

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

BACHELORARBEIT

ausgearbeitet von

Torben Krause

vorgelegt an der

TECHNISCHEN HOCHSCHULE KÖLN
CAMPUS GUMMERSBACH
FAKULTÄT FÜR INFORMATIK UND
INGENIEURWISSENSCHAFTEN

im Studiengang

MEDIENINFORMATIK

Erster Prüfer: Prof. Dr. Martin Eisemann
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Matthias Böhmer
Technische Hochschule Köln

Gummersbach, im Juli 2019



Adressen: Torben Krause
An der Wende 8
51643 Gummersbach
torben.krause@smail.th-koeln.de

Prof. Dr. Martin Eisemann
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
martin.eisemann@th-koeln.de

Prof. Dr. Matthias Böhmer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
matthias.boehmer@th-koeln.de

Abstract

Bei der Klassifizierung von künstlichen neuronalen Netzen kann es passieren, dass es zu Fehlern bei der Identifizierung von Objekten kommt. Das liegt häufig daran, dass nicht jede Farbe trainiert wurde, welche bei unterschiedlichen Beleuchtungen auftreten kann.

Das Ziel dieser Arbeit ist es, zu überprüfen, ob Farbnormalisierung von Datensätzen die Klassifizierung mittels künstlicher neuronaler Netze positiv beeinflusst, und welche Beschränkungen diese hat.

Um dieser Frage nachzugehen, werden zunächst die theoretischen Grundlagen zum digitalen Bild und der neuronalen Netze erläutert. Anschließend werden verschiedene Verfahren zur Farbnormalisierung vorgestellt und auf ihre Eignung, zur Lösung der gegebenen Problemstellung hin, untersucht. Es erfolgt die Darstellung der Implementierung der Software unter Verwendung des OpenSource Frameworks Tensorflow. Nachfolgend werden vier verschiedene Datensätze mit den unterschiedlichen Normalisierungsverfahren bearbeitet. Im Anschluss kommt es zur Durchführung des Trainings der verschiedenen Netze, wobei je ein Netz pro Normalisierungsverfahren trainiert wird. Diese Netze werden daraufhin verglichen und ausgewertet. In der Diskussion werden die Ergebnisse evaluiert und ein Fazit der Arbeit gezogen.

Die Ergebnisse des Trainings bestärken, dass die Farbnormalisierung positive Einflüsse auf die Datensätze haben kann. Voraussetzung dafür ist, dass die Datensätze an die Kriterien der Normalisierungverfahren angepasst werden.

Inhaltsverzeichnis

Vorwort	1
1 Einleitung	2
1.1 Problemstellung und Ziele	2
1.2 Erläuterung der These	3
1.3 Anforderungen	3
1.4 Struktur der Arbeit	4
2 Theoretische Grundlagen	6
2.1 Digitale Bilder	6
2.1.1 Farträume digitaler Bilder	7
2.1.2 Histogramme	8
2.1.3 Einfluss von Belichtungen	10
2.2 Künstliche Intelligenz	11
2.3 Neuronale Netze	12
2.3.1 Entstehung von Trainingsdaten	20
2.3.2 Mean average Precision	21
2.4 Arten der Farbnormalisierung	22
2.4.1 Ansatz des Gray-World-Algorithmus	22
2.4.2 Ansatz der Histogramm-Ausgleichung	23
2.4.3 Ansatz der Histogramm-Spezifikation	25
3 Methodik und Durchführung	28
3.1 Vortrainierte Netze	28
3.2 Trainingsdaten	28
3.3 Normalisierungs-Algorithmen	30
3.3.1 Gray-World-Algorithmus	31
3.3.2 Histogramm-Spezifikation	33
4 Ergebnisse	34
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	39
4.7 Auswertung des Obst-Datensatzes	39
5 Diskussion	41
5.1 Datensätze und Netze	41

5.2 Laufzeittest	42
6 Fazit	43
Abbildungsverzeichnis	47
Tabellenverzeichnis	48
Literaturverzeichnis	50
ANHANG	

Vorwort

Vor Ihnen liegt die Bachelorarbeit *Der Einfluss von Farbnormalisierung auf die Klassifizierung von Bildern durch künstliche neuronale Netze*, welche an der Technischen Hochschule Köln erstellt wurde.

Diese Bachelorarbeit habe ich zum Abschluss meines Studiums der Medieninformatik an der Technischen Hochschule Köln eigenständig verfasst. Ziel war es, herauszufinden, welchen Einfluss die Farbnormalisierung auf die Klassifizierung von Bildern, mittels künstlicher neuronaler Netze, hat. Von Mai 2019 bis Juli 2019 beschäftigte ich mich intensiv mit den einzelnen Themenbereichen und dem Schreiben der Bachelorarbeit. Zusammen mit meinem Professor, Dr. Martin Eisemann, habe ich die Fragestellung für diese Bachelorarbeit entwickelt. Die Forschung für meine Arbeit sowie auch die Durchführung waren nicht immer einfach und haben stellenweise zu Problemen geführt. Nichtsdestotrotz gelang es mir, nach einer umfangreichen Analyse der Ergebnisse, die Forschungsfrage zu beantworten. Während dieser Untersuchung waren meine Betreuer, Uwe Müsse und Dr. Benjamin Meyer, stets an meiner Seite. Sie beantworteten meine Fragen und gaben wertvollen Input für die methodische Vorgehensweise, sodass ich meine Forschung erfolgreich fortführen konnte.

Aus diesem Grund möchte ich meinem Professor und meinen Betreuern für die Unterstützung und ihre gute Anleitung während dieses Prozesses danken. Sie haben mir viele Blickwinkel gezeigt, Probleme zu betrachten. Außerdem möchte ich meiner Schwester Nicole Teismann und meinem Schwager Michael Teismann danken, dass sie immer ein offenes Ohr hatten und mir bei der Struktur sowie Korrektur geholfen haben. Zu guter Letzt möchte ich meinem Kommilitonen Binh Ngo danken, der parallel mit mir seine Bachelorarbeit geschrieben hat und immer hilfsbereit war.

Ich wünsche Ihnen viel Freude beim Lesen dieser Bachelorarbeit.

Torben Krause

Gummersbach, 17. Juli 2019

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 in anderen Bereichen 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 und die Qualität der Trainingsdaten dar und zum anderen die Zeit, in welcher das neuronale Netz trainiert wird. Da normalerweise viel Zeit für das Training benötigt wird, kann das Verfahren mithilfe von steigender Rechenleistung beschleunigt werden. In der Vergangenheit konnten die Forschungen aufgrund zu langsamer Hardware nicht effektiv weiter geführt werden. Mit der Leistung heutiger CPUs und GPUs ist dies mit wesentlich weniger Problemen verbunden.

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. Trainingsdaten bestehen üblicherweise aus mehreren Hunderttausend Bildern. Bei der Verwendung von vortrainierten Netzen reichen um die Tausend Bilder aus, um neue Klassen zu trainieren. Die richtige Ausleuchtung ist ein wichtiger Aspekt bei der Generierung von Trainingsbildern, 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 1.1 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, dass dieses Problem hauptsächlich in der Nachbereitung der Bilder angegangen werden kann.



Abbildung 1.1: Szene mit unterschiedlicher Ausleuchtung

1.2 Erläuterung der These

Für solche Probleme bietet die Bildverarbeitung einige Lösungsansätze, welche theoretisch eine konstante Farbgebung und dadurch eine Erhöhung der Genauigkeit bringen könnten. Ob in der praktischen Ausführung die erwarteten Ergebnisse erzielt werden können, soll getestet werden. In dieser Arbeit sollen verschiedene Verfahren zur Farbnormalisierung von Bildern auf die Trainings- und Testdatensätze angewendet werden. Mit diesen normalisierten Datensätzen sollen daraufhin 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 Normalisierungsverfahren aufweisen.

1.3 Anforderungen

Bei dem geplanten Vorhaben, welches diese Arbeit thematisiert, ist es wichtig Anforderungen an das neuronale Netz, den Datensatz und die verwendeten Algorithmen zu formulieren, um sicher zu stellen, dass keine vermeidbaren Probleme auftreten. Zunächst werden Trainingsdaten benötigt, mit denen 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 in Anbetracht der verfügbaren Zeit nicht für drei verschiedene Datensätze generiert werden, weswegen eine alternative Lösung gefunden werden muss. Für gute Vergleichswerte werden die Normalierungsmethoden auf mehreren Datensätzen angewendet. Auch die Datensätze, welche verwendet werden, sollen durch einige Anforderungen geprüft und ausgewählt

werden.

Daraus ergeben sich folgende Anforderungen:

1. Die Menge an benötigten Trainingsdaten pro Klasse soll gering gehalten werden.
2. Die Normalisierungsverfahren sollen auf verschiedene Datensätze angewendet werden.
3. Die Datensätze sollten nicht unordentlich sein, da das Reinigen 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 dadurch Daten entfernt oder hinzugefügt werden.
4. Die Datensätze sollten Bilder enthalten, welche eine nicht zu hohe Auflösung besitzen, da sie sonst den Trainingsfortschritt zurückhalten und mehr Ressourcen benötigen. Je größer das Bild ist, desto mehr Zeit wird für die Verarbeitung benötigt.
5. Beim Generieren der Trainingsbilder sollte darauf geachtet werden, saubere und gut ausgeleuchtete Aufnahmen zu machen. Aufnahmen der Objekte sollten nicht zu stark verschwommen und unscharf sein.
6. Das Ziel der Datensätze sollte genau definiert werden. Namen und Thema des Datensatzes sollten auf einen gewissen Bereich beschränkt werden. Beispielsweise sollten keine Gesichter zusammen mit Pflanzen kombiniert werden.

1.4 Struktur der Arbeit

Für ein besseres Verständnis der einzelnen Entwicklungsschritte in den späteren Kapiteln, werden in Abschnitt 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 Abschnitt 2.3 auf die Funktionsweise künstlicher neuronaler Netze eingegangen. Hierbei werden die unterschiedlichen Schichten, welche durchlaufen werden, aufgeführt und beschrieben. Anschließend werden in Abschnitt 3.3 die verwendeten Farbnormalisierungsverfahren erklärt und beschrieben. Der Ansatz der einzelnen Methoden wird erläutert, sowie auch 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 Farbnormalisierungsverfahren zusätzlich in der technischen Umsetzung erläutert. Daraufhin werden die Ergebnisse aufgeführt 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 gezogen.

2 Theoretische Grundlagen

Das folgende Kapitel soll eine kurze und leicht verständliche Einführung sein, in der die theoretischen Grundlagen zu dem Aufbau von digitalen Bildern, dem Einfluss von Belichtung sowie den verwendeten Bildbearbeitungsmethoden gelegt werden. Anschließend folgt ein Überblick über die Grundlagen der künstlichen Intelligenz. Der letzte Abschnitt widmet 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 dieser Arbeit nachvollziehbar wird, soll in diesem Abschnitt 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. Die digitale Repräsentation eines aufgenommenen Bildes erfolgt als zweidimensionale Matrix, wie in Abbildung 2.1 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 8-Bit-Graustufenbild sind das normalerweise 256 Abstufungen.

In diesen Pixeln werden Bildinformationen, wie Farbe und Position festgehalten. Jeder Pixel besteht dabei oft aus drei Kanälen, dem R Kanal für Rot, dem G Kanal für Grün und dem 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 andere Farbräume [up19] (RGB, HSV, CMY(K), YUV, etc.).

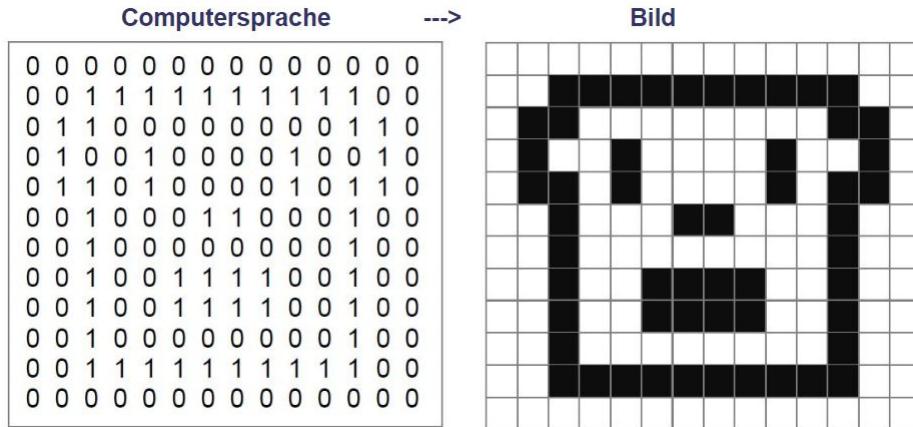


Abbildung 2.1: Aufbau eines Digitalen Bildes mit zwei Helligkeitsabstufungen

2.1.1 Farträume digitaler Bilder

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 quadratische, wie auch rechteckige Abmessungen haben. Jeder Punkt in der Matrix stellt einen Pixel dar und enthält jeweils einen Rot-, Grün- und Blauwert, welche zusammen eine beliebige Farbe darstellen. 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. Einfachheitshalber werden Graustufenbilder 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 Farträumen (Rot-Grün-Blau). Er beschreibt, welche Art von Licht ausgestrahlt werden muss, um eine gewünschte Farbe zu erreichen. Die Kombination der einzelnen Farträume ist eine additive Farbmischung. RGB ist eher ein Farbmodell, da es viele Farbräume gibt, welche sich davon ableiten lassen (sRGB, AdobeRGB, etc.).

