerdem lässt sich erkennen, ob bei der Bildaufnahme Probleme aufgetreten sind. Diese lassen sich erkennen indem Ausschläge ganz links oder ganz rechts, des Graphen zu erkennen sind. Ein Ausschlag bei 0 oder 255 bedeutet das dort keine Farbinformationen mehr vorhanden sind, und das teile des Bildes, Über- beziehungsweise unterbelichtet sind.

## Kontrast

Der Kontrast bezeichnet den Bereich von Intensitätstufen, welche in einem Bild effektiv genutzt werden, also die Differenz der maximalen und minimalen Pixelwerten [BB09, S. 44]. Ein Bild welches einen hohen Kontrast hat, nutzt das gesamte Spektrums des Histogramms (Schwarz bis weiß). Der Kontrast, lässt sich also leicht aus einem Histogramm entnehmen, wie in Abbildung 5 die verschiedenen Histogramme zeigen.

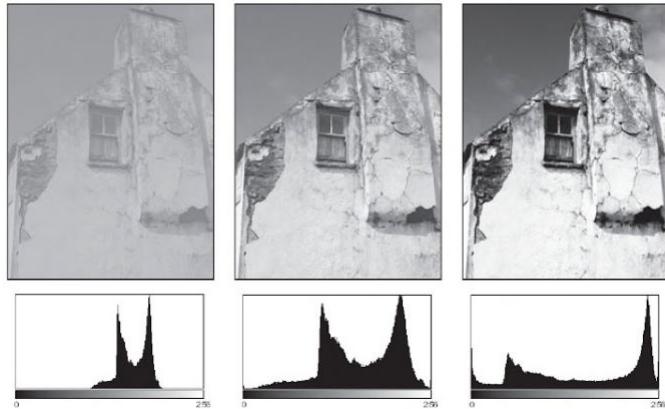


Abbildung 4: *Unterschiedlicher Kontrast und die Auswirkungen im Histogramm: niedriger Kontrast (links), normaler Kontrast (Mitte), hoher Kontrast (rechts)* [BB09, S. 45]

## Dynamik

Dynamik bedeutet, wie viel verschiedene Intensitätswerte in einem Bild genutzt werden, und die jeweilige Anzahl [BB09, S. 44]. Im besten Fall entspricht die Dynamik der insgesamt genutzten Anzahl an Pixelwerten (Abbildung 5). Also  $K - 1$  genutzter Farbabstufungen, damit der Wertebereich voll ausgeschöpft wird.

Anders als der Kontrast, welcher immer erhöht werden kann, solange der maximale Wertebereich nicht erreicht worden ist, kann die Dynamik eines Bildes nicht so leicht verbessert werden. Aus diesem Grund ist eine hohe Dynamik von Vorteil, denn dadurch wird die Gefahr von Qualitätsverlust bei Verarbeitungsschritten verringert. Trotz der Tatsache, dass viele Ausgabegeräte mit Farbtiefen von 8 Bit arbeiten, nehmen heutige Kameras bis zu 12-14 Bit pro Kanal auf, um einen möglichen Qualitätsverlust vorzubeugen.

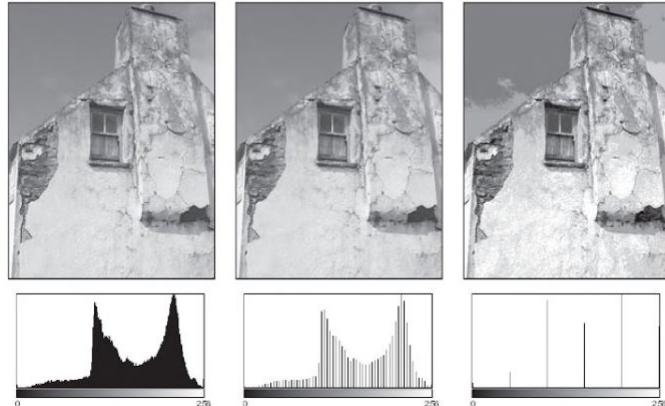


Abbildung 5: *Unterschiedlicher Dynamik und die Auswirkungen im Histogramm: hohe Dynamik (links), niedrige Dynamik mit 64 Graustufen (Mitte), extrem niedriger Kontrast mir 6 Graustufen (rechts)* [BB09, S. 45]

### 2.2.1 Einfluss von Belichtungen

In dem Kapitel 2.1.1 wurde Beschrieben wie die Farben in Bildern gespeichert werden und welche unterschiedlichen Arten von Farbräumen es gibt [BB09, 41ff.]. Bei der Aufnahme von Bildern ist gerade die Belichtung entscheiden. Eine hohe Belichtung sorgt dafür, dass Farben heller wirken, wobei wenig licht dafür sorgt das Farben dunkler erscheinen. Diese Tatsache kann beim Deep Learning zu Problemen führen, da ein und dasselbe Objekt unterschiedliche Farben annehmen kann und je nach Belichtungsintensität variiert. Ein künstliches neuronales Netz kann Objekte welche es normalerweise trainiert hat nicht immer erkennen, wenn die Farbe zu stark von der eigentlichen Farbe abweicht. Um Belichtungen auf Bildern auszugleichen und zu normalisieren, gibt es einige verfahren, welche diesem Problem entgegenwirken. Die Ausgewählten verfahren werden aufgeführt und beschrieben. Diese stellen nicht die Originalfarbe des Objektes wieder her, sondern sorgen für eine konstante Farbvariation, welche sich nicht stark verändert.

## 2.3 Künstliche Intelligenz

Künstliche Intelligenz wird heute immer mehr von Firmen und Wissenschaft genutzt. In den letzten Jahren hat die nutzung von künstlichen Inteligenzen enorm zugenommen und Firmen nutzen diese Technologie beispielsweise für die analyse und voher sage von Kundenverhalten, für die Klassifikation von Bildern oder zur Spracherkennung in heutigen Smartphones. Mit künstlicher Intelligenz beschreibt man solche Computer, welche gewisse Aufgaben erledigen, ohne ausdrücklich dafür programmiert worden zu sein. Dabei werden Informationen anhand von Vergleichsdaten dem System zugeführt, wodurch er sich Eigenschaften und Strukturen merkt und durch diese Rückschlüsse und Prognosen ziehen kann. Eine gute Beschreibung geben dabei die Autoren Rich und Knight, in dem

sie schreiben:

*Das Studium des Problems, Computer dazu zu bringen, Dinge zu tun, bei denen ihnen momentan der Mensch noch überlegen ist. - (Rich und Knight 1991)*

Ein gutes Beispiel in welchem man das Potenzial von künstlicher Intelligenz sehen kann, ist die Spiele KI *AlphaGo*. Diese KI wurde von Google entwickelt und ist auf das Spielen des Brettspiels Go trainiert. So konnte *AlphaGo* in einem Match 2016, den damaligen Weltmeister in Go *Lee Sedol* schlagen [Dee16]. In vier von fünf spielen gewann der Computer. Solche Ergebnisse konnten bis zu diesem Zeitpunkt mit anderen Methoden nicht erzielt werden, da es für damalige Algorithmen zu viele mögliche Spielzüge gab. Durch Deep Learning konnte der Computer ohne feste Algorithmen, besser als der Weltmeister seine Spielzüge Planen. Anhand der Definition von Rich kann man feststellen, dass *AlphaGo* im Bereich des Spieles Go als intelligent zählt, da sie dem besten Go-Spieler der Welt deutlich überlegen war.

## 2.4 Neuronale Netze

Um das Konzept hinter den künstlichen neuronalen Netzen besser verstehen zu können, betrachten wir zunächst wie der Lernprozess eines Menschen abläuft. Menschliches Lernen ist das Verändern der eigenen Verhaltensstrukturen und die Folgen individueller Erfahrungen [HE16]. Der Lernprozess, kann sowohl bewusst, so wie auch unbewusst unser Verhalten verändern. Menschen adaptieren also die eigenen als auch fremde Erfahrungen zu ihren Gewohnheiten. Diese Prozesse Finden im Gehirn statt, welches aus vielen Nervenzellen besteht. Diese Nervenzellen werden Neuronen genannt und sind untereinander verknüpft, weshalb auch von einem neuronalen Netz gesprochen wird. Maschinelles Lernen nimmt sich also die Funktionsweise des menschlichen Gehirns zum Vorbild [Ert13].

### Menschliches lernen

Wie bereits erwähnt, sind der Aufbau sowie die Funktionsweise von künstlichen neuronalen Netzen vom menschlichen Gehirn abgeleitet. Das Gehirn besteht aus ungefähr  $10^{11}$  Nervenzellen, welche untereinander verknüpft sind [Ert13, 265ff.]. Für den Lernprozess sind demnach viele Neuronen nötig. In der Abbildung 6 kann man den wesentlichen Aufbau eines Neurons sehen.

In diesem vereinfachten Modell eines Neurons lassen sich die wesentlichen Komponenten erkennen. Das Neuron selber besteht aus einem Zellkörper (Soma), einem Axon, den Dendriten auf der linken Seite und den Synapsen(Axon Terminals) auf der rechten Seite. Eine Dendrite stellt mit der Synapse eine Verbindung zu anderen Neuronen her. Diese senden elektrische Impulse an die Soma weiter, welche verarbeitet werden. Wenn die Spannung

## Neuron

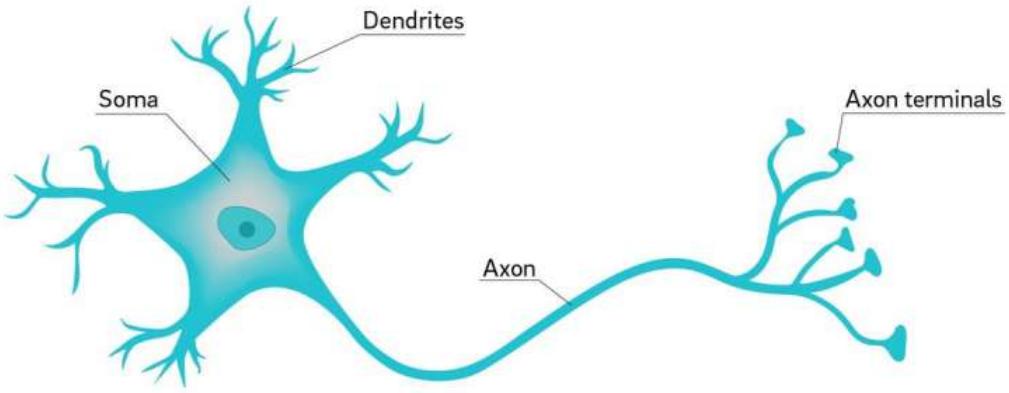


Abbildung 6: Aufbau eines Neurons [CSD18]

einen gewissen Schwellwert übersteigt, sendet die Soma einen elektrischen Impuls über das Axon. Jede Synapse sendet das Signal an die Dendriten des nächsten Neurons weiter. Die Stärke des Impulses und der Schwellwert, ist bei jedem Neuron unterschiedlich und verändern sich ständig. Der Schwellwert des Neurons ist niedriger, desto öfter es aktiv ist und steigt wenn es selten aktiv ist [ST13]. Die aufgeführte Funktionsweise beschreibt nur oberflächlich das Verhalten eines neuronalen Netzes und soll hauptsächlich dem weiteren Verständniss dienen.

