

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Motivation . . . . .	2
1.2	Ziel der Arbeit . . . . .	3
1.3	Aufgabenstellung . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>5</b>
2.1	Maschinelles Lernen . . . . .	5
2.1.1	Definition des Maschinellen Lernens . . . . .	5
2.1.2	Arten des Maschinellen Lernens . . . . .	6
2.2	Neural Networks . . . . .	8
2.2.1	Ursprünge von Neural Networks . . . . .	8
2.2.2	Perceptron Multi-layer Networks . . . . .	10
2.2.3	Deep Learning . . . . .	11
2.3	Convolutional Neural Networks . . . . .	12
2.3.1	Definition . . . . .	12
2.3.2	Operationen . . . . .	14
2.4	Computer Vision . . . . .	16
2.4.1	Definition . . . . .	16
2.4.2	Objekterkennung . . . . .	17
2.5	Deployment Pipeline . . . . .	21
2.5.1	Definition . . . . .	22
2.5.2	Grundlagen der Deployment Pipeline . . . . .	23
2.5.3	Anforderungen der Arbeit . . . . .	23
<b>3</b>	<b>State of the Art</b>	<b>26</b>
3.1	Machine Learning . . . . .	26
3.2	Deep Learning . . . . .	27
3.3	Object detection . . . . .	29
<b>4</b>	<b>Implementierung</b>	<b>32</b>
4.1	Projektarchitektur . . . . .	32
4.2	Software und Hardware . . . . .	34
4.2.1	Graphics Processing Unit :GPU . . . . .	34

4.2.2	Tensorflow	35
4.3	Datasets	37
4.4	Training	40
4.5	Evaluation	42
4.6	Deployment in Production	44
4.6.1	Mobile Anwendung	44
4.6.2	Web Anwendung	47
4.7	Zusammenfassung	49
<b>5</b>	<b>Analyse der Ergebnisse und Diskussion</b>	<b>50</b>
5.1	Analyse der Ergebnisse	50
5.1.1	Ergebnisse des Trainings	50
5.1.2	Ergebnisse der Evaluation	52
5.1.3	Zusammenfassung	56
5.2	Diskussion	56
5.2.1	Datensätze	57
5.2.2	Mehrere Erkennungen	57
<b>6</b>	<b>Zusammenfassung</b>	<b>59</b>
<b>Quellenverzeichnis</b>		<b>61</b>
Literatur		61
Online-Quellen		65

# Abbildungsverzeichnis

2.1	Biological Neuron gegen Artificial Neural Network [46] . . . . .	9
2.2	multi-layer neural network [47] . . . . .	10
2.3	Architektur des Convolutional Neural Network [48] . . . . .	12
2.4	Architektur von R-CNN[31] . . . . .	18
2.5	Architektur von Fast R-CNN[30] . . . . .	19
2.6	Deployment Pipeline des Projekts . . . . .	24
3.1	Basisblockdiagramm eines typischen Objekterkennungs [9] . . . . .	29
4.1	Ablaufprozess des Projekts . . . . .	33
4.2	Vortrainierte Tensorflow-Modell Ergebnis[51] . . . . .	37
4.3	Annotiertes Bild . . . . .	38
4.4	XML-Dateistruktur . . . . .	39
4.5	Confusion-matrix . . . . .	43
4.6	Android-App-Architektur[50] . . . . .	45
4.7	Android-App:NIO Objekterkennung . . . . .	45
4.8	Android-App:IO Objekterkennung . . . . .	46
4.9	Screenshot der Anwendung: Bild aufnehmen von Kamera . . . . .	48
4.10	Screenshot der Anwendung: Bild aufnehmen von webcam . . . . .	48
5.1	Gesamtverlust beim Trainingsmodell . . . . .	51
5.2	Präzision beim Trainingsmodell . . . . .	52
5.3	Confusion Matrix der Evaluationsschritte . . . . .	53
5.4	Confusion Matrix der neuen Evaluationsschritte . . . . .	55
5.5	Beispiele für Mehrfachobjekt-Erkennungen . . . . .	58

# Kapitel 1

## Einleitung

Künstliche Intelligenz (KI) ist das allgemeine wissenschaftliche und technische Prinzip, das sich mit der Erstellung intelligenter Maschinen beschäftigt. KI erlangte vor kurzem eine große öffentliche Aufmerksamkeit. Die Forschung in KI begann in den 1950er Jahren. Einer der größten Meilensteine war IBMs Supercomputer Deep Blue. Eine Untergruppe in KI, maschinelle Lernalgorithmen lernen aus großen Mengen statistischer Daten und treffen Vorhersagen über unbekannte Daten.

Innerhalb des maschinellen Lernens ist Deep Learning ein neuer Bereich des Algorithmus, der aus mehreren nichtlineare-schichten(layers) besteht. In den letzten Jahren haben Deep-Learning-Algorithmen zu neuen Durchbrüchen beigetragen. Die Prinzipien des Deep Learning, insbesondere der Convolutional Neural Networks (CNNs), sind jedoch kein neues Konzept. Die allgemeinen Ideen existieren seit den 1990er Jahren [29, 44]. Seitdem waren CNNs sehr beliebt, aber ihre Bedeutung nahm mit der Einführung von Support-Vektor-Maschinen (SVMs) ab. Durch leistungsfähigere Grafikprozessoren (GPUs), größere Datenmengen und bessere Algorithmen. In den letzten Jahren ist Deep Learning wieder sehr populär geworden und die Erfolge des Deep Learning haben sich beschleunigt.

Convolutional Neural Networks sind mehrstufige Architekturen von nichtlineare Schichten, die in großen Datensätzen trainiert werden können. Sie können für viele Objekterkennungen und Computer-Vision-Aufgaben verwendet werden.

Computer Vision ist die Fähigkeit von Maschinen zu sehen und zu verstehen, was in ihrer Umgebung ist. Dieses Feld enthält Methoden zum Erfassen, Verarbeiten und Analysieren von Bildern, um wichtige Informationen extrahieren zu

können, die von künstlichen Systemen verwendet werden. In letzter Zeit werden in der Computervision viele Forschungsarbeiten durchgeführt, insbesondere in den wichtigsten Subdomänen wie Objekterkennung, Bewegungsanalyse oder Szenenrekonstruktion. Die Objekterkennung beschreibt die Aufgabe, ein Bild in bestimmter Weise zu bearbeiten, um Objekte zu lokalisieren und zu klassifizieren. Es gibt eine große Vielfalt von Objekterkennungsansätzen, aber das allgemeine Konzept bleibt dasselbe. Ein Objekterkennungssystem verwendet Trainingsdatensätze, die Bilder mit bekannten und etikettierten Objekten enthalten, und extrahiert verschiedene Arten von Information basierend auf dem gewählten Algorithmus. Im Allgemeinen werden für jedes neue Bild die gleichen Informationen gesammelt und mit dem Trainingsdatensatz verglichen, um die am besten geeignete Klassifikation zu finden.

## 1.1 Motivation

Vision ist die fortschrittlichste unserer Sinne. Daher ist es nicht überraschend, dass Bilder eine wichtige Rolle in der menschlichen Wahrnehmung spielen. Dies ist analog zur maschinellen Bildverarbeitung wie der Formerkennung, die heutzutage ein wichtiges Gebiet ist. Machine Vision ist eine der Anwendungen von Computer Vision für Industrie und Fertigung, während Computer Vision hauptsächlich auf maschinelle Bildverarbeitung ausgerichtet ist.

Seit einigen Jahrzehnten haben Computerwissenschaftler und Ingenieure Kameras und einfache Bildinterpretationsmethoden an einen Computer angeschlossen, um der Maschine eine Sicht zu geben. Es wurde viel Interesse für Objekterkennung, Objektkategorisierung gezeigt. Objekterkennung befasst sich mit dem Training des Computers, um ein bestimmtes Objekt aus verschiedenen Perspektiven, unter verschiedenen Lichtbedingungen und mit verschiedenen Hintergründen zu identifizieren. Objekterkennung befasst sich mit der Identifizierung der Anwesenheit von verschiedenen einzelnen Objekten in einem Bild und die Objektkategorisierung befasst sich mit der Erkennung von Objekten, die zu verschiedenen Kategorien gehören.

Große Erfolge wurden in einer kontrollierten Umgebung für das Problem der Objekterkennung erreicht. Viele Anwendungen, die Objekterkennung verwenden, können im heutzutage gefunden werden. Z.B: Gesichts- oder Handschrifterkennung.

## 1.2 Ziel der Arbeit

Maschinelles Lernen ermöglicht Lösungen, die in der Vergangenheit als zu komplex angesehen wurden. Beispiele sind die Erkennung falsch montierter Teile in einer komplexen Produktionsumgebung. Maschinelles Lernen beruht auf einem kontinuierlichen Lernzyklus. Um dies in der Produktion nutzbar zu machen, müssen die meist manuellen Prozesse industrialisiert werden. Machine vision ist eine gute Möglichkeit, maschinelle Lernalgorithmen zu üben. Machine-Vision-Anwendungen können die Produktqualität erheblich verbessern, indem sie Probleme erkennen, wenn sie auftreten. Objekterkennung ist eines der klassischen Probleme des Computer Vision und wird oft als eine schwierige Aufgabe beschrieben. Das Verfahren erfordert normalerweise das Vorhandensein eines Datensatzes, der mit Ortsinformationen der Objekte versehen ist, die in Form von Begrenzungsrahmen um die Objekte herum vorliegen.

Das Ziel dieser Arbeit ist somit die Entwicklung und Implementierung einer Industrialisierung von maschinellem Lernen in der Produktion mit Verwendung einer Deployment Pipeline, um Objekterkennung in einer schwach überwachten Art und Weise durchzuführen, d. h. Unter Verwendung eines teilweise annotierten Datensatzes. Der Datensatz liefert Informationen darüber, welche Objekte im Bild vorhanden sind. Die Arbeit ist der Entwurf einer Lösung, die eine Deployment-Pipeline für eine Convolutional Neural Network (CNN)-basierte Architektur definiert und implementiert. Diese Pipeline wird automatisierte Verlauf beinhalten, wie die maschinelle Bildverarbeitung auf Bildern basiert, die große Mengen an Bildern verarbeiten und passende Bilder zu einem Experten vorschlagen, ist dies ebenfalls Teil der Lösung.

## 1.3 Aufgabenstellung

Der Einarbeitungsphase liegen drei Hauptaufgaben zu Grunde. Zum einen ist es wichtig, einen Überblick über das gesamte Themenfeld zu erhalten. Die Schwerpunkte liegen dabei bei der generellen Einarbeitung in das Thema „Industrialisierung des maschinellen Lernens“ und „Deployment-Pipeline“. Davon ausgehend soll eine Recherche zum Thema „Objekterkennungs-verfahren“ folgen.

Zum anderen besteht ein wichtigerer Schwerpunkt der Arbeit darin, eine Analyse des aktuellen Prozesses der Softwareentwicklung vorzunehmen. Im dritten Teil

der Arbeit werden der entwickelte Prozess für eine Deployment-Pipeline sowie das Testsystem vorgestellt und Implementierungsdetails begründet. Anschließend folgen die Evaluation des entwickelten Systems und die Zusammenfassung der Arbeit.

Die Struktur dieser Arbeit wird im Folgenden beschrieben. **Kapitel 1** enthält das Ziel und die Aufgabenstellung der Arbeit. **Kapitel 2** beginnt mit einer kurzen Einführung in maschinelles Lernen und neuronale Netze. Als nächstes wird Convolutional neural networks einführt. Das Kapitel wird beendet, indem Computer Vision und Objekterkennung diskutiert werden, wie Convolutional neural networks für die Objekterkennung genutzt werden können und überarbeiten die relevanten Methoden. Das **Kapitel 3** beschreibt Arbeiten, die thematisch verwandt zur vorliegenden Arbeit sind. **Kapitel 4** enthält eine Einführung in die Grundlagen von deployment pipeline. In **Kapitel 5** werden der entwickelte Prozess sowie die praktische Umsetzung der Experimente, indem die benötigte Software und Hardware und deren Einsatz diskutiert werden. in **Kapitel 6** wird die Ergebnisse weiter analysiert und ausgewertet. Anschließend enthält **Kapitel 7** die Zusammenfassung der Arbeit.