CMY(K)

Anders als der RGB-Farbraum, benutzt der CMY-Farbraum die subtraktive anstelle der additiven Farbmischung. Hierbei sind die drei Grundfarben C für Cyan, M für

Magenta und Y für Yellow. Bekannt ist diese Farbmischung durch das Malen von Wasserfarben oder für die Farbpigmente, welche in Druckern genutzt werden. Durch die subtraktive Farbmischung kann der CMY kein richtiges Schwarz erreichen. Damit auch Schwarz richtig dargestellt werden kann und kein dunkles Grau entsteht, wurde der Farbraum CMY durch den Parameter K, für Schwarz, erweitert.

HSV

Der HSV-Farbraum ist anders als der RGB-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 Helligkeitswert oder auch die Helligkeit der Farbe. Dieser Farbraum ist gerade in der Kunst beliebt, da die Farbvorstellung besser funktioniert, als bei additiven und subtraktiven Farbmischungen, denn beim Malen wird mit Farben gearbeitet, bei denen lediglich die Sättigung sowie die Helligkeit verändert werden.

YUV

Das YUV-Farbmodell wird hauptsächlich beim analogen Fernsehen verwendet. Dieser Farbraum verwendet zur Darstellung von Farben zwei Komponenten. Die Lichtstärke, auch Luminanz (Y) genannt, sowie die Chrominanz (Farbanteil), welche wiederum aus den zwei Unterkomponenten U und V besteht. U und V geben somit die Farbe an. Hierfür werden U und V in ein vierteiliges Koordinatensystem gelegt. Auf der X-Achse liegt V und auf der Y-Achse U. Jedes Feld des Koordinatensystems besitzt hierbei eine Farbe (Rot, Violett, Blau und Grün). Die Farbe wird durch die Kombination beider Werte (U und V) bestimmt. Y stellt das Helligkeitssignal in dem Farbraum dar und gibt an, wie hell die Farbe dargestellt werden soll. Anders als beim HSV-Farbraum werden Lichtstärke und Sättigung der Farbe durch einen Kanal bestimmt.

2.1.2 Histogramme

Histogramme beschreiben eine Häufigkeitsverteilung [BB09, 42ff.]. Speziell bei Bildern zeigen Histogramme die Häufigkeit der auftretenden Intensitätswerte. Am einfachsten lässt sich das mit Graustufenbildern nachvollziehen. Ein Beispiel dafür zeigt die Abbildung 2.2. Für ein Graustufenbild I mit möglichen Intensitätswerten 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 $h(i) = \text{card}(u, v) | I(u, v) = i$. Wobei card die Anzahl der Elemente einer Menge be-

schreibt (*Kardinalität*).

Die Abbildung 2.2 zeigt ein mögliches Histogramm mit einer Farbabstufung von 16

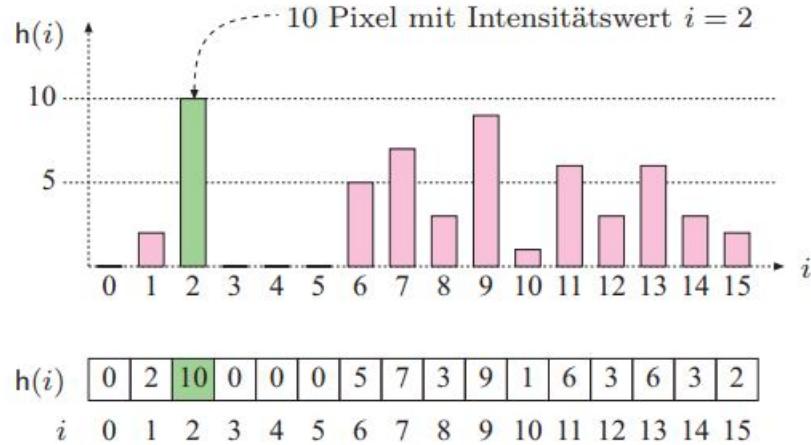


Abbildung 2.2: Histogramm eines Graustufen-Bildes mit 16 Helligkeitsabstufungen [BB09, 42]

Werten. Ein Histogramm zeigt die wichtigsten Eigenschaften eines Bildes, wie beispielsweise die Dynamik oder den Kontrast. Außerdem kann festgestellt werden, ob bei der Bildaufnahme Probleme aufgetreten sind. Diese Probleme lassen sich an ganz linken oder ganz rechten Ausschlägen des Graphen erkennen. Ein Ausschlag bei 0 oder 255 bedeutet, dass dort keine Farbinformationen mehr vorhanden sind und 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 Pixelwerte [BB09, 44]. Ein Bild, welches einen hohen Kontrast hat, nutzt das gesamte Spektrum des Histogramms (Schwarz bis Weiß). Der Kontrast lässt sich also leicht aus einem Histogramm entnehmen, wie die verschiedenen Histogramme in Abbildung 2.4 zeigen.

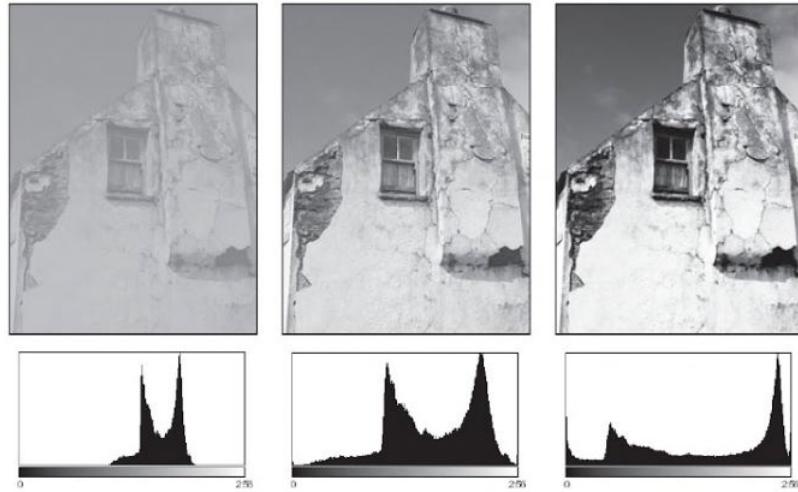


Abbildung 2.3: Unterschiedlicher Kontrast und die Auswirkungen im Histogramm: niedriger Kontrast (links), normaler Kontrast (Mitte), hoher Kontrast (rechts)[BB09, 45]

Dynamik

Dynamik beschreibt den Quotienten aus dem größten und dem kleinsten Intensitätswerten in einem Bild und deren jeweilige Anzahl [BB09, 44]. Im besten Fall entspricht die Dynamik der insgesamt genutzten Anzahl an Pixelwerten (Abbildung 2.4). 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 einem möglichen Qualitätsverlust vorzubeugen.

2.1.3 Einfluss von Belichtungen

In dem Unterabschnitt 2.1.1 wurde beschrieben, wie Farben in Bildern gespeichert werden und welche unterschiedlichen Arten von Farbräumen es gibt [BB09, 41ff.]. Bei der Aufnahme von Bildern ist gerade die Belichtung entscheidend. Eine hohe Belichtung sorgt dafür, dass Farben heller wirken, wobei wenig Licht dafür sorgt, dass 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

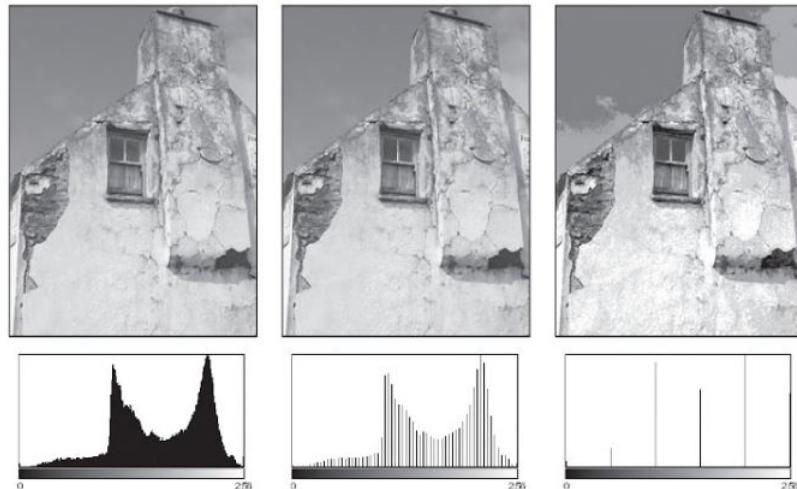


Abbildung 2.4: Unterschiedliche Dynamik und die Auswirkungen im Histogramm: hohe Dynamik (links), niedrige Dynamik mit 64 Graustufen (Mitte), extrem niedriger Kontrast mit 6 Graustufen (rechts)[BB09, 45]

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. Ziel der Farbnormalisierung ist es nicht die Originalfarbe des Objektes wieder herzustellen, sondern für eine konstante Farbvariation zu sorgen. Die für diese Arbeit ausgewählten Verfahren werden im Abschnitt 2.4 aufgeführt und beschrieben.

2.2 Künstliche Intelligenz

In den letzten Jahren hat die Nutzung von künstlichen Intelligenzen enorm zugenommen. Viele Firmen nutzen diese Technologie beispielsweise für die Analyse und Vorher sage von Kundenverhalten, für die Klassifikation von Bildern oder für die Spracherkennung in heutigen Smartphones. Mit künstlicher Intelligenz beschreibt man Computer, welche gewisse Aufgaben erledigen, ohne ausdrücklich dafür programmiert worden zu sein. Dafür werden dem System Informationen anhand von Vergleichsdaten zugeführt, wodurch es sich Eigenschaften und Strukturen merkt und durch diese Rückschlüsse und Prognosen ziehen kann. Eine gute Beschreibung liefern dabei die Autoren Rich und Knight, indem sie schrieben:

Das Studium des Problems, Computer dazu zu bringen, Dinge zu tun, bei denen ihnen momentan der Mensch noch überlegen ist. - (Rich und Knight 1991) [Küp17, 231]

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, seine Spielzüge besser als der Weltmeister planen [Ert13, 331 ff.]. Anhand der Definition von Rich kann das *AlphaGo* im Bereich des Spieles Go als intelligent bezeichnet werden, da es dem besten Go-Spieler der Welt deutlich überlegen war.

2.3 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 aufgrund von individuellen Erfahrungen [HE16]. Der Lernprozess kann sowohl bewusst sowie auch unbewusst unser Verhalten verändern. Menschen adaptieren also sowohl 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 demnach die Funktionsweise des menschlichen Gehirns zum Vorbild [Ert13, 1].

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 demzufolge viele Neuronen nötig. In der Abbildung 2.5 kann man den wesentlichen Aufbau eines Neurons sehen.

In diesem vereinfachten Modell eines Neurons lassen sich dessen wesentliche 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. Sie sendet elektrische Impulse an die Soma weiter, welche dort verarbeitet werden. Wenn die Spannung 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

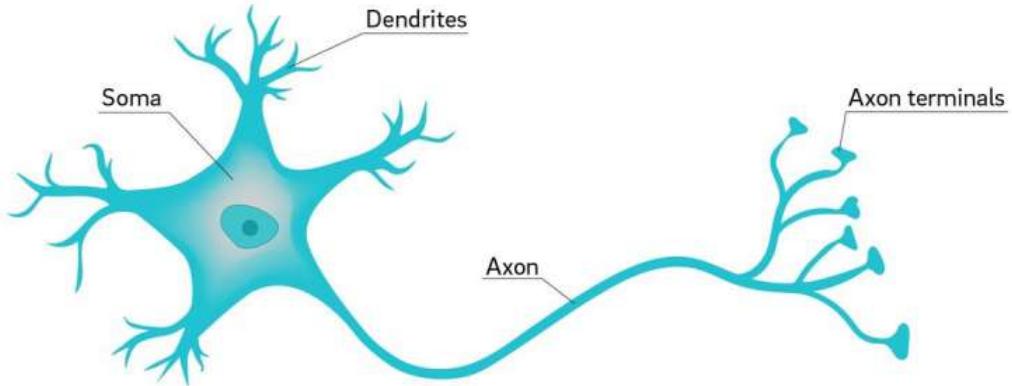


Abbildung 2.5: Aufbau eines Neurons [oCSD18]

Neuron unterschiedlich und sie 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 Verständnis dienen.

Perzeptron

Die Grundlage für den mathematischen Ablauf künstlicher neuronaler Netze wurde in dem Buch von McCulloch und Pitts *A logical calculus of the ideas immanent in nervous activity* erklärt [MP43]. Auf diesen Grundlagen simulierte 1953 der Psychologe und Informatiker Frank Rosenblatt das erste künstliche Neuron, welches auch als Perzepron bekannt wurde. In der Abbildung 2.6 wird der Aufbau eines solchen Perzeptrons dargestellt und Formell beschrieben [Ert13, 269].