## Perzeptron

Die Grundlage für den mathematischen Ablauf künstlicher neuronaler Netze wurde von McCulloch und Pitts in ihrem Buch *A logical calculus of the ideas immanent in nervous activity* erklärt. Auf diesen Grundlagen simulierte 1953 der Psychologe und Informatiker Frank Rosenblatt das erste künstliche Neuron, welches auch als Perzeptron bekannt wurde. In der Abbildung 7 wird der Aufbau eines solchen Perzeptrons dargestellt.

Als Eingabewert bekommt das Perzeptron ein Gewicht  $w$ , welches mit den Eingangswert  $x$  multipliziert wird. Das Ergebnis dieser beiden Werte, kann als lineare Funktion beschrieben werden.

$$f(x) = w * x \quad (1)$$

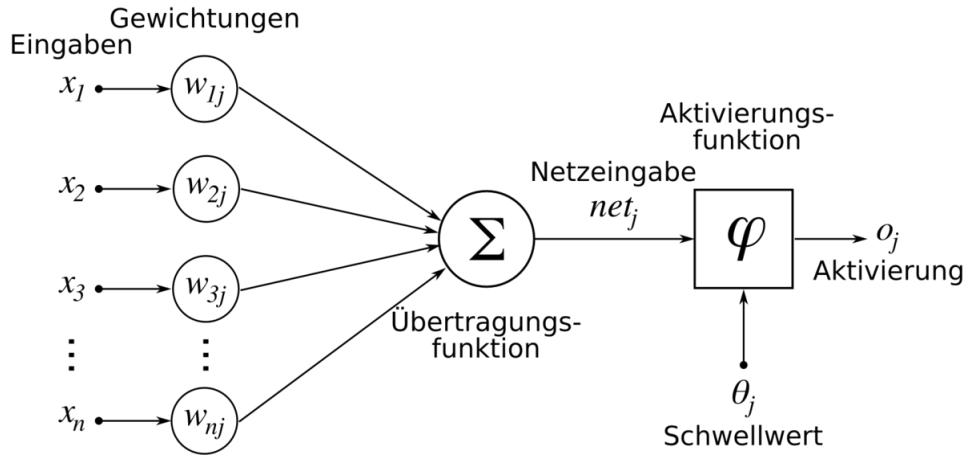


Abbildung 7: Schematische Modellierung eines Perzeptrons nach Rosenblatt [Wik19]

Das Ergebnis der beiden Werte wird an eine Aktivierungsfunktion  $g$  übergeben. Die Aktivierungsfunktion oder auch Schwellwert genannt, gibt ein Signal weiter, sobald dieser Wert überschritten wird.

$$g\left(\sum_{i=0}^n x_i * w_i\right) \quad (2)$$

Als Aktivierungsfunktion können sowohl lineare als auch nicht linearen Funktionen verwendet werden. Zusätzlich wird dem Ergebnis der Funktion der *Bias*  $b$  addiert.

$$g\left(\left(\sum_{i=0}^n x_i * w_i\right) + b\right) \quad (3)$$

Der *Bias* ist eine Art zusätzliches Neuron, für das Verringern von Konvergenz-Problemen, welche bei der Approximation der Funktion auftreten können. Der Bias hat immer einen Eingabewert von 1 und ein Gewicht  $w_{jb}$ , welches unabhängig der Eingabewerte ist.

$$\text{Bias} = 1 * w_{jb} \quad (4)$$

Maschinelles Lernen möchte unter anderem Maschinen in die Lage versetzen Objekte, Strukturen oder auch Abläufe anhand von vorher beigebrachten Beispielen eigenständig

zu erkennen. Eine Methode um dieses Ziel erreichen zu können sind die künstlichen neuronalen Netze. Diese werden mit Datensätzen des jeweiligen Schwerpunktes trainiert. Es gibt vier grundlegende Methoden ein solches Netz zu trainieren.

### **Überwachtes lernen**

Das überwachte Lernen funktioniert hauptsächlich mit Datensätzen, welche Eingabedaten sowie Ausgabedaten enthalten. Das künstliche neuronale Netz versucht bei Eingabe der Daten, eine Funktion aufzustellen, welche eine Hypothese darstellt. Diese beschreibt um, was es sich bei dem Datensatz handelt. Die Genauigkeit dieser Hypothese kann durch den Vergleich der enthaltenden Ausgabedaten ermittelt werden. Wenn die Möglichkeiten der Ausgaben endlich sind, handelt es sich um eine Klassifizierung (Bsp.: Milch, Orangensaft, Wasser, etc.). Sollten nur zwei Ausgabewerte möglich sein, ist es eine boolsche bzw. binäre Klassifizierung. Eine Zahl als Ausgabe wird als Regression bezeichnet.

### **Nicht überwachtes lernen**

Die zweite Methode wie ein künstliches neuronales Netz trainiert werden kann, ist eine gegenteilige Herangehensweise. Hier werden lediglich die Eingabedaten übergeben. Das neuronale Netz versucht selbstständig auftretende Muster, in den Daten zu erkennen. Als Beispiel könnte man die sinnvolle Zuordnung von Bildern anhand ähnlicher Muster nehmen. Ähnliche Datensätze werden in Kategorien aufgeteilt, jedoch nicht benannt.

### **Halb überwachtes Lernen**

Oft werden die beiden Methoden des überwachten und unüberwachten Lernen kombiniert, so hat man meist einen kleinen Datensatz mit Ein- und Ausgabedaten, welche aber nicht explizit wahr sein müssen und einen Teil nur mit Eingabeinformationen. Diese Methode wird häufig bei Umfragen verwendet, wo nicht klar ist, ob der Proband wahrheitsgemäß antwortet. Dabei kann ein systematischer Fehler entstehen. Daraus ergibt sich eine Mischung aus überwachten lernen mit systematischen Fehler und unüberwachten lernen, bei welchem das Netz nach häufig auftretenden Mustern sucht.

### **Verstärktes lernen**

Das verstärkte Lernen arbeitet auf einem Prinzip, mit welchem richtige Entscheidungen belohnt und falsche bestraft werden. Ein bekanntes Beispiel ist ein Netz, welches virtuell eine Rennstrecke ohne Fehler (nicht von der Strecke abkommen) beenden muss. Bei jedem Fehler muss von vorne begonnen werden, bis die Strecke beendet wird. Ein besonders bekanntes Beispiel ist die GO-KI *AlphaGo* [Dee16], welche nur durch die Informationen der Spielregeln trainiert wurde. Die KI wurde in diesem Spiel so gut, dass sie den Weltmeister *Lee Sedol* 2016 in vier von fünf Partien GO besiegte.

## Künstliche neuronale Netze

Nachdem etwas auf die Grundlagen die verschiedenen Lernverfahren eingegangen wurde, soll in diesem Abschnitt der Aufbau künstlicher neuronaler Netze und die Logik fokussiert werden. Der mathematische Ablauf der Schichten wird in einem späteren Unterkapitel behandelt.

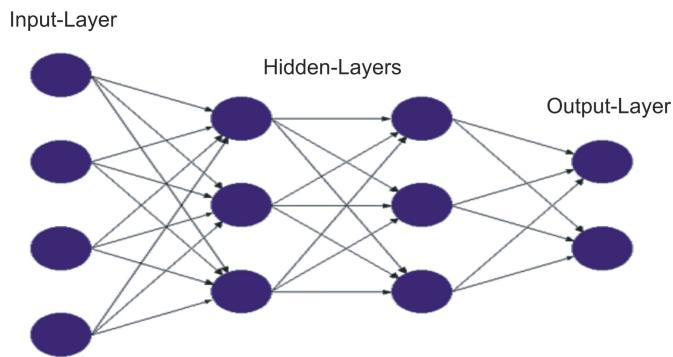


Abbildung 8: *Aufbau eines künstlichen neuronalen Netzes [Ang18]*

Der Aufbau von klassischen künstlichen neuronalen Netzen kann grundsätzlich in drei Bereiche unterteilt werden. Die erste Schicht stellt der *Input Layer* da. Dieser besteht wie alle folgenden Schichten aus mehreren Neuronen. In diese Schicht werden die Trainingsdaten eingegeben. Jeder Wert eines Datensatzes wird mit einem Neuron verbunden. Die Neuronen besitzen eine Gewichtung und einen Schwellwert. Wenn der Schwellwert durch den Eingabewert erreicht wird, sendet das Neuron das Ergebnis seiner Berechnungen an das Neuron der nächsten Schicht weiter. Die folgenden Schichten werden *Hidden Layer* genannt. Die Anzahl der *Hidden Layer* ist variabel und kann so viele Schichten enthalten wie benötigt. Diese Schichten werden auf mehrere verschiedene Merkmale der Eingabedaten trainiert. Dabei werden in frühen Schichten zunächst grobe Strukturen erkannt, welche nach jeder Schicht immer feiner werden. Die letzte Schicht des *Hidden Layer* über gibt seine Informationen an den *Output Layer*. Dieser stellt durch die erhaltenen Daten eine Hypothese auf, in welcher er beschreibt, wie er den Datensatz zuordnet. Dabei gibt das Netz an, zu welcher Wahrscheinlichkeit die Hypothese richtig sein sollte.

## Faltende neuronale Netze

*Convolutional neural Networks* oder auch faltende neuronale Netze (Abb. 9) sind anders als das KNN, an das Konzept des menschlichen Sehens angelehnt [SCL12] und für das Verarbeiten von matrixartigen Datensätzen konzipiert. Dazu zählen beispielsweise Bildaufnahmen [Goo+16]. Die erste Schicht besteht aus den *Convolutional Layer*. Bei dieser Schicht wird über die Pixel des Eingabebildes eine Schablone gelegt und vertikal wie auch horizontal verschoben. Diese Schablone hat eine ungerade quadratische Abmessung (3x3,

$5 \times 5$ ,  $7 \times 7$ ) und eigene Gewichtungen. Diese Schablone bildet ein Skalarprodukt der unterliegenden Pixel und der Gewichtung. Dabei werden die Strukturen hervorgehoben. Diese Schicht kommt mehrmals in einem Netz vor.