# Kapitel 2

## Grundlagen

In diesem Kapitel bietet die theoretischen Grundlagen, die benötigt wird, um die durchgeführten Arbeiten vollständig zu verstehen.

Zuerst wird das maschinelle Lernen und die Hintergründe erklärt. Zunächst werden die relevanten Details des maschinellen Lernens, Neural Network und Convolutional Neural Network behandelt. Danach wird das Computer Vision und Objekterkennung Verfahren erklärt. Anschliessend wird einen Überblick über die Deployment Pipeline gegeben.

### 2.1 Maschinelles Lernen

#### 2.1.1 Definition des Maschinellen Lernens

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz (KI). Im Allgemeinen besteht das Ziel des Maschinellen Lernens darin, die Datenstruktur zu verstehen und in Modelle zu integrieren, die von jedem verstanden und genutzt werden können. Maschinelles Lernen ist das Gebiet der wissenschaftlichen Forschung, das sich auf Induktionsalgorithmen und andere Algorithmen konzentriert. Allgemein um maschinelles Lernen zu beschreiben, wird zwei Zitate verwendet. Der Begriff "Lernen" kann weiter verdeutlicht werden: "Im weitesten Sinne nutzt jede Methode, die Informationen aus Trainingsbeispielen enthält, das Lernen[32].

Maschinelles Lernen bedeutet, Computer nicht nur für Berechnungen und Datenabfragen zu verwenden, aber diese beiden Fähigkeiten eines Computersystems

zu kombinieren, um es zu veranlassen, nach vorher beobachteten Umständen und früheren Handlungen oder Reaktionen zu lernen und vernünftige Entscheidungen zu treffen, und nicht nur nach einem festen Plan handeln. Dies ist besonders wichtig, wenn die vom System auszuführende Funktion zu kompliziert ist, um im Code beschrieben zu werden, oder die Zuordnungen von Bedingung und Aktion unbekannt sind.

Maschinelles Lernen hat sich als ein nützliches Werkzeug zur Modellierung von Problemen erwiesen, die ansonsten nur schwer exakt zu formulieren sind. Klassische Computerprogramme werden explizit per Hand programmiert, um eine Aufgabe auszuführen. Beim maschinellen Lernen wird ein Teil des menschlichen Beitrags durch einen Lernalgorithmus ersetzt [17]. Da die Verfügbarkeit von Rechenkapazität und Daten zugenommen hat, wurde das maschinelle Lernen im Laufe der Jahre immer praktischer und fast allgegenwärtig.

Maschinelles Lernen ist ein Bereich in ständiger Entwicklung. Aus diesem Grund müssen bestimmte Überlegungen berücksichtigt werden, wenn mit maschinellen Lerntechnologien gearbeitet wird oder Auswirkungen von maschinellen Lernprozessen analysiert werden. Deswegen stützt dieses Kapitel sich auf, um die Arten von Maschinelles Lernen zu erklären

### 2.1.2 Arten des Maschinellen Lernens

In diesem Schritt werden die verschiedenen Arten des Maschinellen Lernens erklärt. Diese Informationen basieren auf [3, 12].

#### Überwachtes Lernen (engl. Supervised learning)

Überwachtes Lernen ist einfach eine Formalisierung der Idee des Lernens von überwachten Klassen. Beim Supervised learning werden dem Lernenden zwei Sätze von Daten, ein Trainingssatz und ein Testset zur Verfügung gestellt. Der Lernende soll aus einem Satz markierter Beispiele im Trainingssatz so "lernen", dass er unbeschriftete Beispiele im Testsatz mit der höchstmöglichen Genauigkeit identifizieren kann. Das heißt, das Ziel des Lernenden ist es, eine Regel, ein Programm oder eine Prozedur zu entwickeln, die neue Beispiele einteilt, indem Beispiele analysiert werden, denen bereits eine Klassenbezeichnung zugewiesen wurde. Zum Beispiel

könnte ein Trainingssatz aus Bildern verschiedener Fruchtarten bestehen, wobei dem Lernenden die Identität der Frucht in jedem Bild gegeben wird. Das Ziel besteht darin, dass der Lernende eine Regel entwickelt, die die Elemente im Testset identifizieren kann.

Ein typischer Weg, maschinelles Lernen zu nutzen, ist das überwachte Lernen. Ein Lernalgorithmus zeigt mehrere Beispiele, die von Menschen kommentiert oder etikettiert wurden. Zum Beispiel werden in dieser Arbeit Objekterkennung Problem Trainingsbilder verwendet, bei denen Menschen die Orte und Klassen relevanter Objekte markiert haben. Nach dem Lernen ist der Algorithmus in der Lage, die Anmerkungen oder Markierungen von zuvor ungesehenen Daten vorherzusagen. **Classification** and **regression** sind die wichtigsten Aufgabenarten

- **Classification:** Bei der Classification versucht der Algorithmus, die korrekte Klasse eines neuen Datenstücks basierend auf den Trainingsdaten vorherzusagen.
- **regression :** In der Regression versucht der Algorithmus anstelle von diskreten Klassen eine kontinuierliche Ausgabe vorherzusagen.

### Unüberwachtes Lernen (engl. Unsupervised learning)

Das Ziel dieses Verfahrens ist es, dass der Computer lernt, etwas zu tun, was die Benutzer ihm nicht sagen. Es kann so einfach sein wie das Auffinden von Mustern, die in einem Datensatz verborgen sind. Es kann aber auch ein Feature Lernziele haben, mit dem die intelligente Maschine automatisch die Repräsentationen erkennen kann, die zur Klassifizierung der Rohdaten erforderlich sind.

Es gibt zwei Ansätze für unüberwachtes Lernen. Der erste Ansatz besteht darin, den Agenten nicht durch explizite Kategorisierungen zu unterrichten, sondern durch eine Art Belohnungssystem, um den Erfolg anzuzeigen. Diese Art von Training wird in der Regel in den Entscheidungsproblemrahmen passen, denn das Ziel ist nicht, eine Klassifizierung zu erstellen, sondern Entscheidungen zu treffen, die die Belohnungen maximieren.

Beim unsupervised Learning sind die Daten nicht markiert, so dass der Lernalgorithmus die gemeinsamen Punkte aus seinen Eingabedaten allein herausfinden muss. Unbeschriftete Daten sind häufiger als markierte Daten.

Beim unsupervised learning versucht der Algorithmus, nützliche Eigenschaften der Daten zu lernen, ohne dass ein menschlicher Lehrer angibt, was die korrekte Ausgabe sein soll. Klassisches Beispiel für unüberwachtes Lernen ist das **Clustering**. In noch jüngerer Zeit, insbesondere mit dem Aufkommen von Deep-Learning-Technologien, ist die unsupervised Vorverarbeitung zu einem beliebten Werkzeug bei überwachten Lernaufgaben geworden, um nützliche Darstellungen der Daten zu entdecken.

### Verstärkungslernen (engl. Reinforcement Learning)

Verstärkendes Lernen unterscheidet sich von überwachtem Lernen und unüberwachtem Lernen, dabei geht es darum, Strukturen zu finden, die in Sammlungen nicht beschrifteter Daten verborgen sind. Verstärkung Lernen liegt zwischen diesen beiden Ansätzen. Ohne externen Supervisor erhält der Agent eine Rückmeldung über die Qualität seiner Aktionen. Verstärkendes Lernen basiert also auf der Interaktion eines Agenten, der eine Aktion ausführt, und seiner Umgebung, die positives oder negatives Feedback gibt.

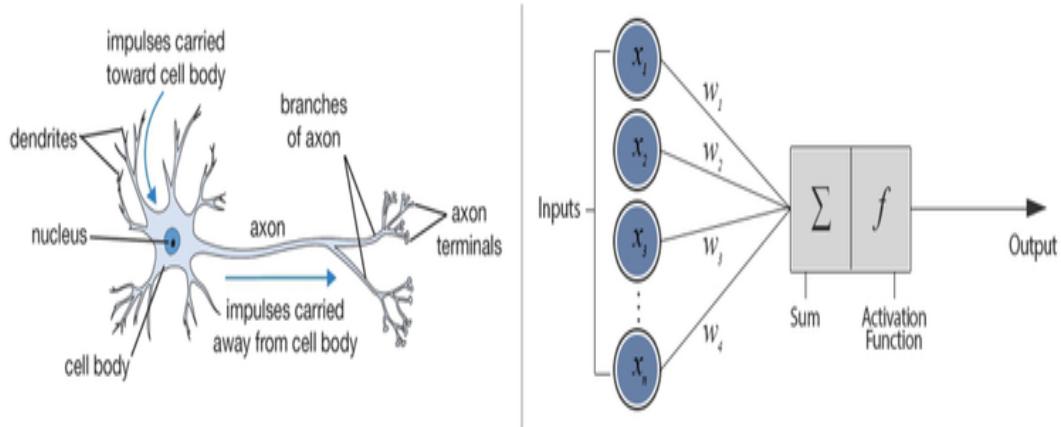
## 2.2 Neural Networks

Neural Networks sind eine beliebte Art von maschinellem Lernmodell. Ein spezieller Fall eines neuronalen Netzwerks, Convolutional Neural Network (CNN) genannt, ist der Schwerpunkt dieser Arbeit. Bevor CNN, wird in diesem Abschnitt diskutiert, wie reguläre neuronale Netzwerke funktionieren. Diese Informationen basieren auf [17, 24, 33].

### 2.2.1 Ursprünge von Neural Networks

Neuronale Netzwerke wurden ursprünglich künstliche neurale Netzwerke genannt, weil sie entwickelt wurden, um die neurale Funktion des menschlichen Gehirns nachzuahmen. Die ersten Formulierungen des neuronalen Netzes stammen aus dem Jahr 1943, in der Arbeit von McCulloch und Pitts. Die Idee ist, die Funktionsweise eines menschlichen Neurons zu reproduzieren [41] (wie in Abbildung 2.1 gezeigt).

## Biological Neuron versus Artificial Neural Network



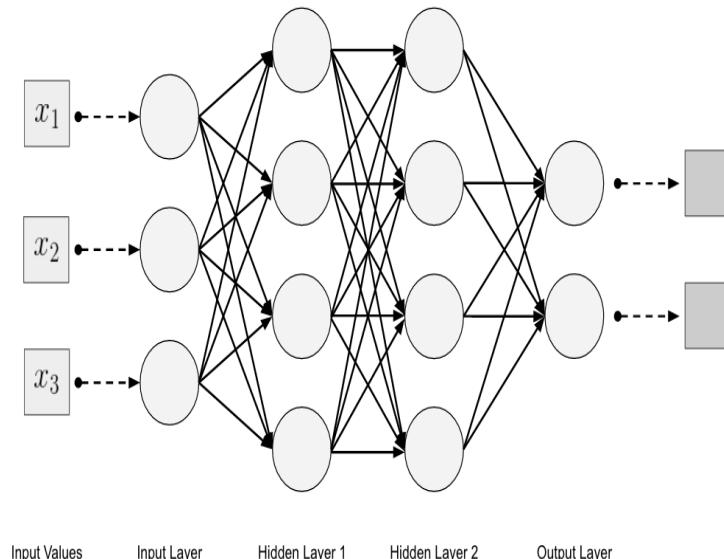
**Abbildung 2.1:** Biological Neuron gegen Artificial Neural Network [46]

Artificial neural networks (ANN) sind eine Klasse von maschinellen Lernalgorithmen, die bei der Behebung vieler hartnäckiger Probleme in der Informatik vielversprechend sind. ANN inspirieren von der Informationsverarbeitungsstruktur des Gehirns, bestehen aus kleinen miteinander verbundenen Computereinheiten, die Neuronen genannt werden. Neuronen sind in der Regel in Schichten segmentiert: Informationen werden über eine Eingabeschicht an das Netzwerk weitergeleitet, über mehrere verdeckte Ebenen verteilt und transformiert, und Werte, die ein abstraktes Merkmal der Eingabeinformationen beschreiben, werden von der Ausgabeschicht nach dem Training gegeben. Die Transformationen werden durch Gewichte und Verzerrungen parametrisiert, die als trainierbare Parameter in dem Sinne bekannt sind, dass sie optimiert sind, um einige Aspekte der Eingabedaten am besten vorherzusagen. Dies geschieht durch Quantifizieren der Ähnlichkeit zwischen den Zielmarkierungen und den durch das ANN abgeleiteten Ergebnissen durch eine differenzierbare Verlustfunktion. Mit Infinitesimalrechnung und einigen Optimierungsstrategien, die erlernbaren Parameter werden mit dem Ziel modifiziert, die Ähnlichkeit der Markierungen und der durch das ANN erzeugten Voraussagen zu verbessern [5].