Als Eingabewert bekommt das Perzepron ein Gewicht w , welches mit den Eingangs-wert x multipliziert wird. Das Ergebnis dieser beiden Werte, kann als lineare Funktion beschrieben werden.

$$f(x) = w * x \quad (2.1)$$

Anschließend wird das Ergebnis der beiden Werte an eine Aktivierungsfunktion g übergeben. Die Aktivierungsfunktion oder auch Schwellwert genannt gibt ein Signal weiter,

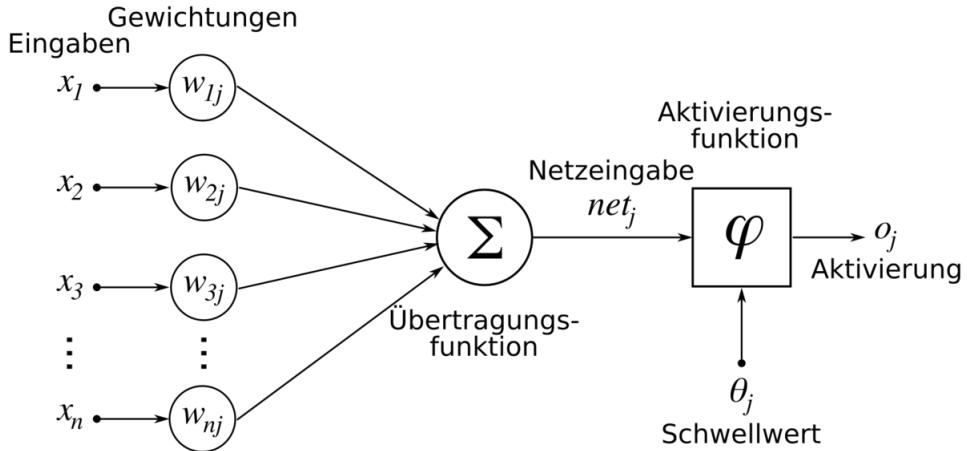


Abbildung 2.6: Schematische Modellierung eines Perzeptrons nach Rosenblatt [Wik19]

sobald dieser Wert überschritten wird.

$$g\left(\sum_{i=0}^n x_i * w_i\right) \quad (2.2)$$

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

$$g\left(\sum_{i=0}^n x_i * w_i + b\right) \quad (2.3)$$

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_{bias} , welches unabhängig von den Eingabewerten ist.

$$Bias = 1 * w_{bias} \quad (2.4)$$

Maschinelles Lernen möchte unter anderem Maschinen dazu 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 Teil mit Ein- und Ausgabedaten, welche nicht explizit wahr sein müssen und einen anderen 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 überwachtem Lernen mit systematischen Fehler und unüberwachtem Lernen, bei welchem das Netz nach häufig auftretenden Mustern sucht.

Bestärkendes Lernen

Das bestärkte Lernen arbeitet nach einem Prinzip, bei dem 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.

Künstliche neuronale Netze

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

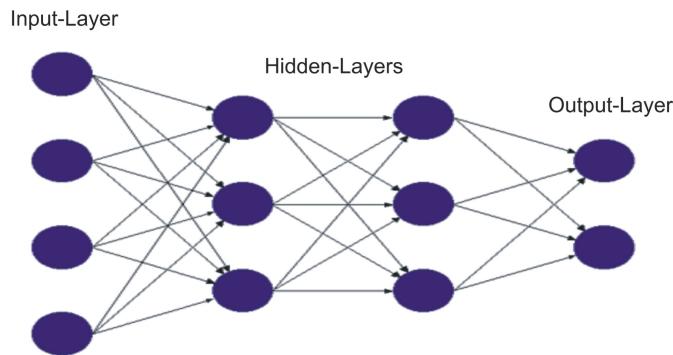


Abbildung 2.7: Aufbau eines künstlichen neuronalen Netzes [Ang18]

Der Aufbau von klassischen künstlichen neuronalen Netzen kann grundsätzlich in drei Bereiche unterteilt werden. Die erste Schicht stellt der *Input Layer* dar. Dieser besteht, wie alle folgenden Schichten auch, aus mehreren Neuronen. In dieser 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 werden. Diese Schichten werden auf mehrere verschiedene Merkmale der Eingabedaten trainiert. Dabei werden in den frühen Schichten zunächst grobe Strukturen erkannt, welche nach jeder weiteren Schicht immer feiner werden. Die letzte Schicht des *Hidden Layer* übergibt 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 (Abbildung 2.8) sind, anders als das künstliche neuronale Netz, an das Konzept des menschlichen Sehens angelehnt [SCL12] und für das Verarbeiten von matrixartigen Datensätzen konzipiert. Dazu zählen beispielsweise Bildaufnahmen [GBCB16]. Die erste Schicht besteht aus dem *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 (3×3 , 5×5 , 7×7) und eigene Gewichtungen. Sie bildet ein Skalarprodukt der unterliegenden Pixel und der Gewichtung. Dabei werden die Strukturen hervorgehoben. Diese Schicht kommt mehrmals in einem Netz vor.

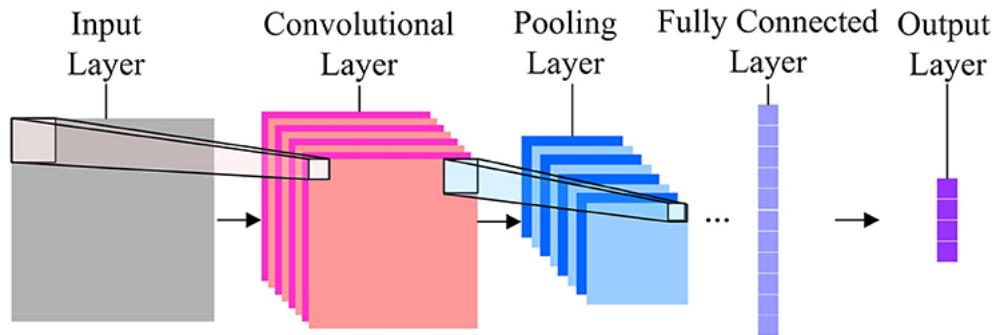


Abbildung 2.8: Aufbau eines Convolutional neural Network [PWCL16]

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 wird. 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 Perzepronen-Schichten und gibt die Informationen an den *Output Layer* weiter. Schlussendlich gibt dieser eine Hypothese mit der dazugehörigen Wahrscheinlichkeit aus.

Loss Funktion

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

$$E_{Euclid} = \frac{1}{2N} \sum_{n=1}^N \|\hat{f}_n - f_n\|_2^2 \quad (2.5)$$

Dabei stellt

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

$f_n \in [-\infty, +\infty]$ 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, dass 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 [GBCB16].

Funktionsweise künstlicher und faltender neuronaler Netze

Die zuvor aufgeführten Netzarten haben einen unterschiedlichen Aufbau und bestimmte Einsatzgebiete. Das Convolutional neural Network ist beispielsweise für das Verarbeiten von Bilddaten optimiert und künstliche neuronale Netze eher für das von seriellen Daten. In ihrer Struktur sind die Netzarten ähnlich aufgebaut. Das *Convolutional neural Network* hat neben den üblichen Neuronenschichten zwei zusätzliche Schichten, welche beim künstlichen neuronalen Netz nicht vorhanden sind. In den folgenden Absätzen soll die Funktionsweise dieser zusätzlichen Schichten beschrieben werden [GBCB16, 343].

Faltungsschicht

Es existiert eine Menge von Filtern n , welche eine Höhe von f_h und eine Breite von f_w haben. Diese Filter werden über das Eingabebild mit der Höhe e_h und der Breite e_w gelegt und in 1-Pixel-Schritten bewegt. Die Ergebnisse stellen eine Liste von Merkmalen dar, welche auf der Höhe m_h und der Breite m_w liegen. In Formeln ausgedrückt:

$$m_h = (e_h - f_h) + 1 \quad (2.6)$$

$$m_w = (e_w - f_w) + 1 \quad (2.7)$$

Um die Verringerung von Informationen nach der Faltung zu verhindern, werden an den Rändern der Bilder Polsterungen, auch *padding* genannt, angehangen [GBCB16,

343]. Diese Polsterungen bestehen aus Pixeln mit einem Wert von θ . Die Anzahl der zusätzlichen Pixel ergibt sich aus der Höhe und Breite des Eingabebildes. In Formeln ausgedrückt:

$$p_h = f_h - 1 \quad (2.8)$$

$$p_w = f_w - 1 \quad (2.9)$$

Die Faltungsschicht hat eigene Gewichte, welche beim Vorgang mit denen des Bildes zusammengerechnet werden [GBCB16, 331ff.]. Für die gleiche Schicht wird derselbe Filter verwendet. Die Weise der Berechnung wird in der Abbildung 2.9 beschrieben.

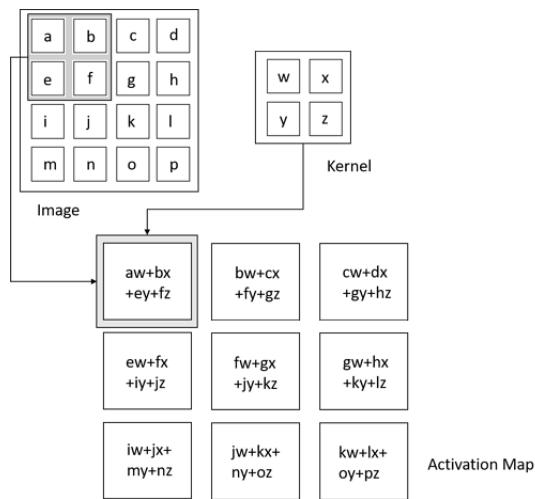


Abbildung 2.9: Funktionsweise der Faltungsschicht [GBCB16, 330]

Pooling Schicht

Die Pooling Schicht oder auch *Pooling Layer* kommt nach jeder Faltungsschicht [GBCB16, 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 2.10). 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 standardmäßigen Kernel von 2x2, wird das Bild um den Faktor 2 verkleinert. Die Abmessungen sind, anders als bei der Faltungsschicht, gerade.

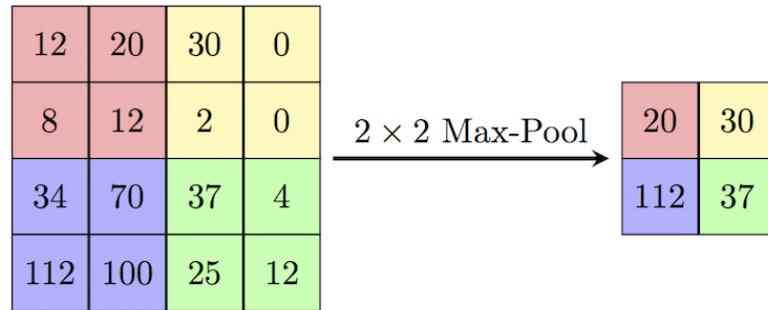


Abbildung 2.10: Funktionsweise des Pooling Layers mit der Max Pooling-Variante [sW18]

Fully connected Layer

Nachdem das Eingabebild durch alle Faltungs- und Pooling-Schichten verarbeitet wurde, werden die Daten an den Fully connected Layer übergeben [SCL12, 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 viele Informationen verarbeiten muss.

2.3.1 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 Datensätze. Das Labeln wurde mit dem Open-Source-Programm *LabelIMG* [Tzu19] durchgeführt. Hierbei wird das zu trainierende Objekt mit einem rechteckigen Kasten umschlossen und mit einer Klasse auf den Bildern markiert. Diese Daten werden zunächst in einer XML-Datei abgespeichert.

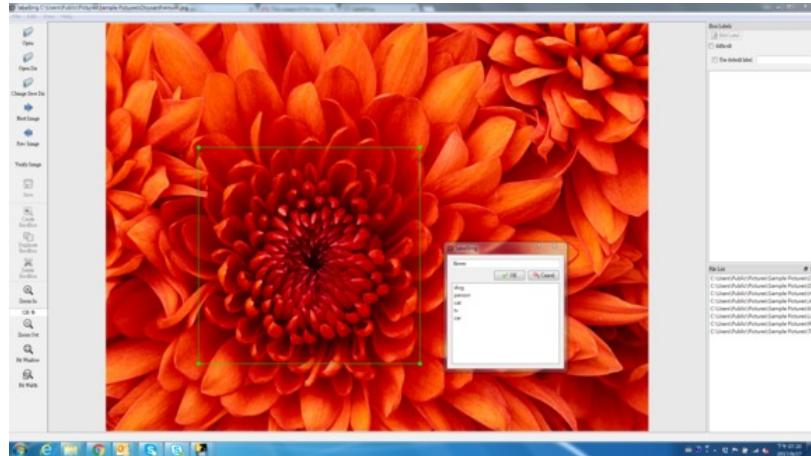


Abbildung 2.11: Vorbereitung der Datenstzen mit dem Labeling-Programm [Tzu19]

2.3.2 Mean average Precision

Oft können neuronale Netze nicht nur ein Objekt erkennen, sondern geben als Ergebnis mehrere Hypothesen aus. Dadurch entsteht das Problem, dass 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) [Hui18]. 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 (Recall) ist der korrekte Anteil von allen möglichen Objekten. In Formeln ausgedrückt:

$$P = \frac{\text{richtig erkannte}}{\text{richtig erkannte} + \text{falsch erkannte}} \quad (2.10)$$

$$R = \frac{\text{richtig erkannte}}{\text{richtig erkannte} + \text{falsche nicht erkannte}} \quad (2.11)$$

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 haben aber weiterhin eine hohe Präzision.

2.4 Arten der Farbnormalisierung

Die Farbnormalisierung ist ein Thema der Bildverarbeitung, welches sich 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 *Machine Learning* problematisch, da diese Farbveränderungen zu Fehlern im Lernprozess und der späteren Genauigkeit führen. Farbnormalisierungs-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.