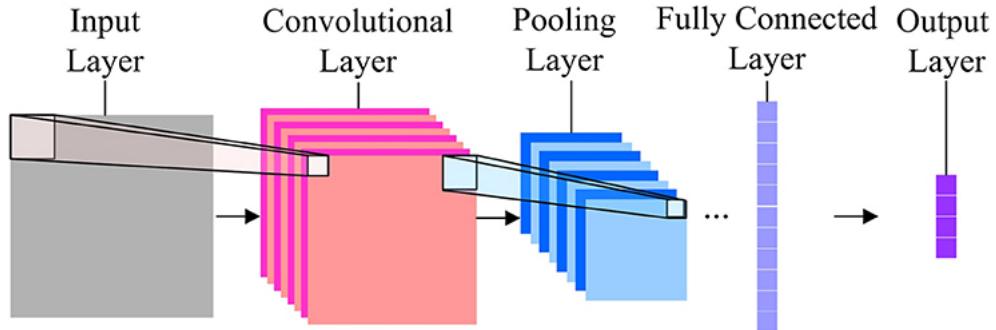


Abbildung 9: *Aufbau eines Convolutional neural Network [Pen+16]*

Auf den *Convolutional Layer* folgt immer der *Pooling Layer*. Dieser ist für die Komprimierung des Eingabebildes zuständig, damit im späteren Verlauf Rechenleistung gespart werden kann. Auch diese Schicht besteht aus einer Schablone welche über die Pixel des Bildes gelegt werden. Hierbei werden die Farbwerte der unterliegenden Pixel verarbeitet. Diese werden zu einem Wert zusammengefasst und übergeben. Nachdem alle *Convolutional* und *Pooling Layer* durchlaufen wurden, werden die Daten an die letzte Schicht (*Fully Connected Layer*) übergeben. Der *Fully Connected Layer* besteht wie beim KNN beschrieben, aus mehreren Perzeprontron-Schichten und gibt die Informationen an den *Output Layer* weiter, welcher schlussendlich eine Hypothese mit zugehöriger Wahrscheinlichkeit ausgibt.

### Loss Funktion

Bei jedem Trainingsschritt versucht das neuronale Netz, eine Hypothese aufzustellen. Damit das Netz Fortschritte im Lernen macht, ist das sogenannte, Backpropagation welches im nächsten Abschnitt genauer beschrieben wird zuständig, welche auf der Basis des Loss Graphen arbeitet. Die Loss Funktion berechnet den Fehler, welcher das Netz in seinem aktuellen Zustand in der Berechnung begeht, also die Differenz zwischen dem Ist- und Sollwert der Ausgabe. Beispiel einer solchen Funktion ist die euklidische Loss-Funktion

$$E_{Euclid} = \frac{1}{2N} \sum_{n=1}^N \|\hat{f}_n - f_n\|_2^2 \quad (5)$$

dabei stellt

$\hat{f}_n$   $[-\infty, +\infty]$  den berechneten Ist-Wert und

$f_n$  den Sollwert dar

Das Ergebnis der euklidischen Funktion ist das Maß für den Grad der Abweichung von ist und Sollwert.

## Rückpropagierung

Die Rückpropagierung oder auch *Backpropagation* [Ert13] ist unter anderem Teil des überwachten Lernens, und beinhaltet das Lernen aus Fehlern. Dadurch das bei einer Klassifizierung Datensätze mit einem Zielwert verwendet werden. Kann das Netz bei einer Zuweisung überprüfen, ob das Ergebnis zu dem Zielwert passt. Dabei wird aus der Ausgabe und dem Zielwert ein Fehler (Loss) berechnet. Sollte das Ergebnis zu weit vom Zielwert abweichen, wird überprüft, welches Perzeptron den größten Einfluss auf die Fehlzuweisung verursacht hat. Dieses Perzeptron wird in seiner Gewichtung angepasst, um den Fehlerwert zu verringern [Goo+16].

### 2.4.1 Funktionsweise künstlicher und Faltender neuronaler Netze

Die zuvor aufgeführten Netzarten haben einen unterschiedlichen Aufbau und bestimmte Einsatzgebiete. Das Convolutional neural Network ist beispielsweise auf das Verarbeiten von Bilddaten optimiert und Künstliche neuronale Netze eher für serielle Daten. In der Struktur sind die Netzarten ähnlich aufgebaut. Das CNN hat neben den üblichen Neuronenschichten zwei zusätzliche Schichten, welche beim KNN nicht vorhanden sind. In den folgenden Abschnitten soll die Funktionsweise dieser zusätzlichen schichten beschrieben werden.

## Faltungsschicht

Es existieren eine menge von Filtern  $n$  welche eine Höhe von  $f_h$  und einer Breite von  $f_w$  haben. Dieser Filter wird über das Eingabebild mit der Höhe  $e_h$  und der breite  $e_w$  gelegt und in ein Pixel schritten bewegt. Die Ergebnisse stellen eine Liste von Merkmalen da welche auf der Höhe  $m_h$  und der Breite  $m_w$  liegen. Formell ausgedrückt:

$$m_h = (e_h - f_h) + 1 \quad (6)$$

$$m_w = (e_w - f_w) + 1 \quad (7)$$

Um die Verringerung von Informationen nach der Faltung zu verhindern, werden an den Rändern der Bilder Polsterungen, auch *padding* genannt, angehangen [Goo+16, S. 343]. Diese Polsterungen bestehen aus Pixeln mit einem Wert von 0. Die Anzahl der zusätzlichen Pixel ergibt sich aus der Höhe und Breite des Eingabebildes, um wie viele Pixel das

Bild vergrößert wird. Formell ausgedrückt:

$$p_h = f_h - 1 \quad (8)$$

$$p_w = f_w - 1 \quad (9)$$

Die Faltungsschicht hat eigene Gewichte, welche beim Vorgang mit denen des Bildes zusammengerechnet wird [Goo+16, 331ff.]. Für die gleiche Schicht wird derselbe Filter verwendet. Die Weise der Berechnung wird in der Abbildung 10 beschrieben.

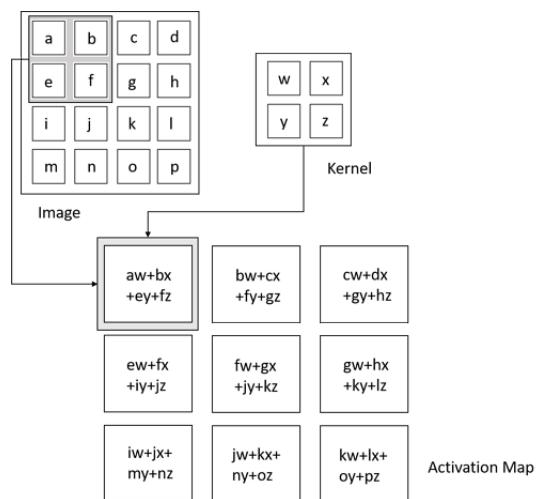


Abbildung 10: *Funktionsweise der Faltungsschicht*  
[Mis19]

## Pooling Schicht

Die Pooling Schicht oder auch *Pooling Layer* kommt nach jeder Faltungsschicht [Goo+16, 336f.]. Die Aufgabe dieser Schicht ist es das Eingabebild in der Auflösung und den zugehörigen Rechenaufwand für die folgenden Schichten zu reduzieren. Üblicherweise gibt es zwei Verfahren, welche beim Pooling eingesetzt werden. Zum einen das Average Pooling und das am häufigsten verwendete Max Pooling (Abbildung 11). Auch diese Schicht besteht aus einem quadratischen Filter, welcher wie bei der Faltungsschicht über die Bildpunkte gelegt wird. Dabei wird beim *Max Pooling* der höchste Wert in diesem Bereich ermittelt und übernommen. Bei einem standardmäßigem Kernel von  $2 \times 2$ , wird das Bild um den Faktor 2 Verkleinert. Die Abmessungen sind, anders wie bei der Faltungsschicht, gerade. **Fully Connected Layer** nachdem das Eingabebild durch alle Faltungs- und Pooling

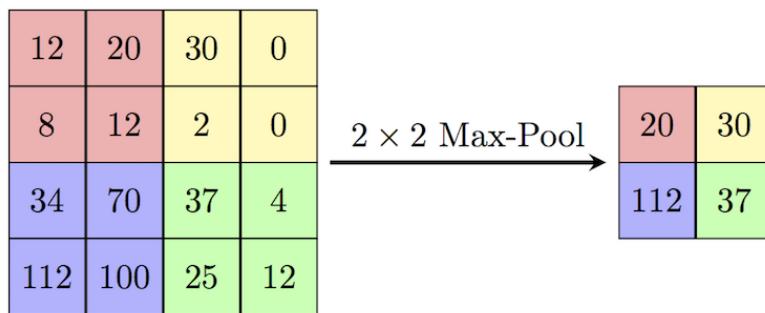


Abbildung 11: *Funktionsweise des Pooling Layers mit der Max Pooling variante [Wik18]*

Schichten verarbeitet wurde, werden die Daten an den Fully connected Layer übergeben [SCL12, S. 14]. Dieser verwendet, wie bei einem normalen KNN, Perzeptoren um die Daten zu verarbeiten. Die vorherigen Schichten wurden durchlaufen, damit diese rechenintensive Schicht nicht zu viel Informationen verarbeiten muss.

### 2.4.2 Entstehung von Trainingsdaten

Damit ein künstliches neuronales Netz Objekte erkennen und zuordnen kann, benötigt es zunächst eine Reihe von Beispieldaten. Anhand dieser Beispieldaten lernt das neuronale Netz, welche Eigenschaften das zu erlernende Objekt hat und wie es aufgebaut ist. Für diesen Vorgang werden üblicherweise mehrere Hunderttausend Daten benötigt, um ein akzeptables Ergebnis zu erzielen. Eine Möglichkeit, das Training mit weniger Trainingsdaten durchzuführen, ist das Verwenden vortrainierter Netze. Diese Netze wurden mit rund 300.000 Bildern über mehrere Tage trainiert und als Open Source zur Verfügung gestellt. Beim Verwenden dieser Netze werden lediglich die obersten Schichten, welche für die Zuweisung verantwortlich sind, durch den neuen Datensatz verändert wodurch wesentlich weniger Testdaten pro Objektklasse benötigt werden.

Eine Klassifizierung gehört zum überwachten Lernen und benötigt deswegen gelabelte Da-

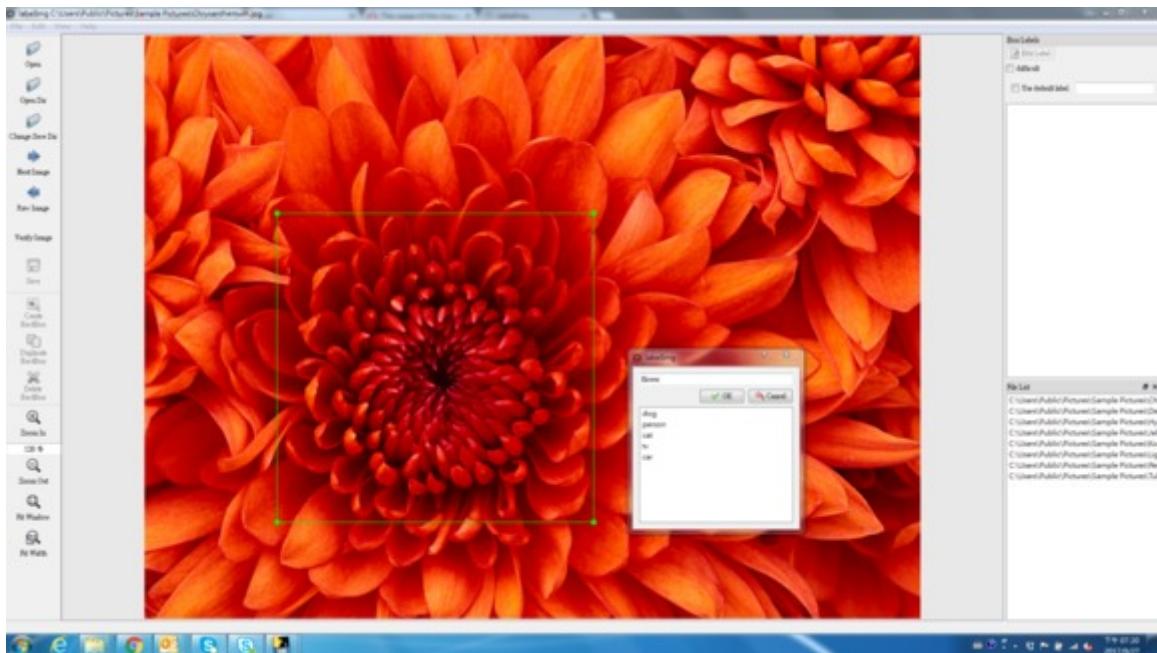


Abbildung 12: Vorbereitung der Datensätze mit dem Labeling Programm [tzu19]

tensätze. Das Labeln wurde mit dem Open Source Programm *LabelIMG* [tzu19] durchgeführt. Dabei wird das zu Trainierende Objekt mit einem rechteckigen Kasten umschlossen und einer Klasse auf den Bildern markiert. Diese Daten werden zunächst in einer XML Datei abgespeichert.