Ein künstliches Neuron, das auf dem McCulloch-Pitts-Modell basiert, ist in Abbildung 2.1 dargestellt. Das Neuron  $k$  empfängt  $m$  input  $X_j$ . Das Neuron hat auch Gewichtsparameter(weight)  $W_j$ . Die Gewichtsparameter enthalten oft einen Bias-Term, der eine passende Dummy-input mit einem festen Wert von 1 hat. Die Eingaben und Gewichte werden linear kombiniert und summiert. Die Summe wird dann einer Aktivierungsfunktion zugeführt, die den Ausgang erzeugt. Das Neuron wird trainiert, indem die Gewichte sorgfältig ausgewählt werden, um eine gewünschte Ausgabe für jede Eingabe zu erzeugen.

### 2.2.2 Perceptron Multi-layer Networks

Eine neue Art von Netzwerk, um die Grenzen des einfachen Perzeptrons zu überwinden, wurde in den 80er Jahren vorgeschlagen: mehrschichtiges Perzepron (Multi Layer Perceptron) (MLP). In Wirklichkeit stammt diese Formulierung aus dem Jahr 1969 [2], wurde aber tatsächlich seit 1986 durch den Artikel von Rumelhart und Hinton verwendet [34].



**Abbildung 2.2:** multi-layer neural network [47]

Ein neuronales Netzwerk ist eine Kombination künstlicher Neuronen. Die Neuronen sind typischerweise in Schichten (Layers) gruppiert. In einem vollständig verbundenen mehrschichtigen Netzwerk, (wie in Abbildung 2.2 gezeigt), wird jeder Ausgang einer Neuronenschicht als Eingang für jedes Neuron der nächsten Schicht zugeführt. Daher verarbeiten einige Schichten die ursprünglichen Eingabedaten, während einige Prozessdaten von anderen Neuronen empfangen werden. Jedes Neuron hat eine Anzahl von Gewichten, die der Anzahl von Neuronen in der vorherigen Schicht entspricht.

Ein Netzwerk mit mehreren Schichten umfasst typischerweise drei Arten von Schichten: eine Eingabe Schicht, eine oder mehrere ausgeblendete Schichten und eine Ausgabeschicht. Die Eingabeschicht leitet Daten einfach weiter, ohne sie zu modifizieren. Der größte Teil der Berechnung findet in den versteckten Schichten(hidden layers) statt. Die Ausgabeschicht konvertiert die Aktivierungen der verborgenen Schicht in eine Ausgabe, beispielsweise eine Klassifizierung.

### 2.2.3 Deep Learning

Moderne neuronale Netze werden oft als Deep neuronale Netze bezeichnet. Obwohl seit den 1980er Jahren mehrschichtige neuronale Netze existieren, verhinderten mehrere Gründe das effektive Training von Netzwerken mit mehreren versteckten Schichten.

In den letzten zehn Jahren haben neuronale Netze eine Renaissance erlebt, hauptsächlich aufgrund der Verfügbarkeit von leistungsfähigeren Computern und größeren Datensätzen. Anfang der 2000er Jahre wurde entdeckt, dass neuronale Netzwerke effizient mit Grafikverarbeitungseinheiten(GPU) trainiert werden können. GPUs sind für diese Aufgabe effizienter als herkömmliche CPUs und bieten eine relativ günstige Alternative zu spezieller Hardware. Heutzutage verwenden Forscher typischerweise High-End-Grafikkarten für Verbraucher wie NVIDIA. Mit deep learning, gibt es weniger Bedarf für hand-tuned Machine Learning-Lösungen, die zuvor verwendet wurden. Ein klassisches Mustererkennungssystem enthält beispielsweise eine hand-tuned Merkmalserfassungsphase vor einer maschinellen Lernphase. Das Deep Learning Equivalent besteht aus einem einzigen neuronalen Netzwerk. Die unteren Schichten des neuronalen Netzes lernen die grund-

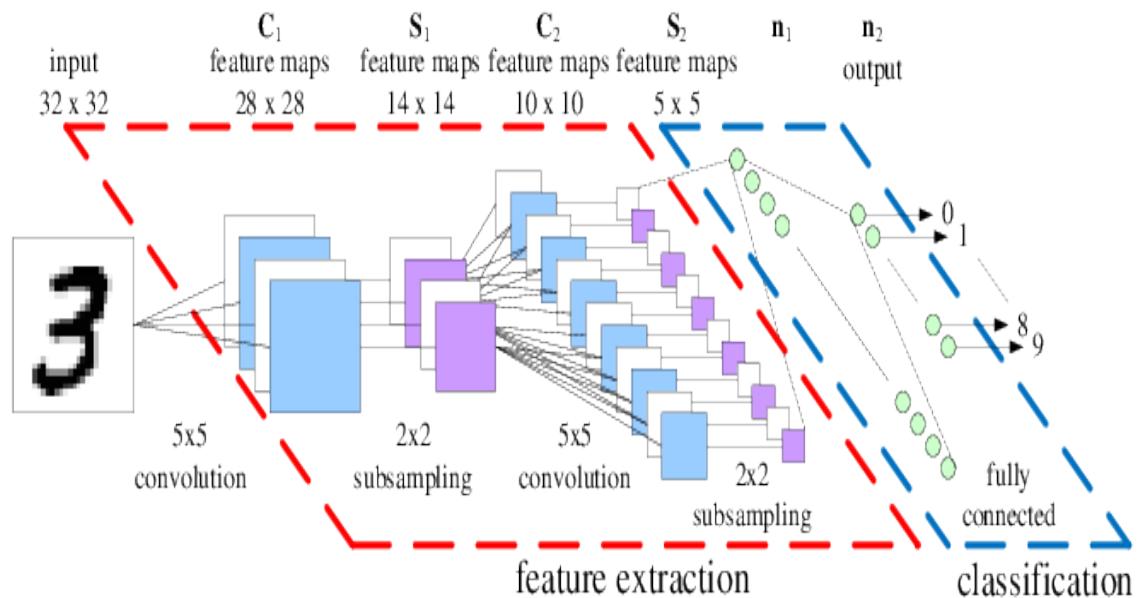
legenden Merkmale zu erkennen, die dann an höhere Schichten des Netzwerks weitergeleitet werden.

## 2.3 Convolutional Neural Networks

Dieser Abschnitt erklärt der zugrunde liegenden Konzepte und die Architektur der Convolutional neural networks. Diese Informationen basieren auf [17, 43].

### 2.3.1 Definition

Während CNN in den letzten Jahren eine bedeutende Popularität erlangt haben, haben sie eine Geschichte, die bis in die 1980er Jahre zurückreicht. Diese neuronalen Netzwerkmodelle wurden speziell für die effiziente Verarbeitung von Bildern entwickelt. Sie tun dies, indem sie eine große Anzahl mathematischer Operationen kombinieren.



**Abbildung 2.3:** Architektur des Convolutional Neural Network [48]

Die **Abbildung 2.3** zeigt die Architektur des Convolutional Neural Network. Ein typisches CNN besteht aus mehreren nichtlinearen Schichten. Die Eingabe ist

ein Bild und die Ausgabe ist ein Vektor. Der Ausgabevektor enthält die Wahrscheinlichkeit, dass das Bild zu jeder Klasse gehört. Dieses Netzwerk hat vier verschiedene Arten von Schichten, die wie folgt benannt sind: 'convolution', 'subsampling', and 'fully connected'.

Die beschriebenen Probleme führten zur Schaffung eines neuen Modells mit einer stärkeren Verschiebungsinvarianz, die auf Hierarchien lokaler Merkmale reagiert. Dieses neue Netzwerk wurde Convolutional Neural Network (CNN) genannt. CNNs sind eine Untergruppe von Deep-Learning-Algorithmen, die sich auf Computer Vision und Bildverarbeitungsaufgaben konzentrieren. CNNs sind lose davon inspiriert, wie das Gehirn visuelle Informationen verarbeitet. Das erste CNN wurde in Fukushimas Neocognitron eingeführt[22]. CNNs wurden später durch LeNet-5 von Yann LeCun verbessert, das gradientenbasierte stochastische Gradienten zur Dokumentenerkennung anwendete und sehr erfolgreich für Handschrifterkennungsaufgaben war[43]. CNNs wurden in den 1990er Jahren ausgiebig genutzt, wurden aber aufgrund der fehlenden Rechenleistung weitgehend aufgegeben. Sie wurden in erster Linie durch Support-Vektor-Maschinen und komplizierte manuelle Extraktionsverfahren ersetzt. Ein Hauptnachteil in der Forschung und Entwicklung von CNNs in den 1990ern und 2000ern war die erforderliche Rechenleistung, um sie in großem Maßstab(scale) auf hochauflösende Bilder anzuwenden. Dies hat sich jedoch seit den 2010er Jahren geändert. In der Zwischenzeit hat sich CNN stark entwickelt und ist heute der Standardansatz zur Lösung vieler Computer-Vision-Aufgaben.

Es gibt drei Gründe, warum Deep Netzwerke erfolgreich geworden sind :Erstens mehr Rechenleistung aufgrund des Mooreschen Gesetzes, insbesondere in heutigen GPUs, zweitens mehr Trainingsdaten und Drittens bessere Algorithmen. Mit der Zunahme der Rechenleistung beschrieben die Forscher neue Wege, um konvolutionelle neuronale Netze effizienter zu trainieren, so dass Deeper Netze trainiert werden konnten. Im Allgemeinen sind CNNs sehr gut darin, Objekte wie bestimmte Hunde- oder Katzenrassen anhand feinkörniger Details zu klassifizieren, während Menschen damit Probleme haben. Ein Nachteil von Deep Learning ist, dass das Erkennen von Objekten in realistischen Einstellungen sehr große Datensätze für das Training benötigt. Gegenwärtig kämpfen die besten CNNs mit Objekten, die klein oder dünn sind oder die mit digitalen Filtern verzerrt wurden. Eine aktuelle

Studie zeigte die Unterschiede, wie Deep neuronale Netzwerke und Objekte erkennen. In dieser Arbeit werden die Bilder auf eine Weise kodiert, die Objekt nicht erkennen können, aber die Deep neuronalen Netze haben die korrekte Klasse des Objekts mit fast 99% Genauigkeit erkannt.

### 2.3.2 Operationen

Dieser Abschnitt gibt einen Überblick über die wichtigsten Konzepte, die in konvolutionellen neuronalen Netzen angewendet werden. Diese Modelle bestehen typischerweise aus mehreren Schichten und jede Schicht ist in mehrere Stufen unterteilt: convolutional layer, applying a rectified linear function, max oder average pooling.