2.4.1 Ansatz des Gray-World-Algorithmus

Bei der Gray-World-Normalisierung [BB01] geht es nicht, wie der Name vermuten lässt, um die Umwandlung in ein Graustufenbild, eher werden die verschiedenen Farbräume abgedunkelt und eingegrenzt. Wenn über die Farben in einem Bild keine Annahmen getroffen werden können, wird davon ausgegangen, dass die Farbwerte in einem Bild sich vektoriell zu Grau addieren, also das der Mittelwert vom Bild Grau ist. Sollte dies nicht der Fall sein, wird davon ausgegangen, dass das auf eine farbige Beleuchtung zurückzuführen ist und normalisiert das Bild so, dass die Graue-Welt-Theorie erfüllt wird. Die Farbkanäle werden durch einen Skalierungsfaktor normalisiert so, dass der Mittelwert 128 entspricht. Es werden somit Farbstiche herausgefiltert. Wie schon erwähnt, ist diese Art der Farbnormalisierung für verschiedene Farbvariationen unveränderlich. Ein Problem dieses Normalisierungsverfahrens besteht darin, dass es nicht einfach für dynamische Szenen verwendet werden kann. Um solche Probleme zu lösen, gibt es mehrere Ansätze dieser Ausgleichung.

$$(\alpha R, \beta G, \gamma B) \rightarrow \left(\frac{\alpha R}{\frac{\alpha}{n} \sum_i R}, \frac{\beta G}{\frac{\beta}{n} \sum_i G}, \frac{\gamma B}{\frac{\gamma}{n} \sum_i B} \right) \quad (2.12)$$

Eine Veränderung der Beleuchtungsfarbe kann als Skalierung von α , β und γ für die R-, G- und B-Kanäle modelliert werden (2.12). Als solcher ist der Gray-World für die Variationen der Beleuchtungsfarbe unveränderlich. Dieses Vorgehen berücksichtigt aber nicht alle Intensitäten der Beleuchtungsintensität und ist nicht für dynamische Szenen geeignet. Aus diesem Grund gibt es verschiedenste Ansätze dieses Verfahrens.

2.4.2 Ansatz der Histogramm-Ausgleichung

Die Histogramm-Ausgleichung [GWM⁺03] ist eine nichtlineare Transformation, welche auf Grundlage der Histogramme arbeitet. Der Pixelrang wird dabei nicht verändert und kann bei jeder monoton steigenden Farbformation verwendet werden. Dieses Normalisierungsverfahren wird besser als der Gray-World-Algorithmus angesehen, da nicht nur die Farbkanäle beeinflusst werden, sondern auch die Farbverteilung in den Bildern. Durch die Histogramm-Ausgleichung entsteht ein dominanter blauer Kanal, wodurch das Bild oft unnatürlich erscheint. Für die mathematische Herleitung betrachten wir

Tabelle 2.1: 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	5	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

zunächst ein Graustufenbild X und geben mit n_i die Anzahl der Graustufen an, wobei i die Häufigkeit des auftretenden Pixels ist. Zu sehen ist dies in der Funktion 2.13.

$$p_x(i) = p(x = i) = \frac{n_i}{n}, 0 \leq i \leq L \quad (2.13)$$

L ist die Gesamtzahl aller Graustufen im Bild (normalerweise 256), n ist die Gesamtzahl der Pixel im Bild und $p_x(i)$ das normalisierte Histogramm für den Pixelwert i . Die Definition der Verteilungsfunktion kann man in 2.13 sehen:

$$cdf_x(i) = \sum_{j=0}^i p_x(j) \quad (2.14)$$

Um die Funktionsweise etwas verständlicher zu machen, folgt ein Beispiel in Form eines 8x8-Bildes (Tabelle 2.1) mit acht Graustufen. Die Pixelverteilung wird in dem tabellarischen Histogramm zusammengefasst. Aus der Pixelverteilung wird nun ein Ausgleich berechnet, welcher den kumulativen Pixelwert durch die Anzahl aller Pixel teilt und das Ergebnis wiederum durch die höchste Graustufe. Das wird für jede Grauabstufung durchgeführt, wie in Tabelle 2.2 beschrieben. Das Ergebnis wird nun gerundet und der passenden Graustufe zugeordnet. Die Vorgehensweise mit Farbbildern ist, abgesehen

davon, dass die R-, G- und B-Kanäle verwendet werden, gleich. Häufig wird für diese Verfahren ein Farbraum gewählt, welcher einen Kanal für die Luminanz besitzt. Die

Tabelle 2.2: Mathematische Umsetzung der Histogrammausgleichung

r_k	P_k	Kumulative Werte	$Kumulative/Gesamt * (L - 1)$	Gerundete 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

Graustufen werden im nächsten Schritt übernommen und in das Quellbild eingesetzt. Hierbei verschieben sich die Grauwerte so, dass sie den gerundeten Grauwerten aus Tabelle 2.2 entsprechen. Das ausgeglichene Bild kann man in der Tabelle 2.3 betrachten.

Tabelle 2.3: Bildmatrix des ausgeglichenen Histogramms

1	2	6	2	7	3	1	3
1	1	6	6	6	3	5	6
5	6	2	5	2	6	2	5
6	2	3	5	6	3	7	3
6	3	7	5	1	5	1	6
5	1	3	5	7	5	7	3
6	2	7	2	1	2	2	6
5	6	3	5	3	6	3	6

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

2.4.3 Ansatz der Histogramm-Spezifikation

Ähnlich wie die Histogramm-Ausgleichung arbeitet die Histogramm-Spezifikation [GWM⁺03]. 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 Farbverteilung haben. Zunächst muss das Bild so konvertiert werden, dass das Histogramm eine bestimmte Farbverteilung hat. Für diese Methode werden üblicherweise zwei ähnliche Bilder verwendet, das Quell- und das Referenzbild. Das Referenzbild hat die gewünschte Form des Histogramms, wohingegen das Quellbild angepasst werden soll. Häufig wird zunächst eine Histogramm-Ausgleichung der beiden Bilder durchgeführt. Sie kann aber auch mit den Originalbildern durchgeführt werden. Im Weiteren wird das Histogramm des Quellbildes auf das Histogramm des Referenzbildes angepasst. Die mathematische Herleitung beginnt auch hier mit einem Grasstufeneingabebild S . Dieses Bild hat eine Wahrscheinlichkeitsdichtefunktion $p_r(r)$, wobei r ein Graustufenwert ist und $p_r(r)$ die Wahrscheinlichkeit für diesen Wert darstellt. Aus dem Histogramm des Bildes, kann die Wahrscheinlichkeit berechnet werden:

$$p_r(r_j) = \frac{n_j}{n} \quad (2.15)$$

Dabei ist n_j die Frequenz des Graustufenwerts r_j und n die Gesamtzahl der Pixel im Bild. Bei Betrachtung der gewünschten Ausgabewahrscheinlichkeitsdichtefunktion $p_z(z)$, ist eine Transformation von $p_r(r)$ erforderlich, damit es nach $p_z(z)$ angepasst werden kann. Gegeben sind die Funktionen der beiden Bilder:

$$S(r_k) = \sum_{j=0}^k p_r(r_j), \quad k = 0, 1, 2, 3, 4, \dots, L \quad (2.16)$$

$$G(z_k) = \sum_{j=0}^k p_z(z_j), \quad k = 0, 1, 2, 3, 4, \dots, L \quad (2.17)$$

Hierbei stellt S das Quellbild (2.16) und G das Referenzbild (2.17) dar. k beinhaltet alle Graustufen bis hin zur höchsten Graustufe L . Auch in dieser Herleitung bildet die Gesamtzahl an Graustufen (Normalerweise 256). Im weiteren wird versucht, jeden r -Wert in S auf dem z -Wert von G abzubilden, welche die gleiche Wahrscheinlichkeit

haben. Das heißt:

$$S(r_j) = G(z_i) \quad \text{oder} \quad z = G^{-1}(S(r)) \quad (2.18)$$

Um die Funktionsweise verständlicher darzustellen, wird kurz auf das Vorgehen und die mathematische Herleitung eingegangen. Dafür gibt es in den Tabellen 2.4 und 2.5 zwei Histogramme mit acht Graustufungen. In diesen Tabellen wird aufgeführt, wie oft der jeweilige Grauwert in dem Bild vorkommt.

Tabelle 2.4: Tabellarisches Histogramm des Quell-Bildes (B1)

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	5	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

Tabelle 2.5: Tabellarisches Histogramm des Referenz-Bildes (B2)

4	6	5	6	6	7	5	5
5	5	4	4	4	7	4	4
5	6	4	5	5	6	6	5
5	4	7	4	5	4	6	7
4	5	5	5	4	4	6	5
6	5	4	5	6	6	7	4
6	4	5	4	7	4	6	5
7	6	6	5	4	5	6	7

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

Nun werden die Histogramme, wie in Abschnitt 2.4.2 beschrieben, ausgeglichen. Von 0 bis $L - 1$ werden die Summen der Pixel berechnet, durch die Anzahl der im Bild enthaltenden Pixel geteilt und mit der höchsten Graustufe ($L - 1$) multipliziert. Das Ergebnis des Ausgleiches ist in Tabelle 2.6 dargestellt. Nun werden die Grauwertverteilungen von B1 und die Verteilung von B2 angepasst. Dafür wird der nächste Wert aufgerundet von B1 übernommen.

Ein Beispiel von Tabelle 2.6:

Bei Graustufe 0 liegt der Wert von B1 bei 1. Der nächst höhere Wert in B2 liegt bei

Tabelle 2.6: 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

2 an der Graustufe 4. Dieser Wert wird in das Zielhistogramm übernommen. Dabei werden alle Pixel, welche vorher den Grauwert 0 haben in den Grauwert 4 übertragen. Nach diesem Prinzip wird für jeden Farbwert des Histogramms vorgegangen.

Aus der neuen Stufen-Verteilung entsteht ein neues Histogramm (Tabelle 2.7), welches Ähnlichkeiten zum Referenz-Histogramm hat. Dadurch verschiebt sich das Histogramm in Richtung Weiß und das Bild wird erhellert.

Tabelle 2.7: Tabellarisches Histogramm des Quell-Bildes (B1) auf Basis des Referenz-Bildes (B2)

4	4	6	4	7	5	4	5
4	4	6	6	6	5	6	6
6	6	4	6	4	6	4	6
6	4	5	6	6	5	7	5
6	7	7	6	4	6	4	6
6	4	7	6	7	6	7	7
6	4	7	4	4	4	4	6
6	6	5	6	5	6	5	6

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

3 Methodik und Durchführung

In diesem Kapitel werden die verwendeten Datensätze aufgeführt und die Umsetzung der Normalisierungsalgorithmen beschrieben.

3.1 Vortrainierte Netze

Das Unternehmen COCO [Coc18] (Common Objects in Contexts) stellt eine Vielzahl an Datensätzen bereit, mit denen neuronale Netze trainiert werden können. Bei den Datensätzen handelt es sich um unterschiedliche Objekte in verschiedenen Umgebungen. Einige Netze, welche mit den COCO-Datensätzen trainiert wurden, stehen als *Open Source* zur Verfügung. Die folgende Tabelle 3.1 führt ausgewählte neuronale Netze auf, um Genauigkeit und Geschwindigkeit vergleichen zu können, damit ein geeignetes Netz für die Versuche ausgewählt werden kann. Um die Auswirkungen der Normalisierung

Tabelle 3.1: Künstliche neuronale Netze [LLC18c], welche mit dem COCO Dataset trainiert wurden [Coc18]

Neuronales Netz	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

deutlich hervorheben zu können, wurde ein Netz ausgewählt, welches eine recht niedrige mAP hat. Deswegen wurde das faster_rcnn_inception_v2_coco ausgewählt. Es hat eine *mAP* von 28 und eine Geschwindigkeit von 32 ms. Dadurch, dass es die niedrigste *mAP* hat, sollten die Auswirkungen der Normalisierungsverfahren genauer erkennbar sein, als bei Netzen mit einer höheren mAP. Das ausgewählte Netz wird im Folgenden auf die Datensätze, welche im anschließenden Unterkapitel aufgeführt sind, angewendet.

3.2 Trainingsdaten

Wie bereits in Kapitel 2.3.1 beschrieben, werden zum trainieren eines neuronalen Netzes üblicherweise mehrere hunderttausend Daten benötigt. Durch die Verwendung eines

vortrainierten Netzes, werden für das Trainieren der eigenen Klassen nur noch durchschnittlich 150 Bilder pro Objektklasse benötigt. Das hat eine deutliche Verkürzung der Trainingsdauer zur Folge. Die Bilder der eigenen Daten werden mit einer 13-Megapixel-Kamera eines *Huawei P Smart* aufgenommen. Die verwendeten Objekte werden von vielen verschiedenen Seiten aufgenommen. Da die erstellten Bilder mehrere Megabyte groß sind, müssen diese auf 100-300 KB komprimiert werden. Wie auch der Test später zeigen wird, verlängern sich die Laufzeiten der Algorithmen exponentiell, je größer das Bild ist.

Damit am Ende des Projektes möglichst gute Vergleichswerte entstehen, werden mehrere Datensätze für den Versuch verwendet. Insgesamt werden die neuronalen Netze mit drei unterschiedlichen Datensätzen trainiert. Der erste Datensatz, welcher getestet wird, ist der Nahrungsmittel-Datensatz eines früheren Projektes, der Zweite der Datensatz von *Pascal Visual Object Classes* (PascalVOC) und der Dritte der Hunderassen-Datensatz, Stanford Dog Dataset.