## 2.5 Mean average Precision

Oft können neuronale Netze nicht nur ein Objekt gleichzeitig erkennen, sondern geben als Ergebnis mehrere Hypothesen aus. Dadurch entsteht das Problem, das nicht mehr nur zwischen richtig und falsch unterschieden werden kann, sondern eine Genauigkeit benötigt wird, welche alle möglichen Objekte einbezieht. Eine Metrik welche auf diese Informationen eingeht, ist die (mean) average Precision (AP/mAP). Für die Berechnung wird die Präzision (Precision) und der Rückgabewert eines Ergebnisses benötigt. Die Präzision  $P$  beschreibt den Anteil der richtig erkannten Objekte unter allen erkannten Objekten. Der Rückgabewert  $R$  ist der korrekte Anteil von allen möglichen Objekten. Formell ausgedrückt:

$$P = \frac{\text{richtig erkannte}}{\text{richtig erkannte} + \text{falsch erkannte}} \quad (10)$$

$$R = \frac{\text{richtig erkannte}}{\text{richtig erkannte} + \text{falsche nicht erkannte}} \quad (11)$$

Häufig wird ein Schwellwert für die Pseudowahrscheinlichkeit genutzt, mit welchem entschieden wird, wann ein erkanntes Objekt der Rückgabeliste hinzugefügt wird oder nicht. Wird der Schwellwert niedrig angesetzt, werden meist mehr Objekte, als erkannt eingestuft. Dadurch steigt der Rückgabewert mit richtig erkannten Objekten. Gleichzeitig sinkt aber auch die Präzision da natürlicherweise auch mehr falsche Objekte erkannt werden. Objekterkennungsverfahren mit einer hohen Qualität zeigen aber weiterhin eine gute Präzision.

### 2.5.1 Arten der Farbnormalisierung

Die Farbnormalisierung ist ein Thema der Bildverarbeitung und wird hauptsächlich mit künstlicher Farbsicht befasst. Die Verteilung und Darstellung von Farben auf Bildern hängt hauptsächlich von Beleuchtungsbedingungen und der Kamera ab. Das bedeutet, dass sich die Farben bei der Aufnahme, je nach Beleuchtung verändern. Das ist gerade im Bereich von *Maschine Learning* problematisch, da diese Farbveränderungen zu Fehlern im Lernprozess und der späteren Genauigkeit führen. Farbnormalisierung-Algorithmen, sollen dafür sorgen, dass die Farbabweichungen durch Lichteinfluss geringere Auswirkungen haben und zu besseren Ergebnissen führen. Im folgenden werden die ausgewählten Algorithmen erläutert, welche im Rahmen dieser Arbeit getestet werden.

#### Gray-World-Algorithmus

Bei der Gray-World Normalisierung geht es nicht, wie der Name vermuten lässt um die Umwandlung in ein Graustufenbild, eher werden die verschiedenen Farbräume abgedunkelt und eingegrenzt. Es wird davon ausgegangen, dass das Farbspektrum durch drei konstante Faktoren modelliert werden kann. Dafür skalieren die Konstanten  $\alpha$ ,  $\beta$  und  $\gamma$  an die Kanäle R, G und B. Dadurch kann eine Konstanzlösung erzielt werden, welcher bei unterschiedlichen Belichtungen unveränderlich ist, indem jeder Farbkanal durch seinen Durchschnittswert geteilt wird, wie in folgender Formel beschrieben:

$$(\alpha R, \beta G, \gamma B) \rightarrow \left( \frac{\alpha R}{\frac{\alpha}{n} \sum_i R}, \frac{\beta G}{\frac{\beta}{n} \sum_i G}, \frac{\gamma B}{\frac{\gamma}{n} \sum_i B} \right) \quad (12)$$

Wie schon erwähnt, ist diese Art der Farbnormalisierung für verschiedene Farbvariationen unveränderlich. Ein größeres Problem dieses Normalisierung-verfahren besteht darin, das nicht alle Variationen der Beleuchtungsintensität berücksichtigt und auch nicht einfach für dynamische Szenen verwendet werden kann. Um solche Probleme zu lösen, gibt es mehrere Variationen dieses Algorithmus.

## Histogramm Ausgleichung

Die Histogramm Ausgleichung ist eine Nichtlineare Transformation, welche auf Grundlage der Histogramme wie in Kapitel 2.2 beschrieben arbeitet. Der Pixelrang wird dabei nicht verändert und kann bei jeder monoton steigenden Farbformation-Funktion verwendet werden. Diese Normalisierung-Funktion gilt stärker als der Gray-World-Algorithmus. Durch die Histogramm Ausgleichung entsteht ein dominanter blauer Kanal, wodurch das Bild oft unnatürlich erscheint.

Tabelle 1: *Pixelbild in ein tabellarisches Histogramm überführt*

0	1	5	1	7	2	0	3
0	0	5	5	5	2	4	5
4	5	1	4	1	5	1	4
5	1	2	4	5	2	6	3
5	2	6	4	0	4	0	5
4	0	2	4	7	4	6	2
5	1	6	1	0	1	1	5
4	3	2	4	2	5	2	5

Graustufen	0	1	2	3	4	5	6	7
Anzahl der Pixel	8	10	10	2	12	16	4	2

Aus der Pixelverteilung wird nun ein Ausgleich berechnet welche den kumulativen Pixelwert durch die Anzahl aller Pixel teilt und durch die höchste Graustufe teilt. Das Ergebnis wird gerundet und der passenden Graustufe zugeordnet. Die Vorgehensweise mit Farbbildern funktioniert gleich, nur das die R, G und B Kanäle verwendet werden.

Tabelle 2: *Mathematische Umsetzung der Histogrammausgleichung*

$r_k$	$P_k$	Kumulative Werte	$Kumulative/Gesamt * (L - 1)$	Gerundete nahe Grauwerte
0	8	8	$8/64 * 7 = 0.875$	1
1	10	18	$18/64 * 7 = 1.968$	2
2	10	28	$28/64 * 7 = 3.0625$	3
3	2	30	$30/64 * 7 = 3.2812$	3
4	12	42	$42/64 * 7 = 4.5937$	5
5	16	58	$58/64 * 7 = 6.3437$	6
6	4	62	$62/64 * 7 = 6.78125$	7
7	2	64	$64/64 * 7 = 7$	7

## Histogramm Spezifikation

Ähnlich wie die Histogramm Ausgleichung arbeitet die Histogramm Spezifikation. Die Histogramme der roten, grünen und blauen Kanäle werden so umgewandelt, dass sie den Formen von drei spezifischen Histogrammen entsprechen, anstatt sie einfach auszugleichen. Mithilfe dieses Vorgehens wird darauf abgezielt, Bilder zu erhalten welche ein Histogramm einer bestimmten Form haben. Zunächst muss das Bild so konvertiert werden, dass das Histogramm eine bestimmte Form hat. Für diese Methode werden üblicherweise zwei ähnliche Bilder verwendet, einmal das Hauptbild und das Referenzbild. Das Referenzbild hat die gewünschte Form des Histogramms, hingegen das Hauptbild nicht optimiert ist. Zunächst wird mit beiden Bildern eine Histogramm Ausgleichung durchgeführt. Im Weiteren wird das Referenz-Histogramm mit dem des Hauptbildes kombiniert und angepasst.

Tabelle 3: *Tabellarisches Histogramm des Quell-Bildes (B1)*

Graustufen	0	1	2	3	4	5	6	7
Anzahl der Pixel	8	10	10	2	12	16	4	2

Tabelle 4: *Tabellarisches Histogramm des Referenz-Bildes (B2)*

Graustufen	0	1	2	3	4	5	6	7
Anzahl der Pixel	0	0	0	0	20	20	16	8

Um die Funktionsweise verständlicher darzustellen, wird kurz auf das Vorgehen und die mathematische Herleitung eingegangen. Dafür gibt es in den Tabellen?und? zwei Histogramme mit 8 Graustufungen. In diesen Tabellen wird aufgeführt, wie oft der jeweilige Grauwert in dem Bild vorkommt. Nun werden die Histogramme wie in Absch2.2 beschrieben, ausgeglichen. Von 0 bis  $L - 1$  werden die Summen der Pixel berechnet, durch die Anzahl der im Bild enthaltenen Pixel geteilt und mit der höchsten Graustufe ( $L - 1$ ) multipliziert. Das Ergebnis des Ausgleiches wird in Tabelle? Aufgeführt. Nun werden die Grauwertverteilungen von B1 und die Verteilung von B2 angepasst dafür wird der nächste Wert aufgerundet von B1 übernommen.

In einem Beispiel:

- 1 (B1) am nächsten an 2 (B2) liegt bei Graustufe 4
- 2 (B1) am nächsten an 2 (B2) liegt bei Graustufe 4
- 3 (B1) am nächsten an 4 (B2) liegt bei Graustufe 5
- 3 (B1) am nächsten an 4 (B2) liegt bei Graustufe 5

.....

Tabelle 5: Von zwei Ausgeglichenen Histogrammen zu einem Spezifizierten Histogramm

Graustufe	Ausgleichung B1	Ausgleichung B2	Spezifiziertes Histogramm
0	1	0	4
1	2	0	4
2	3	0	5
3	3	0	5
4	5	2	6
5	6	4	6
6	7	6	7
7	7	7	7

Aus der neuen Stufen Verteilung entsteht ein neues Histogramm welches weniger Farbstufen enthält. Dadurch verschiebt sich das Histogramm Richtung weiß und das Bild wird erhellert.

Tabelle 6: Tabellarisches Histogramm des Quell-Bildes (B1) auf Basis des Referenz-Bildes (B2)

Graustufen	0	1	2	3	4	5	6	7
Anzahl der Pixel	0	0	0	0	18	12	28	6

### 3 Methodik und Durchführung

Nach der theoretischen Grundlage soll sich der zweite Teil der Arbeit um die Versuchsbeschreibung, Durchführung und Auswertung der Ergebnisse handeln. Dabei werden zunächst die verwendeten Datensätze sowie die Umsetzung der Normalisierungs Algorithmen. In den Ergebnissen, wird auf die Versuchsumgebung und die verwendeten Frameworks eingegangen, und wie diese für den Versuch genutzt wurden.