#### Convolution

Der Name "convolutional neural network" zeigt an, dass das Netzwerk Faltung (convolution) operationen verwendet. Faltung ist eine lineare Operation, die es ermöglicht, Eingaben variabler Größe zu verarbeiten. Die Eingabe ist ein zweidimensionales Array von Daten und der Kernel(the kernel) ist ein zweidimensionales Array von lernbaren Parametern. In CNNs sind die zweidimensionalen Eingaben und Kernel Bilder. Die Faltungsoperation ist mathematisch definiert als :

$$S[i, j] = (I, K)[i, j] = \sum_n \sum_m I[m, n]K[i - m, j - n]$$

I ist das Eingabebild mit den Dimensionen m und n , K ist der Kernel und s ist die resultierende Ausgabe an der Stelle i und j.

Wenn Faltung auf Bilder angewendet wird, erstellen Faltungen 2-D-Feature-Maps, die anzeigen, wo bestimmte Features in der Eingabe erscheinen. Die Parameterfreigabe bewirkt, dass die Ebene äquivalent zur Übersetzung ist, d. h. wenn sich die Eingabe ändert, ändert sich die Ausgabe auf die gleiche Weise. Wenn das Objekt in der Eingabe verschoben wird, verschiebt sich seine Darstellung um denselben Betrag in der Ausgabe.

## Pooling

In CNN ist es üblich, die Auflösung von width und height während der Faltung beizubehalten. Pooling-Layer sind verantwortlich für die Reduzierung dieser Auflösung, wodurch kleinere und überschaubarere Repräsentationen durch Downsampling entstehen. Diese Operation wird auf jede von einer Ebene unabhängig ausgebogene Merkmalskarte angewendet. Der beliebteste Pooling-Ansatz ist als Max-Pooling bekannt.

Pooling-Layer generieren lokale Feature-bezogene Übersichtsstatistiken. Neben der Komprimierung von Feature-Repräsentationen besteht ein weiterer Vorteil darin, dass es eine eingeschränkte Invarianz für begrenzte Feature-Übersetzungen bietet. mit anderen Worten, die genaue räumliche Position des Merkmals mit der höchsten Aktivierung wird verworfen.

## Activation

Die Aktivierungsfunktion wird verwendet, um die Ausgabe eines Neurons zu begrenzen und auch Nichtlinearitäten in die linearen Aktivierungen einzuführen, die von einer Faltungsschicht erzeugt werden. Es ist wichtig, die Funktion richtig auszuwählen, um ein effektives Netzwerk zu erstellen. Die Eingabe für solche Funktionen ist typischerweise die Ausgabe einer Faltungsschicht. Es gibt mehrere Aktivierungsfunktionen, die in neuronalen Netzwerken verwendet werden, wobei einige populäre Beispiele sind:

- **Sigmoid:** Historisch gesehen ist es die am weitesten verbreitete Aktivierungsfunktion. Es nimmt reale Werte und transformiert sie in den Bereich [0, 1]. Es ist gegeben durch:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- **Hyperbolic Tangent :** Es wird berechnet als:

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

Wie die Sigmoid-Funktion, sättigen Aktivierungen leicht, aber es hat den Vorteil einer zentrierten Ausgabe von Null.

- **ReLU (Rectified Linear Unit):** rectified linear Funktion erzeugt den Ausgang mit einer Rampenfunktion wie:

$$\partial(s) = \max(0, s)$$

Diese Art von Funktion ist leicht zu berechnen und zu differenzieren (für Backpropagation). Die Funktion ist bei Null nicht unterscheidbar, aber dies hat ihre Verwendung in der Praxis nicht verhindert. ReLUs sind in letzter Zeit ziemlich populär geworden und ersetzen oft sigmoide Aktivierungsfunktionen, die glatte Ableitungen haben, aber von Gradientensättigungsproblemen und langsamerer Berechnung betroffen sind. Netzwerke mit ReLUs trainieren mehrmals schneller als solche mit tanh-Funktionen. Die gewünschten Trainingsfehlerraten von CNNs mit ReLUs werden mehrfach schneller erreicht als mit sättigenden Neuronen. ReLUs sind wichtig, da die Lerngeschwindigkeit einen großen Einfluss auf die Leistung des Modells hat, insbesondere wenn der Datensatz sehr groß ist.

## 2.4 Computer Vision

In diesem Abschnitt wird grundlegendes Wissen zur Computer Vision beschrieben und der Hauptgegenstand dieser Arbeit, Objekterkennung, als Teilproblem der Computer Vision.

### 2.4.1 Definition

Computer Vision ist die Fähigkeit von Maschinen zu sehen und zu verstehen, was in ihrer Umgebung ist. Computer Vision ist erforderlich für die Mustererkennungstechnologie, um das interessierende Objekt oder Merkmal im Sichtfeld der Kamera zu finden. Das Lokalisieren des interessierenden Objekts bestimmt oft Erfolg oder Misserfolg. Wenn die Software-Tools zur Mustererkennung den Teil im Bild nicht genau lokalisieren können, kann Computer Vision das Teil nicht verwenden, identifizieren, inspizieren, zählen oder messen.

Computer Vision beschäftigt sich mit der Gewinnung von aussagekräftigen Informationen aus den Inhalten von digitalen Bildern oder Videos. Dies unterscheidet

sich von der bloßen Bildverarbeitung, bei der visuelle Informationen auf Pixelebene manipuliert werden. Zu den Anwendungen der Computer Vision gehören die Bildklassifizierung, visuelle Erkennung, Bildsuche, Objekterkennung und Machine Vision [34].

Heute ist maschinelles Lernen eine notwendige Komponente vieler Computer Vision Algorithmen. Solche Algorithmen können als eine Kombination aus Bildverarbeitung und maschinellem Lernen beschrieben werden. Effektive Lösungen erfordern Algorithmen, die mit der großen Menge an Informationen, die in visuellen Bildern enthalten sind, fertig werden und kritisch für viele Anwendungen die Berechnung in Echtzeit durchführen können.

#### 2.4.2 Objekterkennung

In diesem Schritt werden die verschiedenen Objekterkennungs-verfahren diskutiert und verglichen, die Convolutional neural networks verwenden.

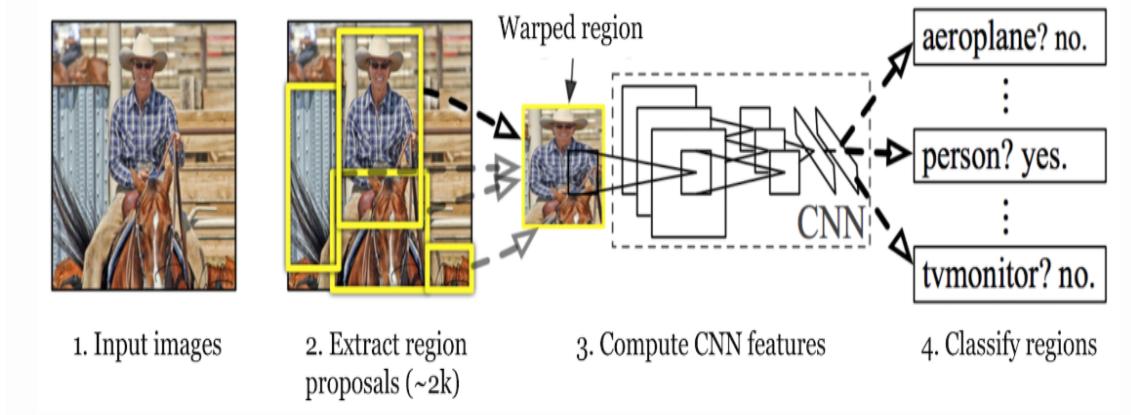
Objekterkennung ist eines der klassischen Probleme des Computer Vision und wird oft als eine schwierige Aufgabe beschrieben. In vieler Hinsicht, es ist ähnlich wie bei anderen Computer-Vision-Aufgaben, weil es eine Lösung erzeugt, die für Deformationen und Veränderungen der Beleuchtung und des Blickpunkts invariant ist. Was die Objekterkennung zu einem besonderen Problem macht ist, dass sowohl Bereiche eines Bildes lokalisiert als auch klassifiziert werden.

Objekterkennung ist die Aufgabe, Klassenlabels für und Begrenzungsrahmen zu erzeugen, die ein oder mehrere Objekte in einem Bild umgeben. Im Jahr 2015 wurde zum ersten Mal eine Maschine herausgebracht, die die menschliche Leistung in der ImageNet-Klassifizierung übertrifft [21]. Im Vergleich zur Klassifikation ist die Objekterkennung ein viel komplexeres Unterfangen und die übermenschliche Leistung durch Deep Lernen bleibt schwer fassbar. Die Schwierigkeit ergibt sich aus der Notwendigkeit, mehrere Objekte zu lokalisieren, wobei die Klasse und die Anzahl der Instanzen nicht spezifiziert sind.

#### Regional CNN (R-CNN)

Die Entstehung von R-CNN erfolgte im Jahr 2014, die mit CNN erzielten Erfolge auf die Objekterkennung zu verallgemeinern. R-CNN Vorwärtsberechnung hat mehrere Stufen ( Abbildung 2.4). Zuerst werden die interessierenden Bereiche

erzeugt. Die Region sind kategorieunabhängige Bounding-Boxen, die mit hoher Wahrscheinlichkeit ein interessantes Objekt enthalten.



**Abbildung 2.4:** Architektur von R-CNN[31]

Als nächstes wird ein convolutional netzwerk verwendet, um Merkmale von jedem Bereichsvorschlag zu extrahieren. Nachdem das Netzwerk Merkmale von der Eingabe extrahiert hat, werden die Merkmale eingegeben, um Vektormaschinen (SVM) zu unterstützen, die die Endklassifizierung bereitstellen.

Die Methode wird in mehreren Stufen trainiert, beginnend mit dem convolutional netzwerk. Nachdem der CNN trainiert wurde, werden die SVMs an die CNN-Merkmale angepasst. Schließlich wird das Erzeugungsverfahren für den Regionsvorschlag trainiert[31].

- **Nachteile**

R-CNN ist eine wichtige Methode, da es die erste praktische Lösung für die Objekterkennung mit CNNs bietet. weil es das erste ist, hat es viele Nachteile, die durch spätere Methoden verbessert wurden. In seiner Arbeit für Fast R-CNN [30] von 2015 führt "Girshick" drei Hauptprobleme von R-CNN auf:

- Erstens besteht das Training aus mehreren Stufen, wie in Abbildung 2.7 beschrieben.
- Zweitens ist Training aufwändig. Sowohl für das SVM- als auch für das regio-

nale Vorschlagstraining werden Features aus jedem Regionsvorschlag extrahiert und auf der Festplatte gespeichert. Dies erfordert Tage der Berechnung und Hunderte von Gigabyte Speicherplatz.

- Drittens, das ist vielleicht am wichtigsten, ist die Objekterkennung langsam und erfordert fast eine Minute für jedes Bild, sogar auf einer GPU. Dies liegt daran, dass die CNN-Vorwärtsberechnung für jeden Objektvorschlag separat durchgeführt wird, selbst wenn die Vorschläge von demselben Bild stammen oder einander überlappen.

### Fast R-CNN

Fast R-CNN veröffentlicht im Jahr 2015 von "Girshick", bietet eine praktischere Methode zur Objekterkennung. Die Hauptidee besteht darin, den Vorwärtsdurchlauf des CNN für das gesamte Bild durchzuführen, anstatt es für jeden **Region** (RoI) getrennt durchzuführen[30].