In den folgenden Tabellen werden alle Datensätze mit den dazugehörigen Klassen aufgeführt.

Der erste Datensatz besteht aus sechs Klassen, welche in der Tabelle 3.2 aufgeführt sind. Jede Klasse besitzt um die 100-150 Bilder, die zusätzlich Annotations-Dateien mit den Koordinaten der Klassifizierungsboxen (Kapitel 2.3.1) beinhalten. Dieser Datensatz ist mit seinen ca. 1.000 Bildern der kleinste Datensatz. Der PascalVOC-Datensatz

Tabelle 3.2: Datensatz mit Nahrungsmitteln

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

umfasst 20 Klassen und besteht aus 5.000 Bildern. Von 2005 bis 2012 wurde jährlich die PascalVOC-Challenge 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 3.3 zu erkennen sind. Es gibt einige Unterschiede in der Häufigkeit der auftretenden Klassen. Beispielsweise sind in rund 2.000 Bildern insgesamt 4.690 Personen enthalten, weswegen es möglicherweise Unterschiede in der Genauigkeit 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. Der dritte Datensatz, welcher in dieser Arbeit verwendet wird, ist der Hunde-Datensatz aus Stanford und beinhaltet 120 verschiedene Klassen.

Tabelle 3.3: Pascal Visual Object Classes [EVGW⁺07]

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

dene 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 Annotierungen der Daten sind in Form von XML-Dateien beigelegt. In der Tabelle 3.4 werden die ausgewählten Klassen des Datensatzes zusammengefasst. Die Struktur der Datensätze ist ähnlich aufgebaut.

Tabelle 3.4: Stanford Dog Dataset [KJYFF11]

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

Das heißt, viele verschiedene Klassen mit verschiedenen Szenen und Kontexten. Im Folgenden wird zunächst auf die Umsetzung der Normalisierungsverfahren eingegangen, um weitere Erkenntnisse zu erlangen.

3.3 Normalisierungs-Algorithmen

Für die Normalisierung der Datensätze ist eine Methode nötig, mit der mehrere Bilder möglichst schnell hintereinander bearbeitet werden können. Für die Normalisierung wurden Python-Programme geschrieben, welche nacheinander die Bilder, anhand eines Algorithmus, verarbeiten. Dafür wurden mitunter die *Python Image Library* und OpenCV genutzt. Beide Bibliotheken werden für das Verarbeiten von Bildern benötigt.

3.3.1 Gray-World-Algorithmus

Im Folgenden wird der Codeblock für den Gray-World-Algorithmus aufgeführt, wie er in den Versuchen verwendet wurde.

```

1 image = cv.imread(i, 1)
2 image = image.transpose(2, 0, 1).astype(np.uint32)
3 averageGreenChannel = np.average(image[1])
4 image[0] = np.minimum(image[0]*(avgGreen/np.average(image[0])),255)
5 image[2] = np.minimum(image[2]*(avgGreen/np.average(image[2])),255)
6 img_output = image.transpose(1, 2, 0).astype(np.uint8)

```

Der Gray-World-Algorithmus [Aih12], welcher hier verwendet wird, funktioniert, wie in den Grundlagen beschrieben. Für die Normalisierung der Kanäle wird der Durchschnitt des Grünkanals verwendet, da diese Farbe wichtig für das Helligkeitsempfinden ist [up19]. Zunächst wird in Zeile 1, das zu normalisierende Bild mittels OpenCV importiert. In der Zeile 2 wird der Farbraum des Bildes in 32 Bit umgewandelt. Daraufhin wird in Zeile 3 der Durchschnittswert des Grünkanals berechnet und zwischengespeichert. In der Zeile 4 wird der Grünkanal nun mit dem Durchschnitt des Rotkanals dividiert und anschließend mit dem Minimum des Rotkanals multipliziert und übernommen. Der gleiche Vorgang erfolgt mit dem Blaukanal in Zeile 5. Abschließend wird das bearbeitete Bild in 8 Bit umgewandelt und ausgegeben (Abbildung 3.1).



Abbildung 3.1: Auswirkung des Gray-World-Algorithmus

Histogramm-Ausgleich

Weiter geht es mit dem ersten Histogramm-Normalisierungsverfahren, der Histogramm-Ausgleichung. Diese wird im Folgenden aufgeführt und beschrieben.

```

1 image = cv.imread('input.jpg')
2 image_yuv = cv.cvtColor(image, cv.COLOR_BGR2YUV)
3 image_yuv[:, :, 0] = cv.equalizeHist(image_yuv[:, :, 0])
4 img_output = cv.cvtColor(image_yuv, cv.COLOR_YUV2BGR)

```

Für die Histogramm-Ausgleichung [Jaz16] wird das Bild in Zeile 2 vom BGR-Farbraum in den YUV-Farbraum (Unterabschnitt 2.1.1) umgewandelt. 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. In Zeile 4 wird das Histogramm des Y-Kanals, wie in Unterabschnitt 2.4.2 beschrieben, ausgeglichen. Das normalisierte Bild wird von dem YUV-Farbraum zurück in den BGR-Farbraum konvertiert und zwischengespeichert. Das Ergebnis kann in Abbildung 3.2 betrachtet werden.



Abbildung 3.2: Auswirkung der Histogramm-Ausgleichung

3.3.2 Histogramm-Spezifikation

Der letzte Algorithmus, welcher geprüft wird, ist die Histogramm-Spezifikation. Hier wird kurz auf die Umsetzung und ein Beispielergebnis eingegangen.

```

1 source_image = io.imread('source_image.jpg')
2 reference_image = io.imread('reference_image.jpg')
3 matched_image = match_histograms(source_image, reference_image,
    multichannel=True)

```

Die dritte Normalisierungsfunktion, die auf die Datensätze angewendet wird, ist die Histogramm-Spezifikation (Unterabschnitt 2.4.3) oder auch *Histogramm Matching*. Für den Algorithmus wird die Python-Bibliothek *Scikit-image* verwendet. Hierfür werden ein Quellbild (Zeile 1) und ein Referenzbild (Zeile 2) geladen. Diese beiden Bilder werden mit der Funktion *match_histograms* verarbeitet (Zeile 3). Das Quellbild wurde auf das Histogramm des Referenzbildes angepasst. In dem Beispiel 3.3 kann man gut erkennen, dass das Quellbild, welches stark verdunkelt ist, durch ein gut ausgeleuchtetes Referenzbild, eine wesentlich bessere Helligkeit aufweist. Die Problematik bei dieser Normalisierungs-Methode besteht darin, dass ein ähnliches Referenz-Histogramm genutzt werden muss, auf welchem dann der gesamte Datensatz normalisiert wird.



Abbildung 3.3: 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 die Entwicklungsumgebung beschrieben. Anschließend werden die Ergebnisse untersucht und ausgewertet.

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 hauptsächlich Grafikprozessoren für die Verarbeitung verwendet, da diese, im Gegensatz zu normalen Prozessoren, für die Berechnung von Matrizen konzipiert wurden.

Auf Softwareseite gibt es eine Menge an Frameworks, mit welchen 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 von künstlichen neuronalen Netzen genutzt werden kann. Programmiert wurde das Framework in den Programmiersprachen C und Python, welche auch für die Entwicklung genutzt werden. Tensorflow verfügt über eine übersichtliche Dokumentation. Außerdem stehen eine Menge an vortrainierte Netzen als Open Source zur Verfügung. Mithilfe von Tensorboard, welches mit dem Framework zu Verfügung gestellt wird, können die Genauigkeiten eines 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 wie folgt verwendet: N = Normaler Datensatz, GW = Gray-World-Algorithmus, HA = Histogramm-Ausgleich, HS = Histogramm-Spezifikation (Die besten Ergebnisse einer Klasse wurden durch Rot und Grün hervorgehoben).

4.2 Nahrungsmittel-Datensatz

Der erste Datensatz, welcher untersucht wurde ist der Nahrungsmitteltest, der in der Tabelle 4.1 aufgeführt ist. Dieser enthält, im Vergleich zu den anderen Datensätzen, weniger Objektklassen und weniger Bilder. Außerdem sind die Objekte unter ähnlichen Bedingungen entstanden. Das bedeutet, dass die Lichteinstrahlung in den meisten Fällen ähnlich ist und die Objekte nur in wenigen unterschiedlichen Szenen aufgenommen wurden. In der Tabelle 4.1 sind die Genauigkeiten des Netzes aufgeführt, welche mit den unterschiedlich behandelten Datensätzen trainiert wurden. Bei genauerer Betrachtung 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 Punkten wiederum ist die Genauigkeit gesunken (Milch, Brunch, mAP).

Bei den Histogramm-Normalisierungen (Ausgleich und Spezifizierung) können größere Unterschiede festgestellt werden. Auch hier sind die Unterschiede in einzelnen Objektklassen besser geworden und in anderen zurückgegangen. Die Genauigkeit der verschiedenen Netze variiert bei dieser hohen Genauigkeit sehr stark, was in der Praxis einen bedeutenden Unterschied machen kann. Insgesamt schneiden die Histogramm-Normalisierungen am schlechtesten ab. Auch ein Blick auf die Datensätze der Histogramm-Normalisierungen zeigt, dass in vielen Fällen Anomalien in den Bildern aufgetreten sind. Gut zu erkennen sind diese in den Abbildungen .6 und .7, im Anhang.

Tabelle 4.1: Durchschnittliche Genauigkeiten des Modells mit dem Nahrungsmittel-Datensatz

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

4.3 PascalVOC Datensatz

Beim Auswerten der Genauigkeit des normalen Modells fällt auf, dass teilweise große Unterschiede der einzelnen Objektklassen auftreten. Das kann unter anderem daraus resultieren, dass unterschiedlich viele Bilder pro Klasse enthalten sind. Ähnlich wie beim vorherigen Datensatz, konnten zwischen dem originalen Datensatz und der Gray-

World-Normalisierung leichte Verluste festgestellt werden. Einige Klassen haben zwar eine höhere durchschnittliche Wahrscheinlichkeit, dennoch gibt es auch hier mehrere Bereiche, in welchen die Genauigkeit abgenommen hat. Auch die mAP hat in der Genauigkeit 0,8 Prozentpunkte verloren. Anders als beim Nahrungsmittel-Datensatz schneidet der Histogramm-Ausgleich weitaus schlechter ab. Nur ein paar Objektklassen konnten besser erkannt werden. Insgesamt verliert das neuronale Netz 3,4 Prozentpunkte in der mAP, was einen wichtigen Unterschied macht. Einige Objektklassen sinken durch die Ausgleichung sogar unter die 50%-Genauigkeit. Hier wird auch die Schwäche der Histogramm-Spezifikation deutlich. Dadurch, dass keine einheitliche Szene verwendet wurde, sinkt die mAP um 0.045. Die Datensätze haben, im Vergleich zum vorherigen Datensatz, deutlich mehr Anomalien (siehe Abbildung .6 und .7 im Anhang).

Tabelle 4.2: Durchschnittliche Genauigkeiten des Modells mit dem PascalVOC-Datensatz

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

4.4 Stanford Dogs Dataset

Auch bei Überprüfung des dritten Datensatzes fallen ähnliche Muster auf. Der GW-Algorithmus weiß bei diesem Datensatz bessere Ergebnisse auf. Sogar eine Verbesserung

rung gegenüber des originalen Datensatzes konnte erzielt werden. Beim vergleichen der Datensätze fällt auf, dass mehr verschiedene Lichtfarben genutzt wurden, als bei den anderen beiden Datensätzen. Diese konnten vom GW-Algorithmus angepasst werden. Beim Histogramm-Ausgleich und der -Spezifikation wurden, wie auch bei den vorherigen Datensätzen, deutliche Verschlechterungen in der Genauigkeiten festgestellt. Auch hier zeigt einen Blick in die normalisierten Trainingsdaten, dass teilweise starke Farbanomalien aufgetreten sind. Diese behinderten das Training mehr, als das sie hilfreich waren.

Tabelle 4.3: 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 abnehmenden Genauigkeiten in der Klassifizierung entstanden sind, wurden die Trainingsdaten überprüft. Dabei ist aufgefallen, dass bei der Histogramm-Ausgleichung und der -Spezifikation große Unterschiede in den Datensätzen vorhanden sind. Das könnte daran liegen, dass die Normalisierung die Farbinformationen des gesamten Bildes nimmt und diese anpasst. Hier macht es einen großen

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 kein einheitliches Referenzbild genutzt werden, da fast jedes Bild unter anderen Bedingungen entstanden ist. Durch diese unterschiedlichen Trainingsbilder kommt es zu Anomalien in den normalisierten Bildern (Abbildung .2). Um diese Vermutung zu bestärken, wurden Objekte mit gleicher Lichteinstrahlung und gleicher Entfernung auf unterschiedlichen Hintergründen aufgenommen (Abbildung 4.1). Daraufhin 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. (Abbildung .2 im Anhang).