#### 3.1 Verwendetes Modell

Das Unternehmen COCO [Con18] (Common Objects in Contexts) besitzen eine Vielzahl an Datensätzen mit welchen, neuronale Netze trainiert werden können. Einige dieser Netze wurde auf der Object detection API von Tensorflow trainiert und stehen als Open Source zur Verfügung. Die folgende Tabelle führt ausgewählte Modelle auf, um die Genauigkeit und Geschwindigkeit vergleichen zu können.

Tabelle 7: *Modelle [LLC18] welche mit dem COCO Dataset Trainiert wurden [Con18]*

Modell	mAP	Geschwindigkeit
faster_rcnn_inception_v2_coco	28	58
faster_rcnn_resnet50_coco	30	89
rfcn_resnet101_coco	30	92
ssd_mobilenet_v1_coco	30	21
ssd_resnet_50_fpn_coco	35	76

Das faster\_rcnn\_inception\_v2\_coco soll für das Projekt verwendet werden. Dieses hat eine mAP von 28 und eine Geschwindigkeit von 32 ms. Dadurch dass es die niedrigste mAP hat, sollten die Unterschiede in der Genauigkeit stärker ausfallen. Das ausgewählte Modell wird im folgenden auf die Datensätze, welche im anschließenden Unterkapitel aufgeführt sind.

#### 3.2 Trainingsdaten

Da für ein solches Training die benötigten Trainingsdaten und die Zeit nicht zur Verfügung stehen, sollte eine Möglichkeit gefunden werden, wie das Training auch mit weniger Trainingsdaten durchgeführt werden kann. Durch das vortrainieren der neuronalen Netze, werden nur noch durchschnittlich 150 Bilder pro Objektklasse und eine geringere Trainingszeit benötigt. Die Bilder der eigenen Daten werden mit einer normalen 12 Megapixel Kamera aufgenommen. Die verwendeten Objekte, werden von allen möglichen Seiten fotografiert, dabei müssen für das Training, pro Klasse ungefähr 150 Bilder vorhanden sein. Da die Bilder mit einer Kamera mehrere Megabyte groß werden, müssen diese auf

100-300 KB komprimiert werden.

Damit am Ende des Projektes möglichst gute Vergleichswerte entstehen, werden mehrere Datensätze vor den Versuch verwendet. Insgesamt werden die neuronalen Netze mit 3 unterschiedlichen Datensätzen trainiert. Dafür wurden 3 Datensätze heraus gesucht. Der erste Datensatz welcher getestet wird, ist der Nahrungsmittel Datensatz eines früheren Projektes, zum anderen der Datensatz von *Pascal Visual Object Classes* (PascalVOC) und der Hunderassen Datensatz, Stanford Dog Dataset.

In den folgenden Tabellen werden alle Datensätze mit den zugehörigen Klassen und Anzahl an Trainingsbildern aufgeführt:

Der erste Datensatz besteht aus mehreren Nahrungsmitteln, welche in der Tabelle 8 jede Klasse besitzt um die 100-150 Bilder welche zusätzlich Annotations-Dateien mit den Koordinaten der Klassifizierungsboxen beinhalten. Dieser Datensatz ist mit seinen ca. 1000 Bildern der kleinste Datensatz

Tabelle 8: *Datensatz mit Nahrungsmitteln*

Klassenname	Klassenname
Milch - Packung	Orangensaft - Packung
Wasser - Flasche	Bier - Flasche
Brunch - Aufstrich	Margarine - Aufstrich

Der PascalVOC Datensatz umfasst 20 Klassen und besteht aus 5.000 Bildern. Von 2005 bis 2012 wurde jährlich die PascalVOC Challenges durchgeführt. Dabei sollte das beste Verfahren für die Segmentierung, Klassifikation und Objekterkennung ermittelt werden. Inhaltlich befasst sich der Datensatz mit einer Reihe unterschiedlicher Klassen, welche in Tabelle 9 zu erkennen sind. Es gibt einige Unterschiede in der Häufigkeit der auftretenden Klassen. Beispielsweise sind in rund 2000 Bildern insgesamt 4.690 Personen enthalten, weswegen es möglicherweise Unterschiede in der Genauigkeit untereinander, der Klassen geben könnte. Die Aufnahmen der Bilder sind thematisch und von der Art der Aufnahme unstrukturiert, was bedeutet, dass teilweise Bilder von einzelnen Objekten, Gruppen von Objekten oder auch ganze Szenen enthalten sind.

Tabelle 9: *Pascal Visual Object Classes [Eve+]*

Klassenname	Klassenname	Klassenname	Klassenname
Person	Vogel	Katze	Kuh
Hund	Pferd	Schaf	Zug
Flugzeug	Fahrrad	Boot	Bus
Auto	Motorrad	Flasche	Stuhl
Tisch	Blumentopf	Sofa	Bildschirm

Der dritte Datensatz welcher in dieser Arbeit verwendet wird, ist der Hunde Datensatz aus Stanford und beinhaltet 120 verschiedene Hunderassen. Jede Klasse hat um die 200 Bilddaten. Wegen der Struktur des Datensatzes, in einzelnen Ordner, kann eine Auswahl der priorisierten Hunderassen zusammengestellt werden. Für den Versuch und damit eine bessere Übersicht erreicht werden kann, wurden 20 Hunderassen ausgewählt. Die Annotationen der Daten sind in Form von XML Dateien beigefügt. In der folgenden Tabelle werden die ausgewählten Klassen des Datensatzes zusammengefasst.

Tabelle 10: *Stanford Dog Dataset [Kho+11]*

Klassenname	Klassenname	Klassenname	Klassenname
Chihuahua	Japanese spaniel	Maltese dog	Pekinese
Shih-Tzu	Blenheim Spaniel	Papillon	Toy Terrier
Rhodesian Ridgeback	Afghan Hound	Basset	Beagle
Bloodhound	Bluetick	coonhound	Walker Hound
Redbone	Borzoi	Irish Wolfhound	Italian Greyhound

Von der Struktur der Datensätze, sind sie ähnlich aufgebaut. Viele verschiedene Klassen mit verschiedenen Szenen und Kontexten. Im weiteren wird zunächst auf die Umsetzung der Normalisierungsverfahren eingegangen, um weiter Erkenntnisse zu erlangen.

### 3.3 Normalisierung-Algorithmen

Für die Normalisierung der Datensätze ist eine Methode nötig, um mehrere Bilder möglichst schnell hintereinander zu bearbeiten. Für die Normalisierung wurden Python Programme geschrieben, welche nacheinander die Bilder, anhand eines Algorithmus verarbeitet. Dafür wurde mit unter die *Python Image Library* und OpenCV genutzt. Beide Bibliotheken werden für das verarbeiten von Bildern benötigt.

#### 3.3.1 Gray-World-Algorithmus

```
1 # Importieren des Bildes
2 nimg = cv.imread(i, 1)
3
4 # Umwandlung des Bildes und der Kanäle in 32 Bit
5 nimg = nimg.transpose(2, 0, 1).astype(np.uint32)
6
7 # Zwischenspeichern des durchschnittlichen Grünkanals in die Variable mu_g
8 mu_g = np.average(nimg[1])
9
10 # Berechnung des rot-Kanals:
11 nimg[0] = np.minimum(nimg[0]*(mu_g/np.average(nimg[0])),255)
12
13 # Berechnung des blau-Kanals:
14 nimg[2] = np.minimum(nimg[2]*(mu_g/np.average(nimg[2])),255)
15
16 Zurückwandlung des Bildes in 8 Bit
17 img_output = nimg.transpose(1, 2, 0).astype(np.uint8)
```

In dem verwendeten Gray World Algorithmus [Aih12] zunächst wird das Bild in 32 Bit umgewandelt. Daraufhin wird der Grünkanal verwendet und der Durchschnitt berechnet und zwischengespeichert. Der Grün-Kanal wird nun mit dem durchschnitt des Rot-Kanals dividiert. Das Ergebnis des wird mit dem Minimum des Rot-Kanals multipliziert und übernommen. Der gleiche Vorgang folgt mit dem Blau-Kanal. Das verarbeitete Bild wird nun wieder zurück in 8 Bit gewandelt und ausgegeben (Abbildung 13). Für das verfahren wird keine spezielle Python Bibliothek eingesetzt.



Abbildung 13: *Auswirkung des Gray-World Algorithmus*

### 3.3.2 Histogramm Ausgleich

```

1 # Importieren des Bildes
2 img = cv2.imread('input.jpg')
3
4 # Umwandlung des Farbraumes von BGR zu YUV
5 img_yuv = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
6
7 # Ausgleichung vom Histogramm des Y Kanals
8 img_yuv[:, :, 0] = cv2.equalizeHist(img_yuv[:, :, 0])
9
10 # Umwandlung des Farbraumes in den BGR Farbraum
11 img_output = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)

```

Für die Histogramm Ausgleichung [Jaz16] wird das Bild zunächst vom BGR Farbraum in den YUV Farbraum umgewandelt. Der Farbraum ist ähnlich dem Lab Farbraum (Absch.2.1.1). Der Y Kanal wird für die Histogramm Ausgleichung verwendet. Das hat den Grund, dass das Luminanzsignal oder auch Leuchtdichte-Signal die Summe der drei Grundfarben Rot Grün und Blau und die Helligkeitsinformation enthält. Das Histogramm des Y Kanals wird, wie in Kapitel 2.2 beschrieben. Das normalisierte Bild wird von dem YUV Farbraum zurück in den BGR Farbraum konvertiert und zwischengespeichert. Das Ergebnis wird als Bild ausgegeben (Abbildung 14).



Abbildung 14: *Auswirkung der Histogramm Ausgleichung*

### 3.3.3 Histogramm Spezifikation

```

1 # Importieren des Referenzbildes
2 reference = io.imread('reference.jpg')
3
4 # Importieren des zu Normalisierenden Bildes
5 image = io.imread('image.jpg')
6
7 # Weiterleitung der beiden Bilder an die match_histogram-Methode der
     Skimage Bibliothek
8 matched = match_histograms(image, reference, multichannel=True)

```

Der dritte Normalisierungsfunktion welche für die Datensätze angewendet wird, ist die Histogramm Spezifikation (Absch.2.5.1) oder auch *Histogramm Matching*. Für den Algorithmus wird die Python Bibliothek *Scikit-image* verwendet. Dafür werden ein Referenzbild und ein Zielbild geladen. Diese beiden Bilder, werden mit der Funktion *match\_histograms* verarbeitet. Das Zielbild wurde auf das Histogramm des Referenzbildes angepasst. In dem Beispiel 15 kann man gut erkennen, das das Quellbild, welches stark verdunkelt ist, durch ein gut ausgeleuchtetes Referenzbild, eine wesentlich bessere Helligkeit aufweist (Abbildung 15). Die Problematik bei dieser Normalisierungs Methode besteht darin, das ein einheitliches Referenzbild genutzt werden muss auf welchem dann der gesamte Datensatz Normalisiert wird.