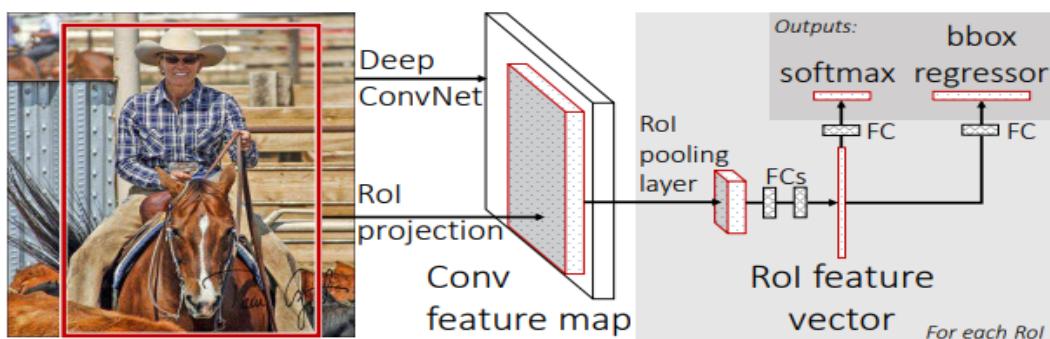


Abbildung 2.5: Architektur von Fast R-CNN[30]

Die allgemeine Struktur von Fast R-CNN ist in Abbildung 2.5 dargestellt. Das Verfahren empfängt als Eingabe ein Bild plus Bereiche von Interesse, die aus dem Bild berechnet wurden. Wie in R-CNN werden die Regions mit einer externen Methode generiert. Das Bild wird unter Verwendung eines CNN verarbeitet, das mehrere convolutional schicht- und Max-Pooling-Schichten enthält.

Die convolutional Feature-Map, die nach diesen Layern generiert wird, wird in eine ROI-Pooling-Ebene eingegeben. Dies extrahiert einen Merkmalsvektor mit fester Länge für jeden ROI aus der Merkmalskarte. Die Feature-Vektoren sind die Eingabe für vollständig verbundene Layer, die mit zwei Ausgabeschichten verbunden

sind: eine Softmax-Schicht, die Wahrscheinlichkeitsschätzungen für die Objektklassen erzeugt, und eine reellwertige Schicht, die mithilfe der Regression berechnete Bounding-Box-Koordinaten ausgibt[30].

### Faster R-CNN

Faster R-CNN von Ren ist eine integrierte Methode. Die Hauptidee besteht darin, geteilte convolutional schichten für die Erzeugung von Regionsangeboten und für die Erkennung zu verwenden. Die Autoren entdeckten, dass Feature-Maps, die von Objekterkennungsnetzwerken generiert wurden, ebenfalls zum Generieren der Regionsvorschläge verwendet werden können. Der vollständig Convolutional abschnitt des Faster R-CNN-Netzwerks, das die Feature-Vorschläge generiert, wird als **region proposal network** (RPN) bezeichnet. Die Autoren verwendeten eine Fast-R-CNN-Architektur für das Detektionsnetzwerk.

Ein Faster R-CNN-Netzwerk wird durch abwechselndes Training für RoI-Erzeugung und -Erkennung trainiert. Zuerst werden zwei getrennte Netzwerke trainiert. Dann werden diese Netzwerke kombiniert und neu abgestimmt. Während der Abstimmung werden bestimmte Ebenen beibehalten und bestimmte Ebenen trainiert [35] .

Das trainierte Netzwerk empfängt ein einzelnes Bild als Eingabe. Die gemeinsam genutzten Convolutional schichten erzeugen aus dem Bild Feature-Maps. Diese Merkmalskarten werden dem RPN zugeführt. Der RPN gibt Regionenvorschläge aus, die zusammen mit den Merkmalskarten an die Enddetektionsschichten eingegeben werden. Diese Layer enthalten eine RoI-Pooling-Ebene und geben die endgültigen Klassifizierungen aus.

### SSD

Der Single Shot MultiBox Detektor (SSD) bringt integrierte Erkennung noch weiter. Die Methode erzeugt überhaupt keine Vorschläge und auch keine Neuabtastung von Bildsegmenten. Es generiert Objekterkennungen mit einem einzigen Durchlauf eines Convolutional netzwerkes.

Der Algorithmus ähnelt einer Sliding-Window-Methode und beginnt mit einem Standardsatz von Bounding-Boxen. Dazu gehören verschiedene Seitenverhältnisse und Skalen. Die Objektvorhersagen, die für diese Felder berechnet werden, enthal-

ten Offset-Parameter, die vorhersagen, um wie viel sich die korrekte Begrenzungsbox, die das Objekt umgibt, von der Standardbox unterscheidet [40].

Der Algorithmus behandelt verschiedene Maßstäbe durch Verwendung von Merkmalszuordnungen von vielen verschiedenen Faltungsschichten (d. h. Größeren und kleineren Merkmalszuordnungen) als Eingabe in den Klassifizierer.

### Zusammenfassung

Zusammenfassend ist CNN eine flexible Methode, die besonders in der Objekterkennung Anwendung gefunden hat. Traditionell waren die Ergebnisse in dieser Aufgabe für die Verwendung in Echtzeitsystemen nicht ausreichend. Liu. [40] verglichen die Leistung von Fast R-CNN, Faster R-CNN und SSD mit dem PASCAL VOC 2007 Testset . Bei der Verwendung von Netzwerken, die auf PASCAL VOC 2007 Trainingsdaten trainiert wurden, Fast R-CNN erreichte eine durchschnittliche mittlere Genauigkeit von 66,9. Faster R-CNN zeigte eine etwas bessere Leistung mit einer Genauigkeit von 69,9. SSD erreicht eine Genauigkeit von 68,0 mit einer Eingangsgröße von 300 x 300 und 71,6 mit einer Eingangsgröße von 512 x 512. Da die Standardimplementierungen von Fast R-CNN und Faster R-CNN 600 als die Länge der kürzeren Dimension des Eingabebilds verwenden, SSD scheint mit ähnlich großen Bildern besser zu funktionieren[40].

in dieser Arbeit wird Faster R-CNN als Lernmethode für Objekterkennungsexperimente ausgewählt. Faster R-CNN ist bereits gut etabliert und verfügt über Implementierungen und vortrainierte Netzwerke für verschiedene Plattformen. Obwohl die fortgeschrittenen Methoden, wie Fast R-CNN, eine geringfügige Erhöhung der Genauigkeit liefern, ist ihr Hauptbeitrag eine verbesserte Geschwindigkeit. Weil die Auswertung der Ausführungszeit größtenteils außerhalb des Rahmens dieser Arbeit liegt, hätten diese Methoden den Experimenten nur wenig zusätzlichen Wert verliehen.

## 2.5 Deployment Pipeline

In diesem Schritt wird einen Überblick über die Deployment Pipeline gegeben. **Der erste Abschnitt** führt eine Einführung in Deployment Pipeline. Darauf folgt, in **Abschnitt 2**, die Grundlagen der Deployment Pipeline. Abschließend wird im

**Abschnitt 3** Die Anforderung der Arbeit behandelt.

### 2.5.1 Definition

Noch einen Schritt weiter geht die Idee des kontinuierlichen Einsatzes. Der einzige zusätzliche Schritt, den es erfordert, ist die Automatisierung der Bereitstellung einer neuen Version in die Produktion. Dies bedeutet, dass die Pipeline, die mit dem Einchecken des Codes beginnt und mit dem bereitgestellten Produkt für den Benutzer in der Produktion endet, vollständig automatisiert ist. Um dies zu erreichen, muss die Umgebung so Produktion wie möglich sein, um unangenehme Überraschungen bei der Produktion zu vermeiden. Auf der anderen Seite stellt dies sicher, dass das Produkt jederzeit zuverlässig eingesetzt werden kann.

Die Deployment Pipeline ist eine automatisierte Manifestation des Prozesses, mit dem Software aus der Versionskontrolle in die Hände der Benutzer gelangt. Jede Änderung der Software durchläuft einen komplexen Prozess auf dem Weg zur Freigabe. Der Fortschritt jeder Änderung kann verfolgt und gesteuert werden, während sie sich von der Versionskontrolle über verschiedene Tests und Bereitstellungen für die Benutzer verschiebt. Dieser Prozess umfasst das Erstellen der Software, gefolgt vom Fortschritt dieser Builds über mehrere Test- und Bereitstellungsphasen. Dies erfordert wiederum die Zusammenarbeit zwischen vielen Einzelpersonen und vielleicht mehreren Teams. Abhängig von der Anwendung und der Änderung kann die Deployment Pipeline in andere erforderliche Umgebungen und Pfade verzweigen. Die Bereitstellung in diesen Umgebungen kann nacheinander, parallel oder als optionale manuell ausgewählte Stufe erfolgen. Die Anzahl der zu verwaltenden Deployment Pipeline kann sich abhängig von der Anzahl der Anwendungen, Teams, Anforderungen und Umgebungen erhöhen[14].

Unabhängig von der tatsächlichen Bereitstellung erfordert die kontinuierliche Softwareentwicklung eine Deployment Pipeline, die einen automatisierten Satz von Werkzeugen vom Code zur Lieferung verwendet. Die Pipeline bietet eine Rückkopplungsschleife für alle Beteiligten in allen Phasen des Lieferprozesses. Bei einem automatisierten System geht es nicht darum, dass Software ohne Bedieneraufsicht in die Produktion geht. Der Sinn der automatisierten Pipeline besteht darin, dass während des Software-Durchlaufs verschiedene Stufen beispielsweise durch Operationen und Testteams ausgelöst werden können [15].

### 2.5.2 Grundlagen der Deployment Pipeline

Der Deployment Pipeline Prozess beginnt damit, dass die Entwickler Änderungen an ihrem Versionskontrollsyste vornehmen. An diesem Punkt reagiert das Managementsystem für die kontinuierliche Integration auf das Commit, indem es eine neue Instanz der Pipeline auslöst. Die erste Stufe der Pipeline kompiliert den Code, führt Komponententests durch, führt eine Codeanalyse durch und erstellt Installationsprogramme. Moderne CI-Server bieten die Möglichkeit, Artefakte wie diese zu speichern und sie sowohl für die Benutzer als auch für die späteren Phasen in der Pipeline leicht zugänglich zu machen. Es gibt noch andere Aufgaben, die Sie möglicherweise auch als Teil der Festschreibungsphase der Pipeline ausführen, z. B. das Vorbereiten einer Testdatenbank für die Akzeptanztests.

Der zweite Schritt besteht typischerweise aus länger laufenden automatisierten Akzeptanztests. Auch hier sollte der CI-Server diese Tests aufteilen, die parallel ausgeführt werden können, um ihre Geschwindigkeit zu erhöhen und einen schnelleren Feedback zu geben. Diese Phase wird automatisch durch den erfolgreichen Abschluss der ersten Stufe in der Pipeline ausgelöst.

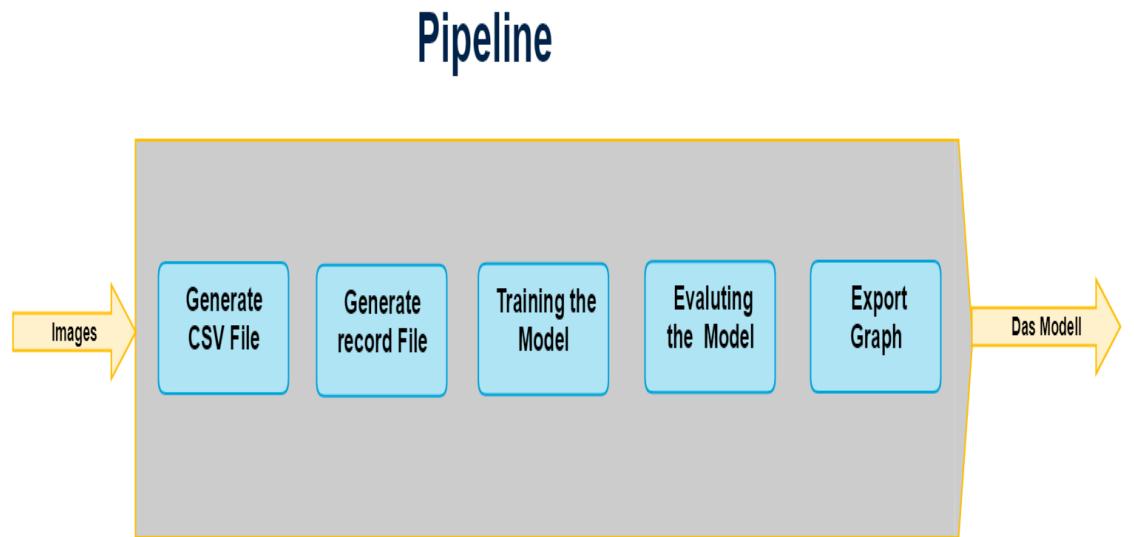
Zu diesem Zeitpunkt wird die Pipeline verzweigt, um eine unabhängige Bereitstellung des Builds für verschiedene Umgebungen zu ermöglichen. Derselbe Grundsatz gilt für weitere Phasen in der Pipeline. In den verschiedenen Umgebungen, sind jedoch unterschiedliche Benutzergruppen vorhanden, die diese Umgebungen bereitstellen können.