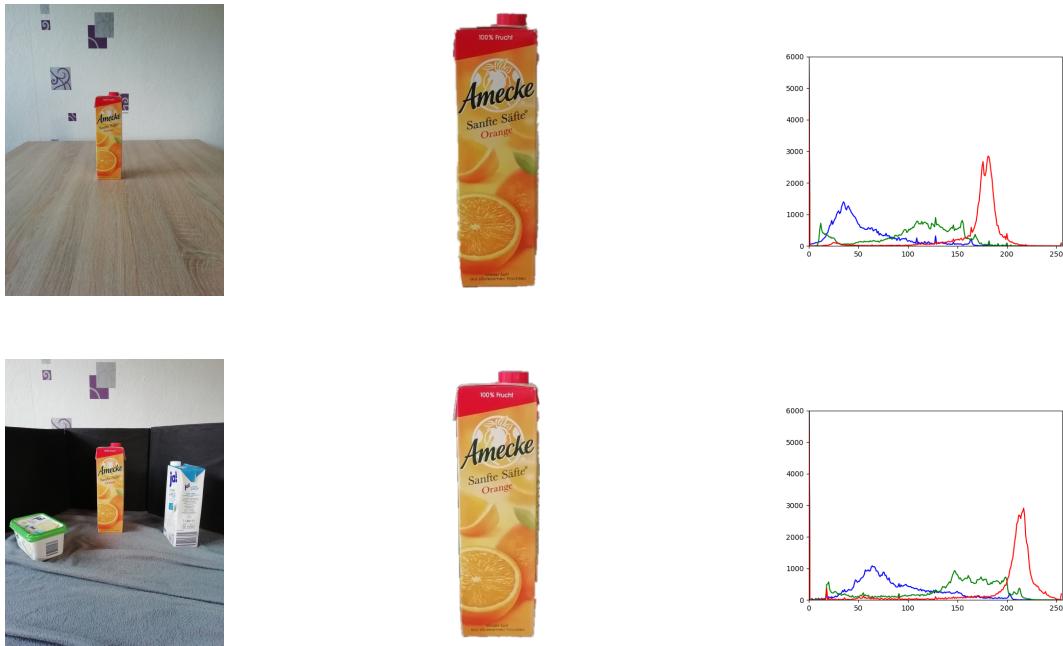


Abbildung 4.1: Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz ohne Normalisierung

Durch diesen Test wurde bestätigt, dass 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 selbst. Ein Datensatz mit einheitlichem Hintergrund in den Trainingsdaten, könnte mit den Normalisierungsverfahren besser funktionieren. Um diese Annahme weiter überprüfen zu können, soll ein weiterer Datensatz generiert werden, in dem ein einheitlicher Hintergrund festgelegt ist.

4.6 Obst-Datensatz

Für einen weiteren Versuch wurde der Obstdatensatz erstellt. Dieser besteht aus den Objekten, welche in Tabelle 4.4 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 3.2), darauf geachtet, ähnlichere Klassen zu verwenden, auf welchen optimalerweise keine Schrift enthalten ist. Das Netz soll nicht durch die Schrift beeinflusst werden, sondern hauptsächlich die Farben als Orientierung nehmen. Dieser

Tabelle 4.4: Klassen des Obst-Datensatzes mit konstantem Hintergrund



Klassenname	Klassenname
Orange	Mango
Banane	Nektarine
Apfel	Birne

Datensatz wird, genau wie die vorherigen Datensätze, mit den unterschiedlichen Methoden normalisiert. Dabei soll zum einen die Genauigkeit der einzelnen Klassen betrachtet werden und zum anderen die Trainingsdaten an sich. Hierbei 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 zwei der drei Normalisierungsmethoden einen positiven Einfluss auf die Klassifizierung haben. Die Genauigkeit der Histogramm-Spezifikation schneidet mit einer mAP von 97% am besten ab. Durch diese Methode konnte eine Erhöhung von 0,4 Prozentpunkten erzielt werden. Auch der Gray-World-Algorithmus hat eine leichte Verbesserung von 0,1 Prozentpunkt bewirkt. Lediglich der Histogramm-Ausgleich hat die Klassifizierung leicht verschlechtert. Eine weitere Möglichkeit den Einfluss zu überprüfen wurde mit Hilfe von besonders schwierigen Testbildern versucht. Dafür sollen Bildaufnahmen mit extremen unterschiedlichen Lichteinflüssen verwendet werden. Die Auswertung der normalisierten Trainingsdaten hat ergeben, dass wesentlich weniger Farbanomalien auftreten, als bei den vorherigen Datensätzen. Dabei ist aufgefallen, dass gerade bei den Histogramm-Normalisierungen mit einheitlichem Hintergrund deutliche Verbesserungen erzielt wurden. Große Veränderungen treten nur dann auf, wenn viele Objekte auf dem Bild

Tabelle 4.5: Genauigkeitsberechnungen des Modells des Obst-Datensatzes

Klassenname	AP(N)	AP(GW)	AP(HA)	AP(HS)
Orange	0.982	0.985	0.983	1.000
Nektarine	0.986	0.988	0.993	0.987
Banane	1.000	1.000	1.000	1.000
Mango	0.993	0.996	0.983	1.000
Apfel	0.995	0.996	0.990	0.999
Birne	0.999	0.998	1.000	1.000
mAP	0.993	0.994	0.992	0.997

enthalten sind, da diese die Umgebung zu stark beeinflussen oder bei der Histogramm-Ausgleichung, wenn die Beleuchtung zu extrem verändert wurde. Die Histogramm-Spezifikation geht mit diesem Problem am besten um, weil sie ein Referenz-Histogramm verwendet, um die Daten anzupassen.

Auch die Helligkeitsverteilungen der Trainingsdaten werden wesentlich besser, wie in den Beispielen in der Abbildung .5 zu erkennen ist. Dafür wurde eine stark unterbelichtete Aufnahme generiert, um die Veränderung nach den Normalisierungsmethoden 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-Algorithmus hat die Helligkeitsverteilung kaum verändert, was daran liegt, dass die Farbverteilung durch die Höhe des Grünkanals 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.

5.1 Datensätze und Netze

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 sind die Datensätze, welche normalisiert wurden. Da manche Normalisierungsverfahren die Farbinformationen des gesamten Bildes verwendeten, ist das Ergebnis der Normalisierungen, je nach Umgebung, in den Bildern unterschiedlich ausgefallen. Um diese Annahme zu überprüfen, wurden Testaufnahmen von Objekten auf verschiedenen Hintergründen mit gleicher Belichtung und gleichem Aufnahmewinkel erstellt. 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 bei den Farben. Das könnte daran liegen, dass die Normalisierung nicht nur die Objekte auf den Bildern normalisiert, sondern auch den Hintergrund, welcher bei vielen der Trainingsbilder unterschiedlich ist.

Die Überprüfung hat gezeigt, dass die Datensätze für den verwendeten Aufbau und den verwendeten Normalisierungsalgorithmen nicht geeignet waren. Bessere Ergebnisse konnten mit dem im nachhinein generierten Obst-Datensatz erzielt werden, bei dem alle Trainingsdaten auf demselben Hintergrund aufgenommen wurden. Dadurch konnten alle Klassen auf derselben Basis normalisiert werden. Gerade die Histogramm-Spezifikation (Abbildung .5) hat hier besonders gute Ergebnisse erzielt, da der Einsatz eines Referenzbildes besser funktioniert hat und wesentlich weniger Anomalien erzeugt wurden. Auch bei der Untersuchung der Helligkeit konnten wesentliche Verbesserungen festgestellt 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 zudem 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 Grünkanals verschiebt und nur Lichtfarben ausgleicht.

5.2 Laufzeittest

Ein weiterer Einfluss, welchen die Normalisierungsalgorithmen auf die Klassifizierung von neuronalen Netzen haben, ist, neben dem Manipulieren von Trainingsdaten, die Zeit welche das jeweilige Verfahren benötigt. Um herauszufinden, wie sich die Laufzeit bei größer werdenden Bilddateien verhält, wurde ein Bild in verschiedenen Größen getestet. Der Gray-World-Algorithmus liegt, im Vergleich zu den anderen Methoden,

Tabelle 5.1: Laufzeiten der Normalisierungs-Algorithmen mit verschiedenen großen Bildern

Bildgröße	Faktor	Dateigröße	GW(s)	HA(s)	HS(s)
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

bei der Laufzeit im Mittelfeld. Bei Verdoppelung der Bildgröße erhöht sich die Laufzeit um das Vierfache. Dabei sind sowohl die Farbgebung, als auch der Kontrast nicht entscheidend. Die Histogramm-Ausgleichung ist in diesem Fall die schnellste Methode, wobei hierbei die Verteilung im Histogramm eine Rolle spielt. Die Laufzeit verändert sich nicht linear, sondern steigt exponentiell an. So steigt die Laufzeit beim Verdoppeln um den Faktor drei und beim nächsten Mal um den Faktor sechs. Der längste Normalisierungsalgorithmus 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 einem gleichbleibenden Referenzbild erhöht sich die Laufzeit um den Faktor vier. Durch die hohe Grundlaufzeit führt das schnell zu langen Laufzeiten.

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

6 Fazit

Zunächst wird auf die Anforderungen eingegangen und wie gut diese erfüllt werden konnten. Für ein abschließendes Fazit wird anschließend erläutert auf welche Art von Datensätzen, die in dieser Arbeit untersuchten Normalisierungsverfahren angewendet werden können und wo die Schwächen liegen.

Bei der ersten Anforderung ging es um das Training des neuronalen Netzes und um eine Möglichkeit mit weniger Bildern pro Datensatz auszukommen. Dieser Punkt konnte erfüllt werden, indem vortrainierte künstliche neuronale Netze verwendet wurden. Dadurch sind die Mengen der Trainingsdaten auf durchschnittlich 200 Bilder pro Klasse gesunken.

Bei den Anforderungen zwei bis fünf wurden die Datensätze, welche bei dem Versuch verwendet werden sollten, spezifiziert. Die Anforderung zwei, bei der es um die Ordentlichkeit der Datensätze geht, konnte nicht bei jedem Datensatz erfüllt werden. Der PascalVOC-Datensatz hat die Trainingsdaten gemischt und erschwerte dadurch das Verändern der einzelnen Klassen. Die dritte Anforderung, in der es um die Größe der Bilder ging, wurde bei jedem Datensatz erfüllt. Die Bilder haben eine Größe zwischen 100-300 KB. Anforderung vier, in der es um die Ausleuchtung der Trainingsdaten ging, wurde nur teilweise erfüllt, da für das Testen der neuronalen Netze auch schlecht ausgeleuchtete Testdaten benötigt wurden. Bei Anforderung fünf ging es um die Definition eines eingegrenzten Themas, der einzelnen Datensätze. Jeder Datensatz hatte ein gut definiertes Thema, welches unter anderem sehr ähnliche Klassen besaß. Beim PascalVOC-Datensatz war dieser teilweise ziemlich willkürlich, hat aber in der Klassifizierung relativ gut funktioniert.

Bei der Anforderungen sechs ging es um die Anwendung der Normalisierungsverfahren auf verschiedene Datensätze. Dabei sollte herausgestellt werden, ob und unter welchen Bedingungen eine Farbnormalisierung einen positiven Einfluss, auf eine Klassifizierung durch neuronale Netze, hat. Dabei sind je nach Normalisierungsverfahren unterschiedliche Ergebnisse entstanden, welche im Folgenden nacheinander behandelt werden.

Gray-World-Algorithmus

Der Gray-World-Algorithmus kann prinzipiell bei jedem Datensatz angewendet werden. Der Vorteil wird deutlich, wenn verschiedene Lichtfarben in der späteren Erkennung und im Datensatz auftauchen. Durch diese Methode werden Kontrast und Dynamik nicht verändert. Das Farbbild wird insgesamt auf Grundlage des Grünkanals verschoben. Da dieses Verfahren relativ simpel arbeitet, wird kein großer Einfluss auf spätere künstliche neuronale Netze erzielt.

Histogramm-Ausgleich

Der Histogramm-Ausgleich arbeitet auf Grundlage des Histogramms und versucht den Kontrast in dem Bild zu erhöhen. Eingesetzt werden kann es bei Datensätzen, welche einen einheitlichen Hintergrund verwenden, da für die Normalisierung die Farbverteilung des gesamten Bildes verwendet wird. Denn je nach Farbverteilung fällt das Ausgabebild, nach dem Ausgleich, unterschiedlich aus. Schwächen gibt es bei starken Lichtunterschieden. Sollte ein Bild zu dunkel sein, können Farbwerte nicht wiederhergestellt werden.

Histogramm-Spezifizierung

Auch die Histogramm-Spezifizierung ist nicht für allgemeine Datensätze geeignet, sondern für Datensätze mit einheitlichem Hintergrund. Das liegt zum einen an der Verarbeitung des Histogramms und zum anderen an der Tatsache, dass für dieses Verfahren ein Referenzbild benötigt wird, auf welchem der Datensatz normalisiert wird. Dieses Referenzbild ist effektiver, desto ähnlicher es dem Datensatz ist. Die Ergebnisse dieses Verfahrens haben beim Obst-Datensatz eine deutlich erhöhte Genauigkeit erzielt. Bei diesem Verfahren muss beachtet werden, dass es die längste Laufzeit hat und eine gewisse Arbeit voraussetzt.

In dieser Arbeit sollte überprüft werden, ob die Anwendung von Farbnormalisierungsalgorithmen einen positiven Einfluss auf die Genauigkeit der Klassifizierung, mittels künstliche neuronaler Netze, haben. Dabei ist herausgekommen, dass Farbnormalisierung einen positiven Einfluss haben kann. Wichtig dabei ist auf die Funktionsweise der Verfahren zu achten und zu prüfen für welche Szenarien diese sinnvoll eingesetzt werden können. Dafür ist es in manchen Fällen notwendig, die Datensätze an die Normalisierungen anzupassen.