Abbildung 15: Das Zielbild wurde mithilfe des Referenzbildes ausgeglichen und angepasst. Auf der linken Seite ist das Ergebnis der Spezifikation

## 4 Ergebnisse

Nachdem alle Trainingseinheiten mit den verschiedenen Normalisierungsverfahren durchlaufen wurden, sollen die Ergebnisse aufgeführt und verglichen werden. Zunächst wird auf die Entwicklungsumgebung beschrieben und im Anschluss auf die Ergebnisse der Auswertung. Als Vergleichswerte konnte mithilfe von Tensorboard die Genauigkeiten des Netzes erhoben werden. Jede Klasse hat eine eigene durchschnittliche Genauigkeit (AP) und eine Genauigkeit für das gesamte Netz in Form einer mAP. In den folgenden Tabellen werden bestimmte Abkürzungen verwendet: N = Normaler Datensatz, GW = Gray-World-Algorihmus, HA = Histogramm Ausgleich, HS = Histogramm Spezifikation (Die besten Ergebnisse einer Klasse wurden durch Fettschrift hervorgehoben)

### 4.1 Frameworks und Entwicklungsumgebung

Wie in vorherigen Kapiteln schon beschrieben ist das Trainieren eines künstlichen neuronalen Netzes oder eines faltenden neuronalen Netzes sehr rechenintensiv. Die enthaltenen Daten werden parallel mittels Matrizenmultiplikation verarbeitet. Aus diesem Grund werden für die Verarbeitung Grafikprozessoren hauptsächlich verwendet da diese im Gegensatz zu normalen Prozessoren für die Berechnung von Matrizen konzipiert wurden.

Auf Softwareseite gibt es eine Menge an Frameworks, mit welcher sich CNNs realisieren lassen. Einige von denen sind beispielsweise Tensorflow, Keras und Theano. Wegen der vorliegenden Erfahrung und der Möglichkeit Trainingsfortschritte grafisch anzeigen zu können, wird der praktische Teil der Arbeit mittels Tensorflow durchgeführt. Tensorflow ist ein Open Source Framework von Google, welches zum entwickeln künstlicher neuronaler Netze genutzt werden kann. Programmiert wurde das Framework in den Programmiersprachen C und Python, welche auch für die Entwicklung genutzt werden. Mit

dem Framework Tensorflow kommen einige Möglichkeiten. Zum einen gibt es eine übersichtliche Dokumentation, zum anderen werden vortrainierte Netze, Open Source und für die Entwicklung die Tensorflow Object Detection API angeboten. Zur Verfügung stehen verschiedenste Netze auf Basis von verschiedenen Datensätzen.

## 4.2 Nahrungsmittel Datensatz

Der erste Datensatz welcher untersucht wurde ist der Datensatz, welcher in der Tabelle 11 aufgeführt ist. Dieser enthält im Vergleich zu den anderen Datensätzen weniger Objektklassen und weniger Bilder. Außerdem ist, sind die Objekte unter ähnlichen Bedingungen entstanden. Das bedeutet, dass die Lichteinstrahlung in den meisten Fällen gleich ist und nur auf wenigen unterschiedlichen Hintergründen aufgenommen wurden. In der Tabelle 11 wurde der Nahrungsmittel Datensatz verwendet und enthält die Informationen des originalen Datensatzes, des Gray-World Algorithmus, des Histogramm Ausgleichs und der Histogramm Spezifizierung. Im genauen Betrachten fällt auf, dass kaum nennenswerte Unterschiede bei GW entstanden sind. In manchen Punkten können Verbesserungen in der Genauigkeit ausgewertet werden(Wasser, Orangensaft, Bier), in anderen wiederum ist die Genauigkeit gesunken(Milch, Brunch, mAP).

Bei den Histogramm Normalisierungen (Ausgleich und Spezifizierung) kann mehr interpretiert werden. Auch hier sind die Unterschiede nur in einzelnen Objektklassen besser geworden und in anderen zurückgegangen. Die Varianz der verschiedenen Netze liegt bei gerade mal 0.05, was in der Praxis keinen bedeutenden Unterschied macht. Dennoch schneiden die Histogramm Normalisierungen insgesamt am schlechtesten ab.

Tabelle 11: Durchschnittliche Genauigkeiten des Modells mit dem Nahrungsmittel Datensatz

Klassenname	AP(N)	AP(GW)	AP(HA)	AP(HS)
Wasser - Flasche	0.948	<b>0.955</b>	0.934	<b>0.917</b>
Orangensaft - Packung	0.999	<b>0.998</b>	<b>1.000</b>	0.999
Milch - Packung	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
Margariene	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>	<b>1.000</b>
Brunch - Aufstrich	<b>1.000</b>	0.995	<b>0.959</b>	0.975
Flasche - Bier	<b>0.997</b>	<b>0.970</b>	0.986	0.979
mAP	<b>0.991</b>	0.987	0.980	<b>0.978</b>

## 4.3 PascalVOC Datensatz

Beim Auswerten der Genauigkeit des normalen Modells fällt auf, das teilweise große Unterschiede der einzelnen Objektklassen auftreten. Das kann daher resultieren, dass unterschiedlich viele Bilder pro klasse enthalten sind. Ähnlich wie beim vorherigen Datensatz konnte zwischen dem originalen Datensatz und der Gray-World-Normalisierung

kein deutlicher Unterschied festgestellt werden. Einige Klassen haben zwar eine höhere durchschnittliche Wahrscheinlichkeit, dennoch gibt es auch hier mehrere Bereiche, wo die Genauigkeit abgenommen hat. Auch die mAP hat ca. 0.008 Genauigkeit verloren. Anders als beim Nahrungsmittel Datensatz schneidet der Histogramm Ausgleich schlechter ab. Nur ein paar Objektklassen konnten besser erkannt werden. Insgesamt verliert das neuronale Netz 0.034 der Genauigkeit, was einen wichtigen Unterschied macht. Einige Objektklassen sinken durch die Ausgleichung unter die 50% Genauigkeit. Hier wird auch die Schwäche der Histogramm Spezifikation deutlich. Durchschnittlich 0.045 Genauigkeit wird durch die Normalisierung eingebüßt.

Tabelle 12: *Durchschnittliche Genauigkeiten des Modells mit dem PascalVOC Datensatz*

Klassename	AP(N)	AP(GW)	AP(HA)	AP(HS)
sofa	0.611	0.612	<b>0.637</b>	<b>0.602</b>
aeroplane	0.845	<b>0.855</b>	<b>0.826</b>	0.843
horse	<b>0.924</b>	0.910	<b>0.891</b>	0.905
train	0.790	<b>0.804</b>	<b>0.797</b>	0.817
bird	<b>0.750</b>	0.733	<b>0.691</b>	0.708
tvmonitor	0.729	<b>0.733</b>	0.725	<b>0.691</b>
boat	<b>0.680</b>	0.644	0.626	<b>0.559</b>
pottedplant	<b>0.430</b>	0.415	0.425	<b>0.340</b>
bus	0.797	<b>0.810</b>	<b>0.763</b>	0.776
diningtable	0.532	<b>0.545</b>	<b>0.507</b>	0.517
car	0.812	<b>0.815</b>	0.789	<b>0.775</b>
bottle	<b>0.553</b>	0.510	0.498	<b>0.451</b>
cat	<b>0.903</b>	0.902	0.872	<b>0.841</b>
person	<b>0.791</b>	0.779	0.775	<b>0.757</b>
chair	0.499	<b>0.515</b>	0.460	<b>0.440</b>
bicycle	0.744	0.730	<b>0.752</b>	<b>0.697</b>
cow	<b>0.718</b>	0.698	0.694	<b>0.653</b>
motorbike	<b>0.739</b>	<b>0.722</b>	0.725	0.725
dog	<b>0.867</b>	0.862	0.835	<b>0.811</b>
sheep	0.645	0.619	<b>0.664</b>	<b>0.561</b>
mAP	<b>0.718</b>	0.710	0.698	<b>0.673</b>

#### 4.4 Stanford Dogs Dataset

Auch bei Überprüfung des dritten Datensatzes fallen ähnliche Muster auf. Der GW-Algorithmus weiß bei diesem Datensatz sogar bessere Ergebnisse wie der originale Datensatz auf. Beim vergleichen der Datensätze fällt hier auf, das mehr verschiedene Lichtfarben genutzt wurden, als bei den anderen beiden Datensätzen. Diese konnten vom GW-Algorithmus vereinheitlicht werden. Beim Histogramm Ausgleich und der Spezifikation wurden wie auch bei den vorherigen Datensätzen schlechtere Genauigkeiten beobachtet.

Tabelle 13: Durchschnittliche Genauigkeiten des Modells mit dem Stanford Dog Datensatz

Objektklasse	AP(N)	AP(GW)	AP(HA)	AP(HS)
Chihuaua	0.729	0.854	0.780	0.746
Shih-Tzu	0.864	0.879	0.857	0.832
Rhodesian Ridgeback	0.741	0.706	0.711	0.699
Bloodhound	0.872	0.878	0.904	0.848
Redbone	0.617	0.546	0.486	0.507
Japanese Spaniel	0.886	0.919	0.939	0.837
Blenheim Spaniel	0.980	0.966	0.944	0.906
Afghan Hound	0.955	0.946	0.947	0.953
Bluetrick	0.897	0.892	0.864	0.870
Borzoi	0.931	0.922	0.932	0.925
Maltese dog	0.905	0.901	0.874	0.784
Papillon	0.971	0.984	0.973	0.952
Basset	0.763	0.779	0.707	0.711
Coonhound	0.863	0.864	0.895	0.901
Irish Wolfhound	0.945	0.930	0.940	0.905
Pekinese	0.723	0.840	0.743	0.782
Toy Terrier	0.838	0.856	0.784	0.804
Beagle	0.725	0.743	0.665	0.669
Walker Hound	0.767	0.705	0.757	0.733
Italian Greyhound	0.822	0.845	0.800	0.785
mAP	0.840	0.848	0.829	0.807

## 4.5 Zwischenstand

Um herauszufinden wodurch die abnehmende Genauigkeit in der Klassifizierung entstanden ist, wurden die Trainingsdaten überprüft. Dabei ist aufgefallen das bei der Histogramm Ausgleichung und der Spezifikation große Unterschiede in den Datensätzen zu erkennen sind. Das könnte daran liegen, dass die Normalisierung die Farbinformationen des gesamten Bildes nimmt und diese anpasst. Dabei spielt es einen Unterschied, ob die Umgebung in dem Bild Hell, dunkel oder eine andere dominante Farbe hat. Bei der Histogramm Spezifikation kommt zusätzlich noch das Referenzbild hinzu welches für den Datensatz verwendet wird. Bei der Verwendeten Art von Datensätzen, konnte keine einheitliches Referenzbild genutzt werden, da fast jedes Bild unter anderen Bedingungen entstanden ist. Durch diese unterschiedlichen Trainingsbildern kommt es zu Anomalien in den Normalisierten Bildern (Abbildung 17). Um diese Vermutung zu bestärken, wurden Objekte mit gleicher Lichteinstrahlung und gleicher Entfernung auf unterschiedlichen Hintergründen aufgenommen (Abbildung16). Darauf hin, wurden die Bilder jeweils mit allen Verfahren Normalisiert. Das Objekt auf dem Bild, wurde vom Hintergrund segmentiert, um zu überprüfen, wie sich die Histogramme des Objektes voneinander unterscheiden. (Abbildung17).