Schließlich ist es wichtig, dass der Zweck all dessen ist, Feedback so schnell wie möglich zu bekommen. Um den Feedback-Zyklus schnell zu machen, welcher Build wird in welche Umgebung implementiert und welche Phasen in der Pipeline hat jeder Build bestanden [15].

### 2.5.3 Anforderung der Arbeit

Deployment Pipeline für maschinelles Lernen setzt sich aus verschiedenen Themen zusammen und wird zusammen mit den verwandten Prozessen Continuous Integration und Continuous Deployment häufig eingesetzt. Es gibt viele Erfahrungsberichte und Tipps, die beachtet werden sollten, um Continuous Delivery erfolgreich zu implementieren. Dabei werden besonders die Aspekte einer möglichst vollständigen

digen Automatisierung, sowie die Akzeptanz von Continuous Delivery unter den Entwicklern häufig aufgeführt.



**Abbildung 2.6:** Deployment Pipeline des Projekts

Das Ziel der Deployment Pipeline ist es, ein Prozess zur Automatisierung der verschiedenen Schritten des maschinellen Lernens zu strukturieren (Abbildung 2.6 zeigt die verschiedene Schritte, die automatisiert sind). Diese Pipeline wird automatisierte Tests umfassen. Da die maschinelle Bildverarbeitung auf Bildern basiert, die große Mengen an Bildern verarbeiten und passende Bilder zu einem Experten vorschlagen, ist dies ebenfalls Teil der Lösung. Die Pipeline des Projekts umfasst mehrere Schritte. “Generate CSV-File, Generate Record File, Training the Model, Evaluation the Model und Export frozen-inference-graph“. Jede dieser Schritte erfüllt unterschiedliche Anforderungen und die Bereitstellung in den verschiedenen Schritten wird über das Versionskontrollsystem verwaltet. Die Pipeline, die mit dem Einchecken der Datensätze beginnt und mit dem bereitgestellten das Modell in der Produktion endet.

Der Deployment Pipeline Prozess beginnt mit der ersten Herausforderung, dass die Bilder der Datensätze überprüft werden und eine CSV-Datei mit den übereinstimmenden Bildern erzeugt. Die wichtigsten Schritte beim ML-Testen sind das Testen des Datensatzes. Als nächstes werden die CSV-Datei und die passenden Bilder verwendet, um eine Record-Datei für das Trainingsmodell zu erstellen. Die dritte Herausforderung, der sich jede maschinelle Lernanwendung stellt, während sie auf Pipeline angewendet wird, ist trainieren des Modells. Das Modell wird mit Record-Datei trainiert und ausgewertet. Der Pipeline-Prozess endet mit dem Schritt “Export Graph“ und liefert das Modell als Ausgabe, das in der Produktion bereitgestellt werden kann. Alle Schritte werden automatisch ausgeführt.

Jede Änderung des Datensatzes durchläuft einen Prozess, der die neuen Datensätze überprüft und zu die geprüften Daten hinzugefügt. Als Nächstes wird das Modell mit allen Daten neu trainiert und ausgewertet.

# Kapitel 3

## State of the Art

In diesem Abschnitt werden die State of the Art für die Objekterkennung behandelt. Es wird einen Überblick über die aktuelle Forschung der verschiedenen Themen gegeben, die mit dieser Arbeit verbunden sind.

### 3.1 Machine Learning

In den letzten Jahrzehnten ist maschinelles Lernen mehr und mehr allgegenwärtig in der Informationstechnologie geworden. Trotzdem ist die Branche noch in ihrer Entstehung. Maschinelles Lernen beschreibt eine Fülle von Techniken, die einen datengesteuerten Ansatz zur Problemlösung teilen. Auf diese Weise "lernen" Algorithmen, Muster in einem Satz von Trainingsdaten zu erkennen, ohne dafür explizit programmiert zu sein. Solche Ansätze haben zum größten Teil hochspezifische Merkmalsextraktionsprogramme in Bereichen wie Computer Vision ersetzt. Der Begriff maschinelles Lernen ist in den letzten Jahrzehnten immer populärer geworden. Andere Felder haben Begriffe mit ähnlichen Bedeutungen, wie Datenanalyse in Statistiken und Musterklassifizierung in der Mustererkennung. In den frühen Tagen der KI verwendete die ML-Forschung hauptsächlich symbolische Daten und der Entwurf von Algorithmen basierte auf Logik.

Die eigentliche Entwicklung des statistischen Lernens erfolgte nach 1986, als David Rumelhart und James McClelland den nicht linearen Backpropagation-Algorithmus vorschlugen[11]. KI, Mustererkennung und Statistikforscher interessierten sich für diesen Ansatz und unter dem Einfluss der realen Anforderungen wurde Machine Learning, insbesondere die statistische Machine Learning, allmählich von der traditionellen KI und Mustererkennung unabhängig. Die Forscher

haben frühe symbolische Machine-Learning-Methoden gemieden, weil ihnen die theoretischen Generalisierungsgarantien fehlten. Da jedoch die Komplexität der realen Daten einen allgegenwärtigen Lernalgorithmus unmöglich macht, könnte die Qualität des Daten- und Hintergrundwissens der Schlüssel zum Erfolg von Machine Learning sein.

Die Forschungsgemeinschaft interessiert sich nicht mehr für die Algorithmenentwurfsprinzipien der nicht linearen Rückpropagation. Aber diese Forschung erinnerte die Menschen an die Bedeutung nicht linearer Algorithmen. Nicht lineare Lernalgorithmen theoretisch und systematisch zu entwerfen, ist ein wichtiger Impuls der aktuellen statistischen ML-Forschung.

Machine Learning ist zu einem wichtigen Thema für Mustererkennungsforscher und andere geworden. Leo Breiman veröffentlicht “Statistical Modeling: The Two Cultures“, die Machine Learning als eine Unterkategorie der Statistik betrachtet[23]. Dieser Ansatz führte zu vielen neuen Richtungen und Ideen für Machine Learning.

Machine Learning hat eigentlich zwei Grundlagen. Der erste ist die Statistik. Da es das Ziel von Machine Learning ist, ein Modell aus beobachteten Daten zu schätzen, muss es statistische Messungen verwenden, um die Leistung des Modells zu bewerten und das Modell zu schätzen. Machine Learning benötigt auch Statistiken, um Rauschen in den Daten zu filtern. Die zweite Grundlage sind computerwissenschaftliche Algorithmusentwurfsmethoden, um Parameter zu optimieren. In den letzten Jahren bestand das Hauptziel von Machine Learning darin, einen Lernenden linear von diesen Parametern abhängig zu machen [20].

## 3.2 Deep Learning

Moderne neuronale Netze werden oft als deep neuronale Netze bezeichnet. Obwohl seit den 1980er Jahren mehrschichtige neuronale Netze existierten, verhinderten mehrere Gründe das effektive Training von Netzwerken mit mehreren versteckten Schichten [17].

Viele Forschungsarbeiten konzentrierten sich darauf, neue Architekturen zu schaffen, die anderen hochmodernen Netzwerken in wettbewerbsintensiven Benchmarks, die auf dem ImageNet-Dataset ausgewertet werden, standhalten[1, 6]. Andere Forschungsstudien konzentrierten sich auf die Analyse der zugrunde liegenden Archi-

tekturen von Faltungsnetzwerken im Allgemeinen und bewerten, warum die Netzwerke so gut funktionieren [2, 13]. In der gegenwärtigen Forschung werden Deep Learning Modelle oft mit anderen Methoden verknüpft und angewendet, wie zum Beispiel natural language processing (NLP) [28] oder reinforcement learning (RL)[18, 10] Algorithmen, um Lösungen für neue Problemdomänen.

Die Fähigkeit, den visuellen Kontext zu verstehen, ist eine kritische und wohl die komplizierteste kognitive Fähigkeit, die reale Welt zu verstehen. Laut Li Fei Fei[25] ist die Computer Vision der Schlüssel, mit dem die Technologie zukünftige intelligente Maschinen bauen kann, die wie Menschen sehen und denken können. Wenn Menschen eine gegebene Aufgabe lösen, verlassen sie sich häufig auf ihre Intuition und nutzen verfügbare kontextuelle Informationen über das Problem [42]. Menschen nutzen ihr Wissen, um das Problem zu vereinfachen und die Aufgabe zu lösen, indem sie ihre Erfahrung mit verwandten Konzepten nutzen. Im Gegensatz dazu fehlt technischen Systemen diese Fähigkeit, Wissen in mehreren Darstellungen von Natur abzuleiten. In Bezug auf visuelle Verarbeitungssysteme, Die Möglichkeiten klassischer Objekterkennungsmethoden beschränken sich meist auf die Kategorisierung bekannter Objekte. In neueren Untersuchungen sind neue Ideen aufgetaucht, wie zusätzliche Informationen aus Bildern extrahiert werden können, um die visuelle Erkennungsleistung zu verbessern [7, 27, 38, 39]. Computer Vision Algorithmen, die Objekte mit Attributen beschreiben und auf kontextabhängigen Informationen basierende Objekteigenschaften ableiten, bieten neue Möglichkeiten, neue Objekte zu lernen und zu erkennen [42]. Dieser attributzentrierte Ansatz hat den Vorteil, dass neue Objekte durch Beschreibungen der Objekte auf hoher Ebene erkannt werden können, anstatt Modelle auf diesen Bildern zu trainieren [7]. In der Literatur [16, 18, 38] werden Attribute wie Material, Form und Farbe als relevanter Kontext eines detektierten Objekts untersucht. Diese Attribute werden für jedes Objekt von deep networks oder maschinellen Lernklassifikatoren vorhergesagt.