Im weiteren sollen Ansätze aufgeführt werden, die die Forschung nach dieser Arbeit weiter verfolgen kann. In dieser Arbeit wurden die Verfahren mit vortrainierten neuronalen Netzen überprüft. Es wäre daher eine interessante Frage, wie im Vergleich dazu

untrainierte neuronale Netze die Verfahren beeinflussen. Zudem konnten im Rahmen dieser Bachelorarbeit nur Datensätze in der Mindestgröße generiert werden. Deswegen wäre es ebenso spannend herauszufinden, wie die Ergebnisse bei wesentlich größeren und ausführlicheren Datensätzen aussehen würden.

Außerdem könnte das Verhalten der Normalisierungsmethoden in Kombination miteinander untersucht werden. Möglicherweise können durch solche Kombinationen die unterschiedlichen Stärken der Verfahren zusammen genutzt werden.

Abbildungsverzeichnis

1.1	Szene mit unterschiedlicher Ausleuchtung	3
2.1	Aufbau eines Digitalen Bildes mit zwei Helligkeitsabstufungen	7
2.2	Histogramm eines Graustufen-Bildes mit 16 Helligkeitsabstufungen [BB09, 42]	9
2.3	Unterschiedlicher Kontrast und die Auswirkungen im Histogramm: niedriger Kontrast (links), normaler Kontrast (Mitte), hoher Kontrast (rechts)[BB09, 45]	10
2.4	Unterschiedliche Dynamik und die Auswirkungen im Histogramm: hohe Dynamik (links), niedrige Dynamik mit 64 Graustufen (Mitte), extrem niedriger Kontrast mit 6 Graustufen (rechts)[BB09, 45]	11
2.5	Aufbau eines Neurons [oCSD18]	13
2.6	Schematische Modellierung eines Perzeptrons nach Rosenblatt [Wik19] . .	14
2.7	Aufbau eines künstlichen neuronalen Netzes [Ang18]	16
2.8	Aufbau eines Convolutional neural Network [PWCL16]	17
2.9	Funktionsweise der Faltungsschicht [GBCB16, 330]	19
2.10	Funktionsweise des Pooling Layers mit der Max Pooling-Variante [sW18]	20
2.11	Vorbereitung der Datensätzen mit dem Labeling-Programm [Tzu19]	21
3.1	Auswirkung des Gray-World-Algorithmus	31
3.2	Auswirkung der Histogramm-Ausgleichung	32
3.3	Das Zielbild wurde mithilfe des Referenzbildes ausgeglichen und angepasst. Auf der linken Seite ist das Ergebnis der Spezifikation.	33
4.1	Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz ohne Normalisierung	38
.2	Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch den Histogramm-Ausgleich	II
.3	Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch den Gray-World-Algorithmus	III
.4	Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch die Histogramm-Spezifikation	IV
.5	Helligkeitsverteilung und der Einfluss der Normalisierungs-Algorithmen auf das Testbild	V
.6	Auftretende Farbanomalien in den drei Datensätzen durch den Histogramm-Ausgleich. Oben: PascalVOC-Datensatz, Mitte: Nahrungsmittel-Datensatz, Unten: Stanford Dog-Datensatz	VI

- .7 Auftretende Farbanomalien in den drei Datensätzen durch die Histogramm-Spezifikation. Oben: PascalVOC-Datensatz, Mitte: Nahrungsmittel-Datensatz, Unten: Stanford Dog-Datensatz VII

Tabellenverzeichnis

2.1	Pixelbild in ein tabellarisches Histogramm überführt	23
2.2	Mathematische Umsetzung der Histogrammausgleichung	24
2.3	Bildmatrix des ausgeglichenen Histogramms	24
2.4	Tabellarisches Histogramm des Quell-Bildes (B1)	26
2.5	Tabellarisches Histogramm des Referenz-Bildes (B2)	26
2.6	Von zwei ausgeglichenen Histogrammen zu einem spezifizierten Histogramm	27
2.7	Tabellarisches Histogramm des Quell-Bildes (B1) auf Basis des Referenz-Bildes (B2)	27
3.1	Künstliche neuronale Netze [LLC18c], welche mit dem COCO Dataset trainiert wurden [Coc18]	28
3.2	Datensatz mit Nahrungsmitteln	29
3.3	Pascal Visual Object Classes [EVGW ⁺ 07]	30
3.4	Stanford Dog Dataset [KJYFF11]	30
4.1	Durchschnittliche Genauigkeiten des Modells mit dem Nahrungsmittel-Datensatz	35
4.2	Durchschnittliche Genauigkeiten des Modells mit dem PascalVOC-Datensatz	36
4.3	Durchschnittliche Genauigkeiten des Modells mit dem Stanford-Dog-Datensatz	37
4.4	Klassen des Obst-Datensatzes mit konstantem Hintergrund	39
4.5	Genauigkeitsberechnungen des Modells des Obst-Datensatzes	40
5.1	Laufzeiten der Normalisierungs-Algorithmen mit verschiedenen großen Bildern	42

Literaturverzeichnis

- [Aih12] Shunsuke Aihara. color correction algorithm in python, 2012.
- [Ang18] Bistra Angelova. Optical character recognition für chinesische schrift, December 2018.
- [BB01] José M Buenaposada and Luis Baumela. Variations of grey world for face tracking. *Image Processing & Communications*, 7(3-4):51–61, 2001.
- [BB09] Wilhelm Burger and Mark James Burge. *Digitale Bildverarbeitung: Eine Algorithmische Einführung Mit Java*. Springer-Verlag, 2009.
- [Coc18] Coco. Coco datenset, December 2018.
- [cs2NA] cs231n. Convolutional neural networks for visual recognition, N/A N/A.
- [Dee16] DeepMind. Google deepmind challenge match: Lee sedol vs alphago, March 2016.
- [Edj18] Evan EdjeElectronics. Testbilder, December 2018.
- [Ert13] Wolfgang Ertel. *Grundkurs künstliche Intelligenz: eine praxisorientierte Einführung*. Springer-Verlag, 2013.
- [EVGW⁺07] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <https://bit.ly/2Yy6tFG>, 2007.
- [GBCB16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [GWM⁺03] Keith A Goatman, A David Whitwam, A Manivannan, John A Olson, and Peter F Sharp. Colour normalisation of retinal images. In *Proceedings of medical image understanding and analysis*, pages 49–52, 2003.
- [HE16] Joachim Hoffmann and Johannes Engelkamp. *Lern- und Gedächtnispsychologie*. Springer-Verlag, 2016.
- [Hui18] Jonathan Hui. map (mean average precision) for object detection, März 2018.
- [Jäh13] Bernd Jähne. *Digitale bildverarbeitung*. Springer-Verlag, 2013.
- [Jaz16] Mohammad Al Jazaery. Opencv python equalizehist colored image, july 2016.

- [KJYFF11] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. 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, June 2011. Aditya Khosla and Nityananda Jayadevaprakash and Bangpeng Yao and Li Fei-Fei.
- [Küp17] EW Udo Küppers. *Die humanoide Herausforderung: Leben und Existenz in einer anthropozänen Zukunft*. Springer, 2017.
- [LLC18a] Google LLC. Bild 3b umschlossen, December 2018.
- [LLC18b] Google LLC. Tensorflow, December 2018.
- [LLC18c] Google LLC. Tensorflow detection model zoo, December 2018.
- [Mis19] Mayank Mishra. Convolutional neural networks, explained, March 2019.
- [MP43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.
- [oCSD18] University of California San Diego. Why are neuron axons long and spindly? study shows they’re optimizing signaling efficiency, July 2018.
- [PWCL16] Min Peng, Chongyang Wang, Tong Chen, and Guangyuan Liu. Nirfacenet: A convolutional neural network for near-infrared face identification. *Information*, 7(4), 2016.
- [SCL12] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3288–3291. IEEE, 2012.
- [ST13] Robert F Schmidt and Gerhard Thews. *Physiologie des Menschen*. Springer-Verlag, 2013.
- [sW18] Computer science Wiki. Max-pooling / pooling, February 2018.
- [Tzu19] Darrenl Tzutalin. labelimg, Juny 2019.
- [up19] uni protokolle. Yuv-farbmodell, April 2019.
- [Wik19] Wikipedia. Künstliches neuron, April 2019.

Anhang

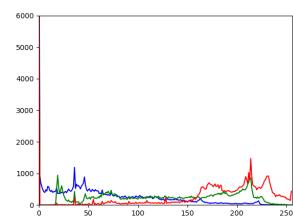
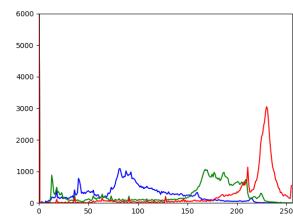
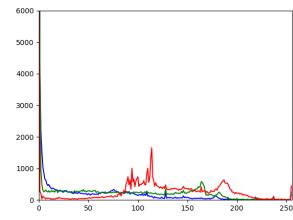


Abbildung .2: Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch den Histogramm-Ausgleich

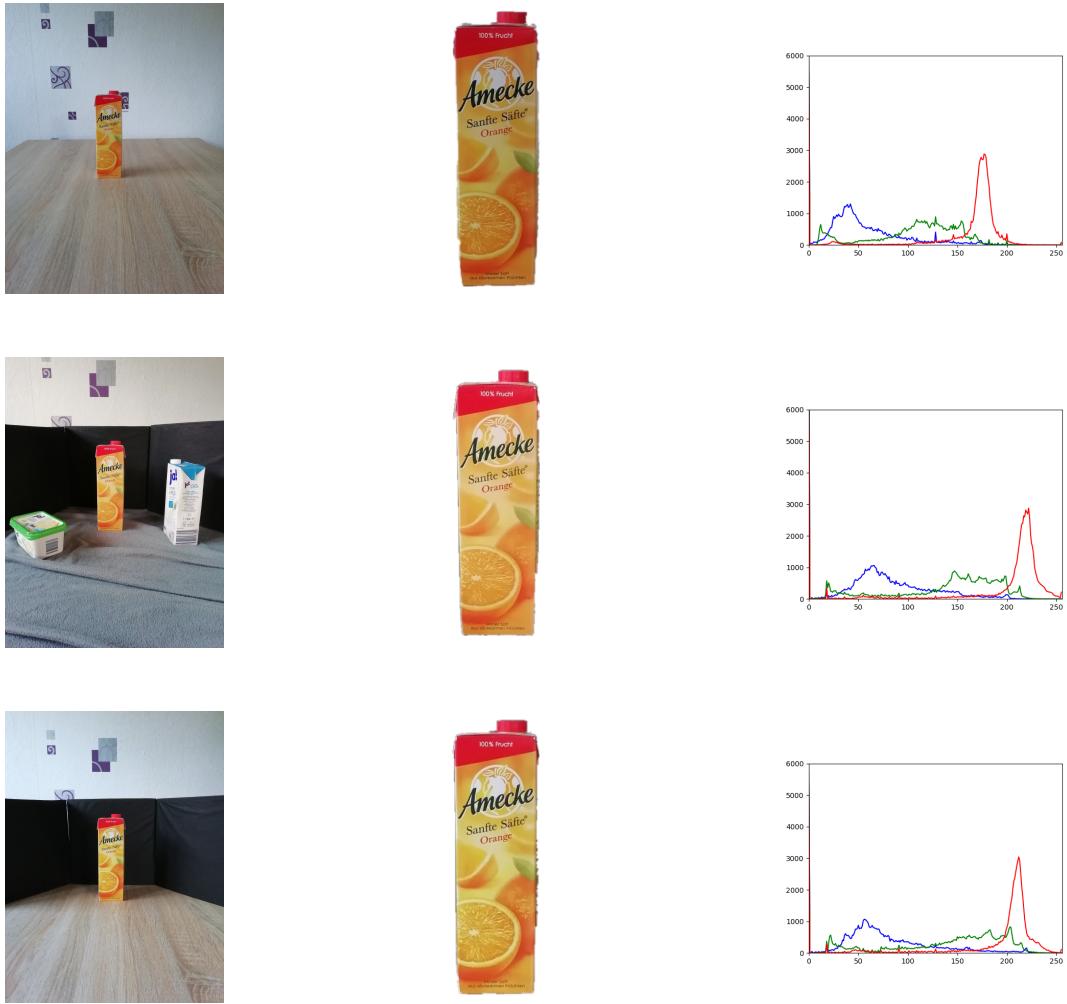


Abbildung .3: Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch den Gray-World-Algorithmus