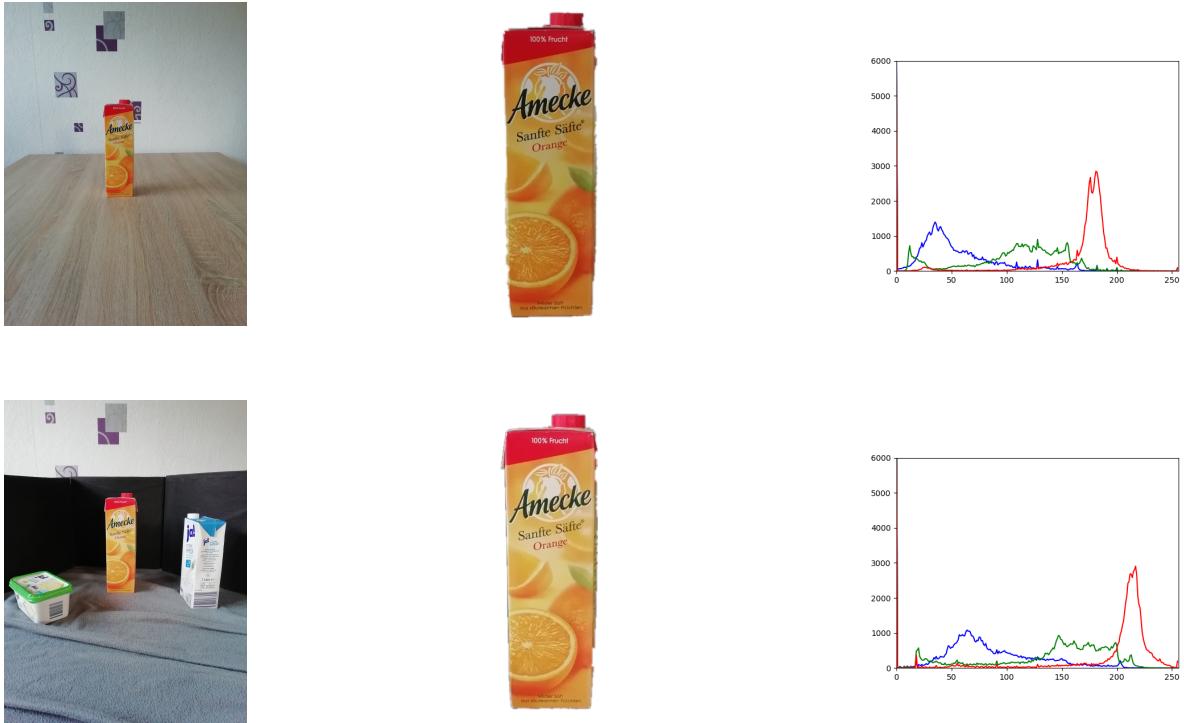


Abbildung 16: *Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Ohne Normalisierung*

Durch diesen Test wurde bestätigt, das die Ergebnisse der Histogramm Ausgleichung und Spezifikation sehr unterschiedlich geworden sind. Die Veränderung des Umfeldes beeinflusst also das Ergebnis entscheidend und sorgt für größere Farbveränderungen als der Lichteinfluss. Ein Datensatz mit einheitlichem Hintergrund in den Trainingsdaten, könnte mit den normalisierungs-Verfahren besser funktionieren. Um diese Annahme weiter überprüfen zu können, soll ein weiterer Datensatz generiert werden, in welchem ein konstanter Hintergrund festgelegt ist.

## 4.6 Obst Datensatz

Für einen weiteren Versuch, wurde der Obstdatensatz erstellt. Dieser besteht aus den Objekten welche in Tabelle 14 aufgelistet sind. Bei diesem Datensatz wurde darauf geachtet einen einheitlichen Hintergrund zu verwenden, um die Funktionalität bei anderen Einstellungen zu testen. Außerdem wurde im Vergleich zum Nahrungsmittel Datensatz (Tabelle.8) darauf geachtet, ähnlichere Kassen zu verwenden, auf welchen optimaler Weise keine Schrift enthalten ist. Das Netz soll nicht durch die Schrift beeinflusst werden, sondern hauptsächlich die Farben als Orientierung nehmen. Dieser Datensatz wird nun genau wie die vorherigen Datensätze, mit den unterschiedlichen Methoden Normalisiert. Dabei soll zum einen die Genauigkeit der einzelnen Klassen betrachtet werden, zum anderen die

Tabelle 14: *Klassen des Obst Datensatzes mit konstantem Hintergrund*



Klassename	Klassename
Orange	Mango
Banane	Nektarine
Apfel	Birne

Trainingsdaten an sich. Dabei soll überprüft werden, ob die Trainingsbilder wie erwartet aussehen und wie stark Anomalien auftreten.

## 4.7 Auswertung des Obst Datensatzes

Nach Auswertung der Trainingsergebnisse kann festgestellt werden, dass die Normalisierungsmethoden keinen Negativen Einfluss auf den Datensatz haben. Da die Genauigkeit bei 100% der Testdaten lagen, kann von den Zahlen aus, leider keine Aussage getroffen werden, inwieweit die Methoden einen positiven Einfluss auf die Klassifizierung haben. Eine weitere Möglichkeit dies zu überprüfen, könnte mit Hilfe besonders schwierigen Testbildern überprüft werden. Dafür sollen Bildaufnahmen bei sehr unterschiedlichen Lichteinflüssen verwendet werden.

Tabelle 15: *Genauigkeits-Berechnungen des Modells des Obst Datensatzes*

Klassename	AP(N)	AP(GW)	AP(HA)	AP(HS)
Orange	1.000	1.000	1.000	1.000
Nektarine	1.000	1.000	1.000	1.000
Banane	1.000	1.000	1.000	1.000
Mango	1.000	1.000	1.000	1.000
Apfel	1.000	1.000	0.999	1.000
Birne	1.000	1.000	1.000	1.000
mAP	1.000	1.000	1.000	1.000

Die Auswertung der normalisierten Trainingsdaten hat ergeben, dass wesentlich weniger farb-Anomalien auftreten. Dabei ist aufgefallen, dass gerade bei den Histogramm Normalisierungen mit einheitlichen Hintergrund deutliche Verbesserungen erzielt wurden. Zu großen Veränderungen treten nur dann auf, wenn viele Objekte auf dem Bild enthalten sind, da diese die Umgebung zu stark beeinflussen oder wenn die Beleuchtungen zu extrem verändert werden. Die Histogramm Spezifikation geht mit diesem Problem noch am besten um, weil dieser ein Referenz-Histogramm hat, um es anzupassen.

Auch eine Betrachtung der Helligkeitsverteilung der Trainingsdaten werden wesentlich besser wie in den Beispielen in Abbildung 20 zu erkennen ist. Dafür wurde eine stark unterbelichtete Aufnahme generiert, um die Veränderung nach den Normalisierungs Methoden besser herauszustellen. Die beste Helligkeitsverteilung konnte dabei der Histogramm Ausgleich aufweisen, wobei durch das dunkle Quellbild die Farbinformationen zum großen Teil verloren gegangen sind. Der Gray World hat die Helligkeitsverteilung kaum verändert, was aber daran liegt, dass die Farbverteilung durch die Höhe des Grün Kanals verschoben wird und nicht für solche Aufgaben ausgelegt ist. Das insgesamt beste Ergebnis liefert die Histogramm Spezifikation. Nicht nur die Helligkeitsverteilung konnte verbessert werden, auch die Farben konnten vergleichsweise gut wieder hergestellt werden.

## 5 Diskussion

In diesem Kapitel sollen die Ergebnisse und Erkenntnisse welche im Verlauf der Arbeit herausgearbeitet wurden kurz aufgeführt und erläutert werden, damit am Ende ein Gesamtfazit gezogen werden kann.

Die Daten der trainierten Netze zeigen, dass mit den verwendeten Datensätzen und den angewandten Normalisierungsmethoden keine Erhöhung der Genauigkeit erzielt werden konnte. Ein möglicher Grund waren die Datensätze selber, welche normalisiert wurden. Da manche Normalisierungs-Verfahren die Farbinformationen des gesamten Bildes verwendeten, ist das Ergebnis je nach Umgebung unterschiedlich ausgefallen. Daraufhin wurden Testaufnahmen von Objekten auf verschiedenen Hintergründen aufgenommen, welche aber gleiche Belichtung und Aufnahmewinkel hatten. In den unveränderten Testbildern ist kein großer Unterschied in den Histogrammen aufgefallen. Bei gleicher Lichteinstrahlung und unterschiedlichen Hintergründen, haben die Objekte eine fast identische Farbverteilung. Lediglich die Helligkeit hat etwas variiert. Der normalisierte Datensatz jedoch, zeigt größere Unterschiede in den Histogrammen und den Farben auf. Das könnte daran liegen, dass die Normalisierung nicht nur die Objekte auf den Bildern normalisiert, sondern auch den Hintergrund, welcher bei vielen der Trainingsbildern unterschiedlich ist.

Die Überprüfung hat gezeigt, dass die Datensätze für den verwendeten Aufbau und den verwendeten Normalisierungs Algorithmen nicht geeignet waren. Bessere Ergebnisse konnten mit dem zusätzlich generierten Obst Datensatz erzielt werden, bei dem alle Trainingsdaten auf dem selben Hintergrund aufgenommen wurden. Dadurch konnten alle Klassen auf der selben Basis normalisiert werden. Gerade die Histogramm Spezifikation (Abbildung 20) hat hier besonders gute Ergebnisse erzielt, da der Einsatz eines Referenzbildes besser funktioniert, und wesentlich weniger Anomalien erzeugt wurden. Auch bei Untersuchung der Helligkeit, konnten wesentliche Verbesserungen erzielt werden. Bei einem stark unterbelichteten Bild wurde durch den Histogramm Ausgleich die Lichtverteilung wesentlich angehoben. Die Farbinformationen konnten jedoch größtenteils nicht wiederhergestellt werden. Durch die Histogramm Spezifikation hat sich auch die Lichtverteilung verbessert und durch das Referenzbild wurden auch die Farbinformationen wiederhergestellt. Der Gray-World-Algorithmus schneidet hier am schlechtesten ab da dieser nicht mit den einzelnen Farbwerten arbeitet, sondern die gesamten Farbkanäle auf Basis des grün-Kanals verschiebt.