### 3.3 Object detection

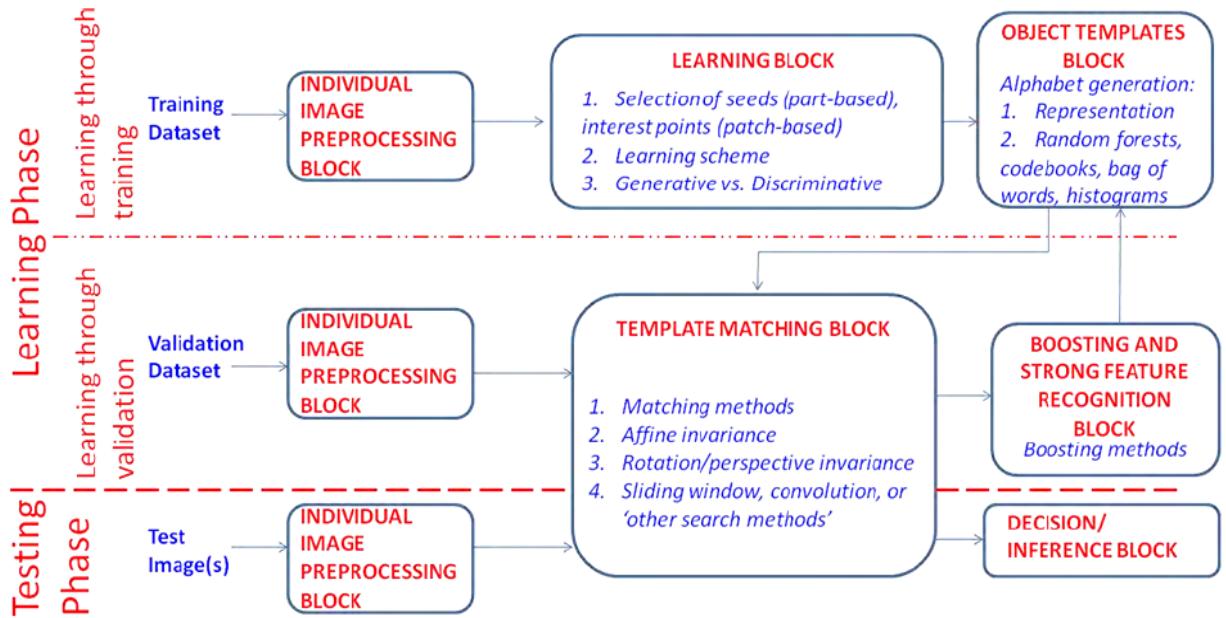


Abbildung 3.1: Basisblockdiagramm eines typischen Objekterkennungs [9]

Im Bereich der Objekterkennung wird viel geforscht. Um die Diskussion über Methoden und Ideen verschiedener Forschungsarbeiten zu erleichtern, zuerst wird ein allgemeines Blockdiagramm präsentiert, das für jede Objekterkennungs- oder Erkennungsmethode in Abbildung 3.1 anwendbar ist. Spezifische Methoden, die von verschiedenen Forschern vorgeschlagen wurden, können geringfügig von diesem allgemeinen Blockdiagramm abweichen [8].

Ein solcher Algorithmus kann in zwei verschiedene Phasen unterteilt werden, nämlich Lernphase und Testphase. In den Lernphasen verwendet die Maschine eine Menge von Bildern, die Objekte enthält, die zu bestimmten vorbestimmten Klassen gehören, um zu lernen, die Objekte zu identifizieren, die zu diesen Klassen gehören. Sobald der Algorithmus trainiert wurde, die Objekte zu identifizieren, die zu den spezifizierten Klassen gehören, verwendet der Algorithmus in der Testphase sein Wissen, um die spezifizierten Klassenobjekte aus den Testbildern zu identifizieren.

Der Algorithmus für die Lernphase kann weiter in zwei Teile unterteilt werden, d.h. Lernen durch Training und Lernen durch Validierung. Eine Gruppe von Bildern, die Objekte der angegebenen Klassen enthalten, das so genannte Trainings-Dataset, wird verwendet, um die grundlegenden Objektvorlagen für die angegebenen Klassen zu lernen. Abhängig von der Art der Merkmale werden die Trainingsbilder vorverarbeitet und in den Lernblock geleitet. Der Lernblock lernt dann die Merkmale, die jede Klasse charakterisieren. Die gelernten Objektmerkmale werden dann als Objektvorlagen gespeichert. Diese Phase wird als "Lernen durch Training" bezeichnet. Die Objektvorlagen, die in dieser Stufe gelernt werden, werden als schwache Klassifizierer bezeichnet. Die gelernten Objektvorlagen werden mit dem Validierungsdatensatz getestet, um die vorhandenen Objektvorlagen auszuwerten. Unter Verwendung von Verstärkungstechniken werden die Vorlagen für gelernte Objekte verfeinert, um eine größere Genauigkeit beim Testen zu erreichen. Diese Phase wird als "Lernen durch Validierung" bezeichnet, und die Klassifizierer, die nach dieser Stufe erhalten werden, werden als starke Klassifizierer bezeichnet.

Die Forscher haben an vielen spezifischen Aspekten des oben erwähnten Systems gearbeitet. Einige Beispiele umfassen die Auswahl des Feature-Typs, die Methode zum Generieren der Features, die Methode zum Erlernen der konsistenten Merkmale einer Objektklasse, die Besonderheit des Lernprogramms, die Darstellung der Vorlagen, die Schemata, um eine Übereinstimmung zwischen einem Test oder Validierungsbild zu finden und eine Objektvorlage (obwohl die Größe und Ausrichtung eines Objekts im Testbild von der gelernten Vorlage abweichen kann) und so weiter.

Viele Forscher haben richtig argumentiert, dass ein robustes Objekterfassungs- und -charakterisierungsschema typischerweise mehr als einen Merkmalstyp erfordern muss, um eine gute Leistung gegenüber einer großen Anzahl von Klassen zu erzielen [4, 19, 45]. Daher werden in dieser Arbeit Regionen Merkmale zusammen

mit Kontur Fragmenten verwendet. Im Vergleich zu [4], das für die endgültige Entscheidung nur eine Art von Objektvorlage verwendet hat, wird eine kombinierte Objektvorlage verwendet, die Kanten-, Form- und Regionsmerkmale speichert und jedem Merkmal einen Stärkewert zuweist, so dass die kombinierte Wahrscheinlichkeitsfunktion verwendet wird Entscheidung kann während des Testens getroffen werden. Ein solches Schema muss sicherstellen, dass potenzielle Objekte häufiger identifiziert werden, obwohl die Wahrheit variiert kann und die Entscheidung durch die Wahl eines geeigneten Schwellenwerts getroffen werden kann. Dies ist besonders nützlich bei stark verdeckten oder verrauschten Bildern.

# Kapitel 4

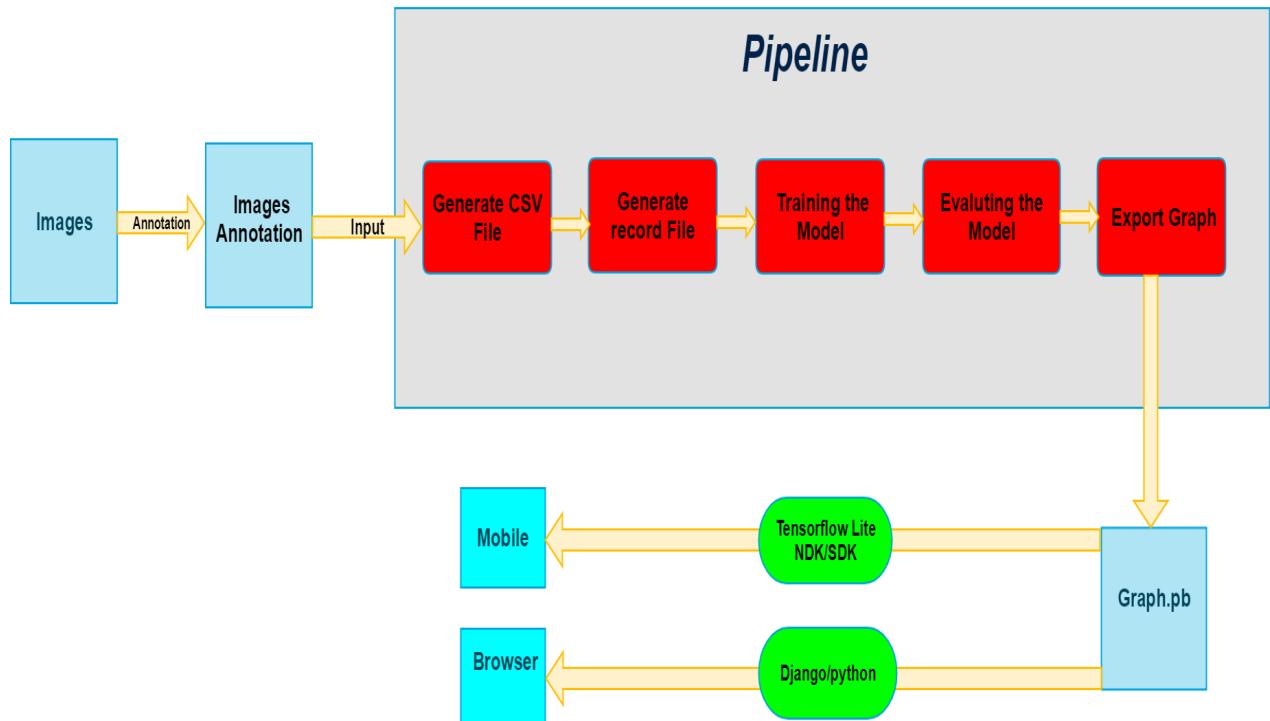
## Implementierung

Dieser Abschnitt gibt einen Überblick über den experimentellen Aufbau und die Frameworks, die in dieser Arbeit verwendet wurden. Erstens wird die Struktur des gesamten Projekts gezeigt und die Software und die Hardware und deren Einsatz beschreibt .Als nächstes werden die Schritt training, die Evaluierung und der Testing eines CNN-Modells beschrieben.

### 4.1 Projektarchitektur

In diesem Projekt wird ein pre-trained Modell verwendet. Der Vorteil eines vortrainierten Modells besteht darin, dass das Modell nicht von Grund auf neu erstellt wird. Ein Modell, das für ein ähnliches Problem trainiert wurde, kann als Ausgangspunkt für das Training des Netzwerks verwendet werden. In dieser Arbeit wird ein Faster-Modell (faster-rcnn-inception-v2-coco) gewählt. Dieses Modell wird als Initialisierungs-Checkpoint für das Training verwendet. Dieses Projekt hat eine Reihe von Schritten zu folgen. Die Abbildung 4.1 zeigt Ablaufprozess des Projekts.

Zuerst sollen die Datensätze für einen bestimmten Anwendungsfall erstellt werden. Die Datensätze sollten alle Objekte enthalten, die sollen erkannt werden. Nachdem alle Bilder aufgenommen wurden, Der nächste Schritt ist sie zu annotieren. **LabelImg** wird zum Beschriften aller Bilder verwendet.**LabelImg** ist ein grafisches Werkzeug für Bildanmerkungen und Beschriftungsobjekte, die in Bildern Begrenzungsrahmen enthalten. Danach werden die Bilder und die Annotation als Input für die Pipeline Prozess aufgenommen.



**Abbildung 4.1:** Ablaufprozess des Projekts

Das Framework **PipelineAI** [49] wird zum Erstellen einer Pipeline verwendet. Das Objekterkennungs-Modell nimmt keine XML-Dateien als Eingabe, aber sie benötigt record-Dateien, um das Modell zu trainieren. Das Pipeline wird die Bilder der Datensätze überprüft und XML-Dateien in CSV-Dateien konvertiert, die verwendet werden, um train.records (was das Ergebnis der Trainingsbilder ist) und test.records (was das Ergebnis der Testbilder ist) zu generieren. Nachdem kann das Training des Modells fortgesetzt werden. Als nächstes wird das Modell mit test Dataset evaluiert. Die letzten Schritte in Pipeline muss das Objekterkennungs-Graph exportiert werden. Es wird Objekterkennungs-Graph (frozen-inference-graph.pb) als Ausgabe geliefert.

Schließlich kann das Objekterkennungs-Graph des Modells bereitgestellt werden. In diesen Schritten werden zwei verschiedenen Verfahren genutzt. Erstens wird das Objekterkennungs-Graph in einem Android-Gerät mit Verwendung Tensorflow Lite getestet. Zweitens wird es in der Webanwendung bereitgestellt.

## 4.2 Software und Hardware

Dieses Projekt nutzte mehrere Softwarebibliotheken, Pakete und Programme, um maschinelles Lernen zu nutzen. Python war die Wahl der Programmiersprache und TensorFlow wurde für die Deep Learning-Berechnungen verwendet, die wiederum eine Liste von Abhängigkeiten enthält. TensorFlow bietet eine Version für CPU-Nutzung und eine andere für GPU, dieses Projekt verwendet die GPU-Version. Diese Version benötigt zusätzliche Programme vom GPU-Designer NVIDIA, wie CUDA Toolkit, cuDNN und deren GPU-Treiber. Um das Modell in der Produktion bereitzustellen, werden Android mit Tensorflow Lite in der mobilen Anwendung und Django mit Python in der Web-Anwendung verwendet.