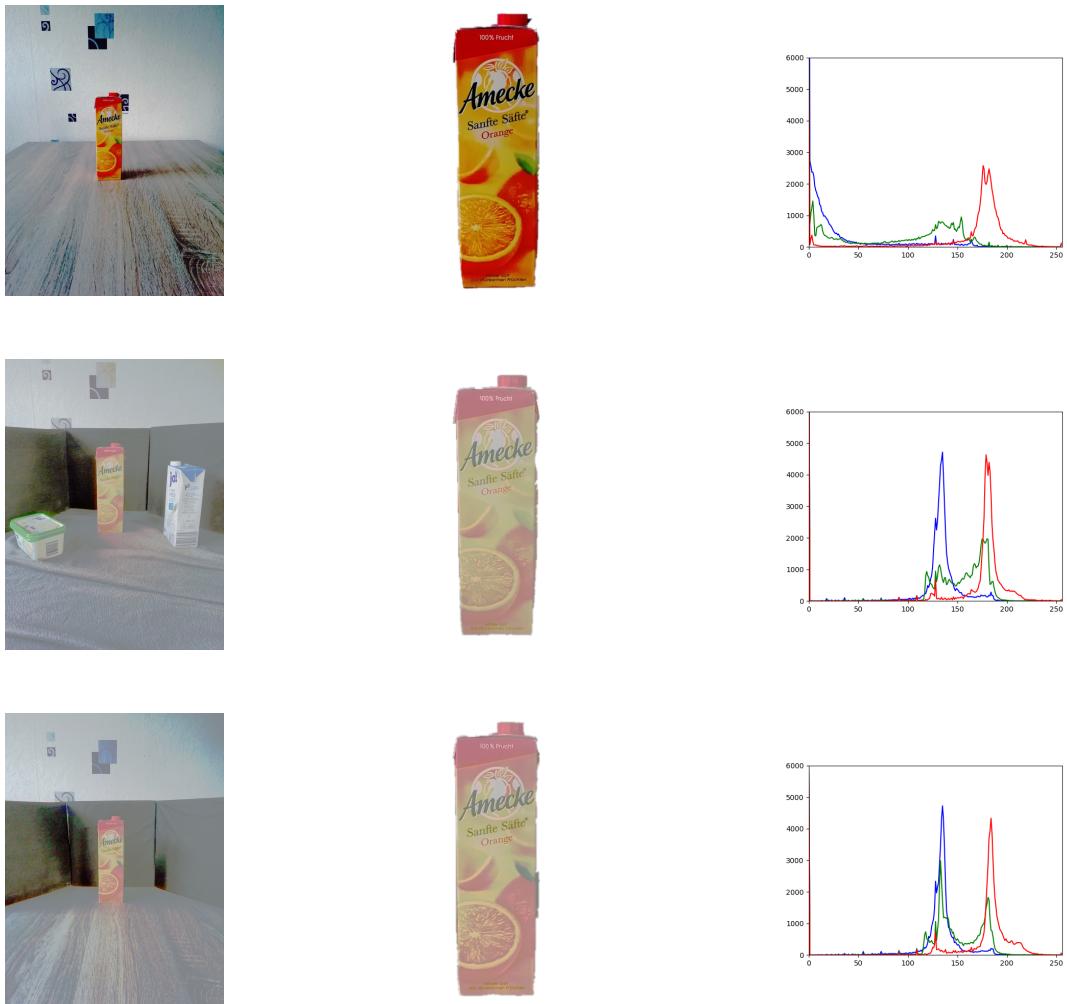


Abbildung .4: Histogramm des segmentierten Objektes aus dem Nahrungsmittel-Datensatz. Normalisiert durch die Histogramm-Spezifikation

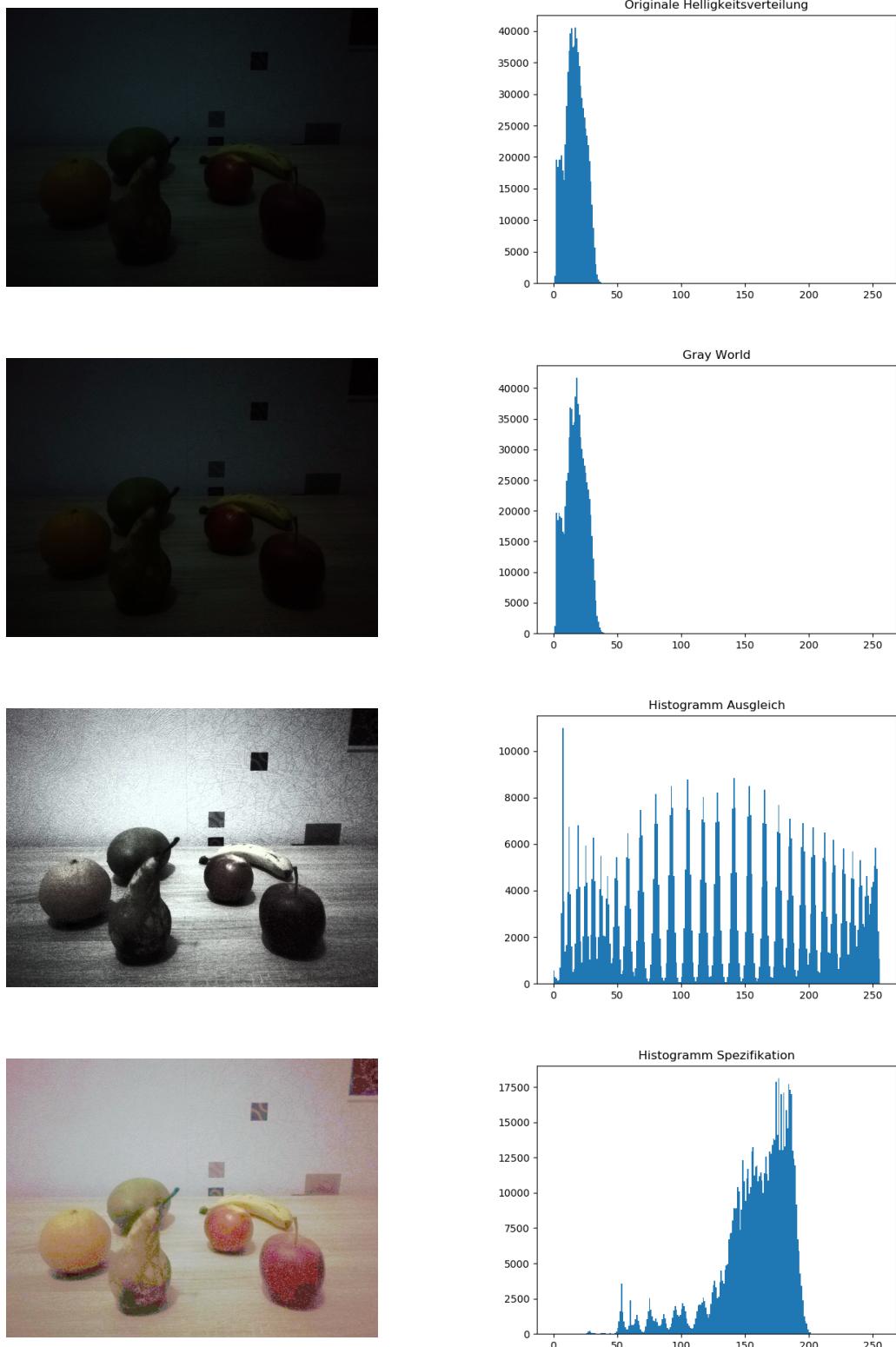


Abbildung .5: Helligkeitsverteilung und der Einfluss der Normalisierungs-Algorithmen auf das Testbild

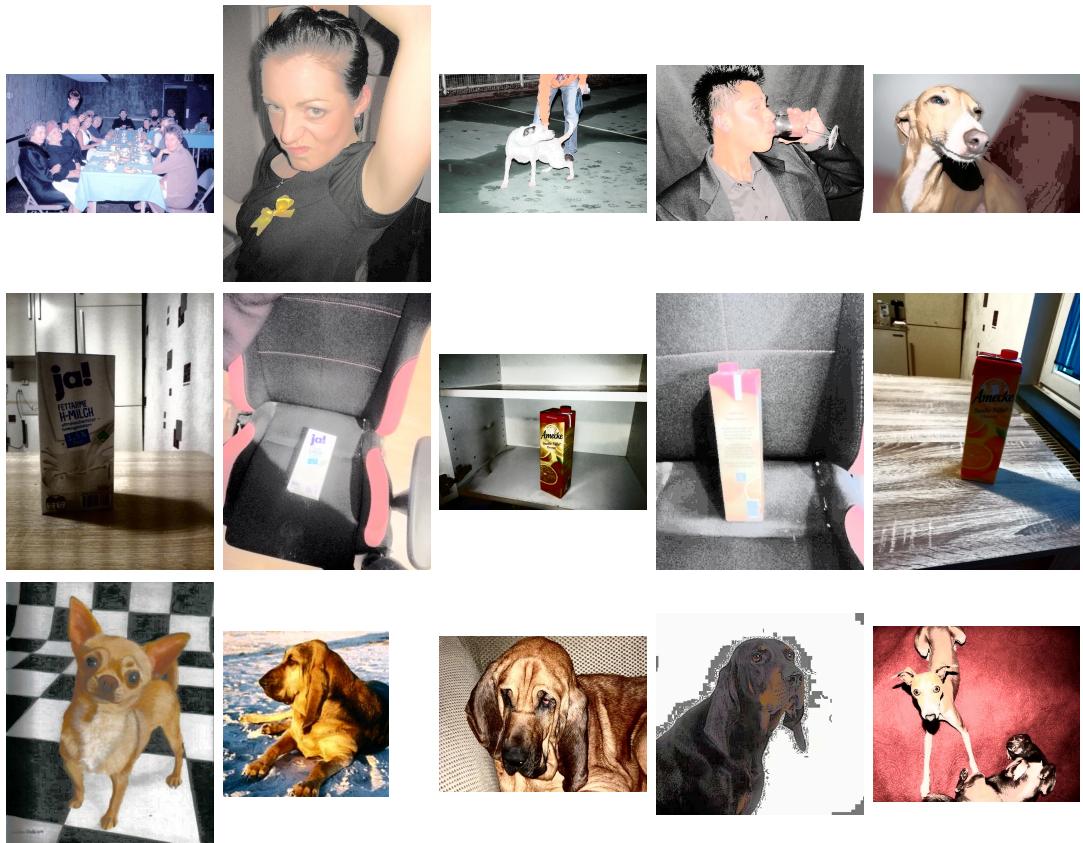


Abbildung .6: Auftretende Farbanomalien in den drei Datensätzen durch den Histogramm-Ausgleich. Oben: PascalVOC-Datensatz, Mitte: Nahrungsmittel-Datensatz, Unten: Stanford Dog-Datensatz

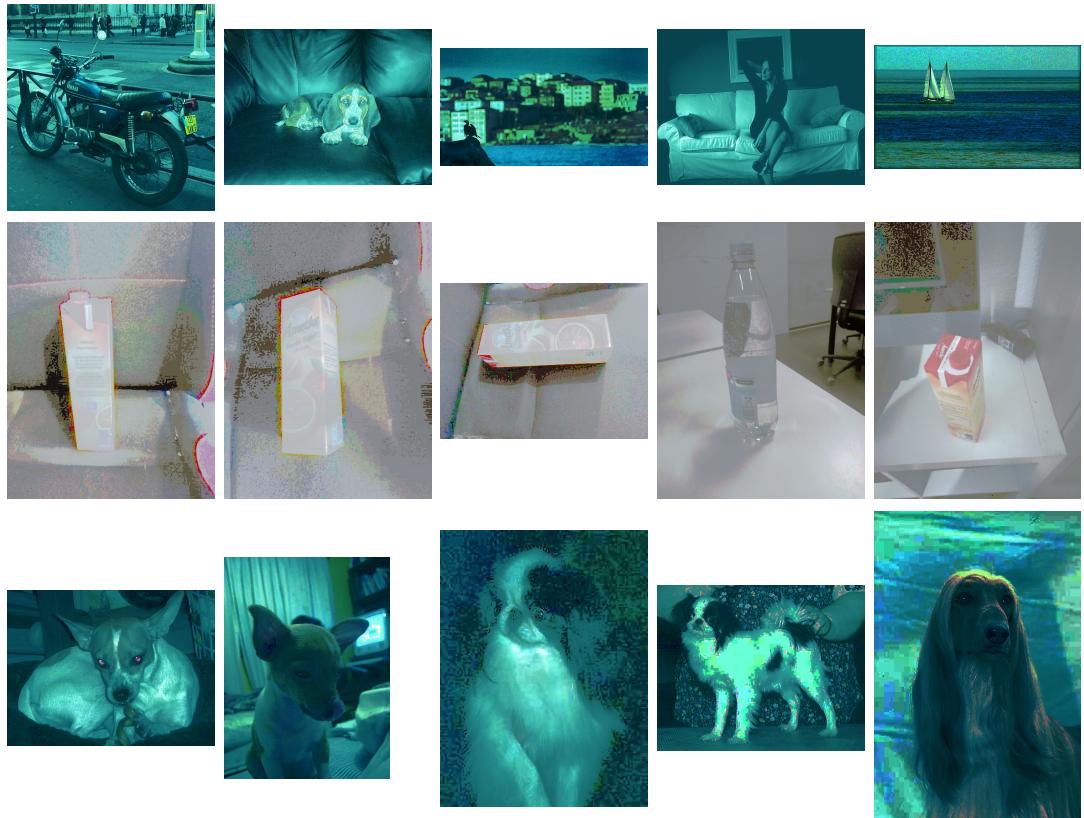


Abbildung .7: Auftretende Farbanomalien in den drei Datensätzen durch die Histogramm-Spezifikation. Oben: PascalVOC-Datensatz, Mitte: Nahrungsmittel-Datensatz, Unten: Stanford Dog-Datensatz

Eidesstattliche Erklärung

Ich versichere, dass ich die Bachelorarbeit selbständig 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