### 5.1 Laufzeittest

Ein weiterer Einfluss welches die Normalisierungs Algorithmen auf die Klassifizierung von neuronalen Netzen haben, ist neben dem Manipulieren von Trainingsdaten, die Zeit welches die jeweiligen Verfahren benötigen. Um einen Eindruck zu bekommen, wurde Ein

Bild in verschiedenen Größen getestet, um herauszufinden wie sich die Laufzeit bei größer werdenden Bilddateien verhält. Der Gray World Algorithmus ist im Vergleich zu den

*Tabelle 16: Laufzeiten der  
Normalisierungs-Algorithmen mit verschiedenen großen  
Bildern*

Bildgröße	Faktor	Dateigröße	GW	HA	HS
4160x3120	100%	2.441 KB	1,447s	0,172s	6,952s
2912x2184	70%	1.000 KB	0,755s	0,078s	3,215s
2080x1560	50%	529 KB	0,362s	0,035s	1,717s
1040x780	25%	114 KB	0,094s	0,013s	0,422s

anderen Methoden in der Mitte von der Laufzeit bei Verdoppelung der Bildgröße, erhöht sich die Laufzeit um das vierfache. Dabei ist auch die Farbgebung und der Kontrast nicht entscheidend. Die Histogramm Ausgleichung ist in diesem Fall die Schnellste Methode, wobei dafür die Verteilung im Histogramm eine Rolle spielt. Die Laufzeit verändert sich nicht Linear, sondern steigt exponential an. So steigt die Laufzeit beim verdoppeln um den Faktor drei, und beim nächsten mal um den Faktor sechs. Der längste Normalisierungs Algorithmus ist in diesem Fall die Histogramm Spezifikation. Dabei spielt die Größe des Referenzbildes eine wichtige Rolle. Mit einem größeren Referenzbild, würde sich die Laufzeit noch weiter erhöhen. Bei gleich bleibenden Referenzbild erhöht sich die Laufzeit um den Faktor vier. Durch die hohe Grunddauer führt das schnell zu langen Laufzeiten.

Dabei muss beachtet werden, das die Laufzeiten nur für ein Bild berechnet wurde. Für ein künstliches neuronales Netz werden meist mehrere Tausend Bilder verwendet. Hier muss, je nach Datensatz, abgewägt werden, ob eine Normalisierung für die erhöhte Laufzeit in Kauf genommen werden soll.

## 6 Fazit

# Literatur

- [Aih12] Shunsuke Aihara. *color correction algorithm in python*. 2012. URL: <https://bit.ly/2LwjBY7>.
- [Ang18] Bistra Angelova. *Optical character recognition für chinesische Schrift*. 2018. URL: <https://bit.ly/2VfsQOL>.
- [BB09] Wilhelm Burger und Mark James Burge. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag, 2009.
- [Con18] Common Objects in Context. *COCO Datenset*. 2018. URL: <http://cocodataset.org/#home>.
- [CSD18] University of California San Diego. *Why are neuron axons long and spindly? Study shows they're optimizing signaling efficiency*. 2018. URL: <https://bit.ly/2z0ekEL>.
- [Dee16] DeepMind. *Google DeepMind Challenge Match: Lee Sedol vs AlphaGo*. 2016. URL: <https://bit.ly/1M7Y9io>.
- [Ert13] Wolfgang Ertel. *Grundkurs künstliche Intelligenz: eine praxisorientierte Einführung*. Springer-Verlag, 2013.
- [Eve+] M. Everingham u. a. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results*. <https://bit.ly/2Yy6tFG>.
- [Goo+16] Ian Goodfellow u. a. *Deep learning*. Bd. 1. MIT press Cambridge, 2016.
- [HE16] Joachim Hoffmann und Johannes Engelkamp. *Lern- und Gedächtnispsychologie*. Springer-Verlag, 2016.
- [Jaz16] Mohammad Al Jazaery. *OpenCV Python equalizeHist colored image*. 2016. URL: <https://bit.ly/3270Z6X>.
- [Kho+11] Aditya Khosla u. a. „Novel Dataset for Fine-Grained Image Categorization“. In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO, 2011.
- [LLC18] Google LLC. *Tensorflow detection model zoo*. 2018. URL: <https://bit.ly/2EtFjJP>.
- [Mis19] Mayank Mishra. *Convolutional Neural Networks, Explained*. 2019. URL: <https://bit.ly/2FMuRMg>.
- [Pen+16] Min Peng u. a. „NIRFaceNet: A Convolutional Neural Network for Near-Infrared Face Identification“. In: *Information* 7.4 (2016). ISSN: 2078-2489. DOI: [10.3390/info7040061](https://doi.org/10.3390/info7040061). URL: <https://bit.ly/2GGNqUP>.

- [SCL12] Pierre Sermanet, Soumith Chintala und Yann LeCun. „Convolutional neural networks applied to house numbers digit classification“. In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, S. 3288–3291.
- [ST13] Robert F Schmidt und Gerhard Thews. *Physiologie des Menschen*. Springer-Verlag, 2013.
- [tzu19] darrenl tzutalin. *labelimg*. 2019. URL: <https://bit.ly/2uvF2jj>.
- [Wik18] Computer science Wiki. *Max-pooling / Pooling*. 2018. URL: <https://bit.ly/20zVjdB>.
- [Wik19] Wikipedia. *Künstliches Neuron*. 2019. URL: <https://bit.ly/2Xmy6oj>.

# Anhang

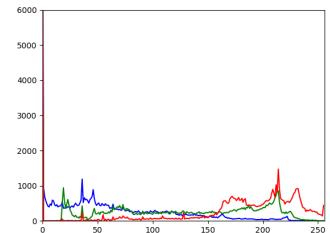
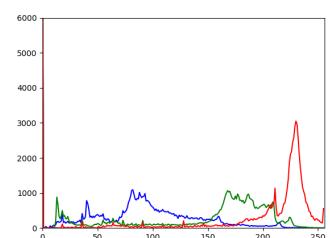
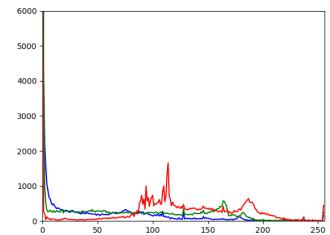


Abbildung 17: Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch den Histogramm Ausgleich

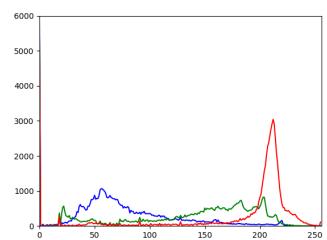
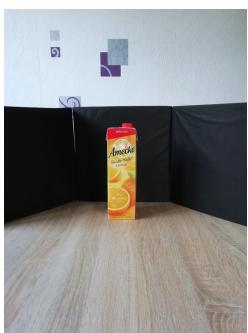
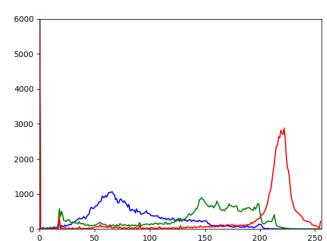
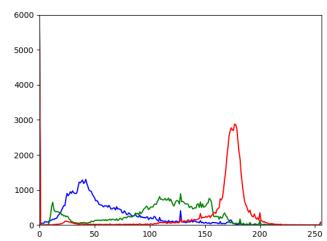


Abbildung 18: *Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch den Gray-World Algorithmus*

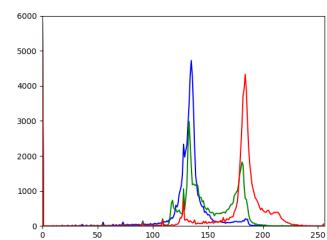
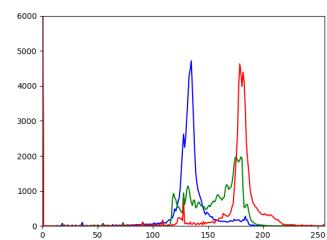
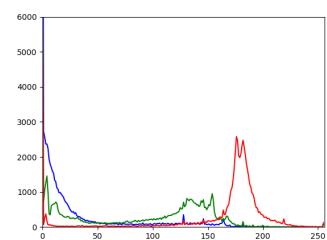
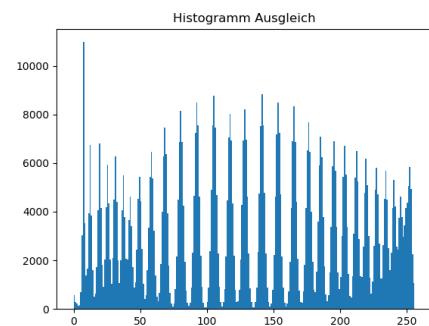
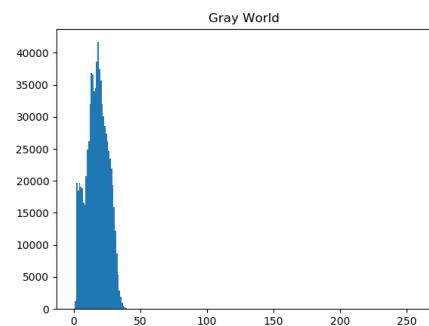
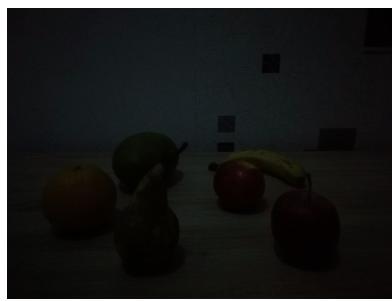
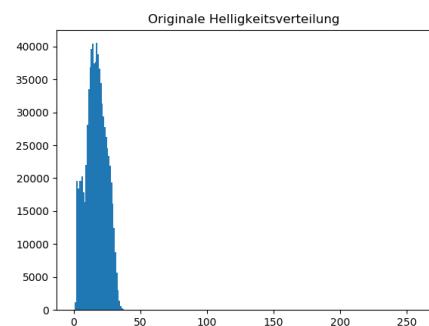
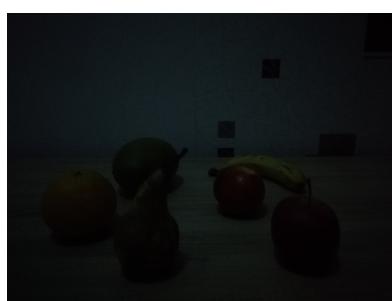
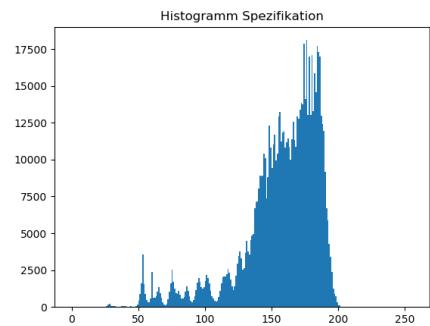


Abbildung 19: *Histogramm des Segmentierten Objektes aus dem Nahrungsmittel Datensatz. Normalisiert durch die Histogramm Spezifikation*

Abbildung 20: Helligkeitsverteilung und den Einfluss  
der Normalisierungs Algorithmen





## Eidesstattliche Erklärung

Ich versichere, dass ich die Bachelorarbeit selbstständig angefertigt und keine anderen als die von mir angegebenen und bei Zitaten kenntlich gemachten Quellen und Hilfsmittel benutzt und die vorliegende Arbeit an keiner anderen Stelle zur Erlangung eines Abschlusses vorgelegt habe.

---

Datum, Ort

---

Unterschrift