### 4.2.1 Graphics Processing Unit :GPU

Ein effizienter Code wird auf der Basis einer gut aufgebauten Softwarearchitektur entwickelt. In den vorherigen Abschnitten wurden einige Optimierungen hinsichtlich des Typs zu entwerfenden Algorithmus vorgestellt. Diese Optimierungen betreffen jedoch ein hohes Designniveau und die zugrundeliegende GPU-Architektur muss ebenfalls berücksichtigt werden, da das Programm tatsächlich auf einer GPU läuft. Die hohe Parallelität der GPU nutzt, um jedem GPU-Thread ein Berechnungsfenster zuzuordnen. Die GPU führt eine Schleife über jede Skalierung durch und initiiert eine integrale Bildberechnung. Dies wird mit CUDA's NVIDIA Performance Primitives (NPP) erreicht [36].

TensorFlow hat zwei Hauptformen seiner Software, eine Version, die die zentrale Verarbeitungseinheit (CPU) verwendet, und eine, die die graphische Verarbeitungseinheit (GPU) verwendet. Die beste Version hängt von der Hardware des Computers ab, die auf die Berechnungen angewendet werden. Je nachdem, welche Hardware am besten ist, sollte die entsprechende Version ausgewählt werden. Allgemein wird jedoch gesagt, dass die GPU besser für maschinelles Lernen geeignet ist: Benchmarks und Vergleiche zeigen diesen Trend ebenfalls [26].

Um den Grund dafür zu verstehen, wird ein allgemeiner Vergleich zwischen CPUs und GPUs benötigt. Eine CPU kann viele schnelle Berechnungen durchführen, solche Berechnungen sind am häufigsten auf einem Computer, dies steht im Gegensatz zu dem Zweck einer GPU, die stattdessen für komplexe, aber langsamere

Berechnungen optimiert ist. Spiele verwenden komplexe Berechnungen, aus diesem Grund wurde die GPU erstellt. Diese komplexen Berechnungen werden auch vom Deep Learning genutzt, aus diesem Grund werden GPUs für ihre Verwendung bevorzugt. CPUs allein sind nicht genug. Sie können die Verarbeitung machen, aber die schiere Menge an unstrukturierten Daten, die analysiert werden müssen, um Modelle für das Deep Learning aufzubauen und zu trainieren, kann sie wochenlang aushalten lassen.

#### 4.2.2 Tensorflow

TensorFlow ist eine Open-Source-Softwarebibliothek für maschinelle Lernanwendungen, die vom Google's Brain Team entwickelt wurde. TensorFlow ist die neueste Bibliothek, die in Python für numerische Berechnungen geschrieben wurde. Es wurde entwickelt, um dem Bedarf des Unternehmens nach einer flexiblen, skalierbaren und tragbaren Plattform für maschinelles Lernen auf Forschungs- und Produktionsebene gerecht zu werden. Es wird derzeit im Unternehmen in mehreren Produkten (z. B. Google Mail) und von Drittanbietern wie Snapchat, Uber und Twitter eingesetzt.

Es hatte einen großen Erfolg in der Machine Learning Community und in weniger als einem Jahr hatte es auch eine Menge Unterstützung und Entwicklung von Google selbst, mehr von vielen Community-Projekten, die in jedem Bereich von Deep Learning entwickelt wurden. Im Mai 2016 hat Google bekannt gegeben, dass es TensorFlow im AlphaGo-Projekt verwendet hat, mit einer speziellen Hardware, die extra dafür entwickelt wurde, die Leistung der Bibliothek zu steigern.

Data flow graphs sind ein zentrales Element von TensorFlow, wobei Knoten im Graphen mathematische Operationen darstellen (wie convolution und Pooling) und Graphenflanken(graph edges) als mehrdimensionale Datenfelder (Tensoren) dienen, die zwischen ihnen kommuniziert werden. TensorFlow kann solche Graphen auf eine Vielzahl von Geräten wie mobilen Geräten oder CPU / GPU-Clustern abbilden. Die Bibliothek kann mit API wie Python, C und C ++, Javascript zugriffen werden[36].

#### TensorFlow Lite

TensorFlow Lite unterstützt bereits die Bereitstellung auf mobilen und eingebetteten Geräten. Weil es einen Trend gibt, ML in mobile Anwendungen einzubauen,

und da die Benutzer höhere Erwartungen an ihre mobilen Anwendungen in Bezug auf Kamera und Sprache haben. Zu den Optimierungen in Tensorflow Lite gehören die Hardwarebeschleunigung durch die Frameworks wie die Android Neural Network API und für Mobilgeräte optimierte CNNs. Tensorflow-trainierte Modelle werden von Tensorflow automatisch in das Tensorflow Lite Modellformat konvertiert[37].

### TensorFlow Object Detection API

Die TensorFlow Object Detection API ist ein Open-Source-Framework, das auf TensorFlow aufbaut, um das "Konstruieren, Trainieren, Evaluation und Deployment von Objekterkennungsmodellen" zu erleichtern[51]. Um dies zu erreichen, bietet die TensorFlow Object Detection API dem Benutzer mehrere vorgebildete Objekterkennungsmodelle mit Anweisungen zur Feinabstimmung und Verwendung der Modelle für Objekterkennungsaufgaben.

TensorFlow Object Detection API kann mit verschiedenen vorgebildeten Modellen verwendet werden. In dieser Arbeit wurde ein Faster-Modell (faster-rcnn-inception-v2-coco) gewählt. Die **Abbildung 4.2** zeigt Das Tensorflow-Modell Ergebnis.

### Why TensorFlow Object Detection API

TensorFlows Object Detection API ist ein leistungsstarkes Tool, mit dem Objekterkennungsmodelle einfach erstellt, trainiert und implementiert werden können. In den meisten Fällen ist das Training eines gesamten Deep Netzwerkes von Grund auf zeitaufwendig und erfordert große Datensätze. Dieses Problem kann gelöst werden, indem der Vorteil des Transferlernens mit einem vorgebildeten Modell mithilfe der TensorFlow-API genutzt wird.



**Abbildung 4.2:** Vortrainierte Tensorflow-Modell Ergebnis[51]

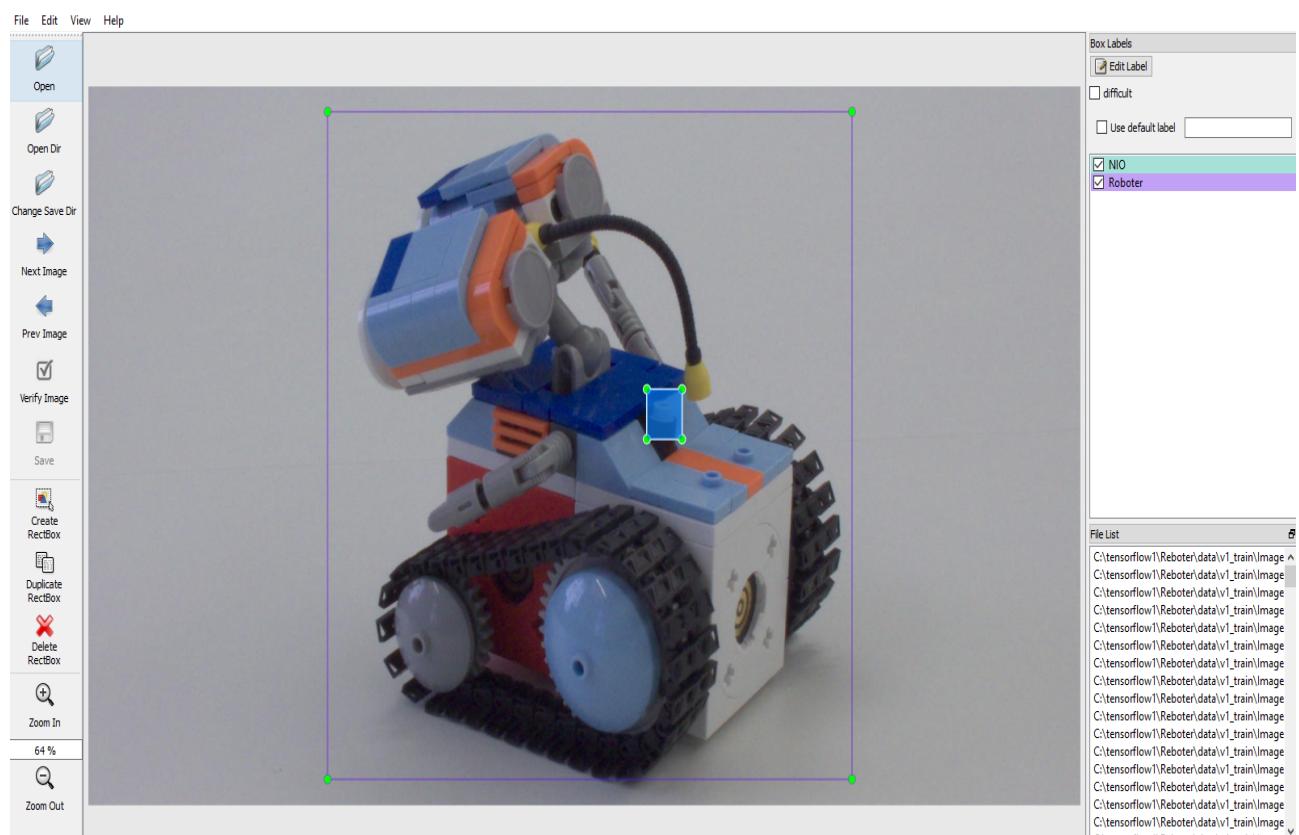
### 4.3 Datasets

Der Datensatz ist die primäre Qualitätsquelle des Lernens. Um ein neuronales Netzwerk für die aufgabenabhängige Vorhersagbarkeit zu trainieren, musste zuerst ein Datensatz von Bildern mit aufgezeichneten Blickdaten erfasst werden. Zwei Untergruppen von Datensatz wurden aufgenommen. eine zum Trainieren des Modells (70%) und eine für Evaluation des Modells(30%) .

#### Annotation

Der Datensatz sollte Anmerkungen zu den Bildern enthalten, die die abgebildeten Objekte und deren Positionen im Bild auflisten. Diese Positionsinformation könnte in Form von Begrenzungsboxen vorliegen. Das ist der erste Schritt dieser Arbeit. Das Hinzufügen einer Anmerkung bedeutet, eine Reihe von Koordinaten

zu begrenzen, die den Umriss des gewünschten Bereichs unter Verwendung einer geometrischen Rechteckform bilden. Sobald der Bereich ausgewählt ist, wird dann eine Klasse zugewiesen. Alle sichtbaren Objekte aus allen Bildern werden unter Verwendung von **LabelImg** ausführlich mit einer Bounding Box versehen. Abbildung 4.3 zeigt ein Beispiel für ein vollständig annotiertes Bild. **LabelImg** ist ein grafisches Werkzeug für Bildanmerkungen und Beschriftungsobjekte, die in Bildern Begrenzungsrahmen enthalten. Es ist in Python geschrieben. Mit dieser Software können verschiedene Formen auf dem Bild gezeichnet werden.



**Abbildung 4.3:** Annotiertes Bild

Die Ausgabe von LabelImg erfolgt in XML. Nach dem Beschriften vieler Bilder kann ein Skript die XML-Informationen über Koordinaten verwenden, um aus dem tatsächlichen Bild die Klassifiziererdaten für die fraglichen Regionen zu ziehen. Die Klassifizierungsdaten hängen von den Eigenschaften des Objekts ab, da einige Attribute aussagekräftiger sind als andere. Die entsprechenden XML-Dateien werden für jedes Bild generiert. XML-Dateien enthalten die Koordinaten der Begrenzungsrahmen, den Dateinamen, die Kategorie usw. für jedes Objekt im Bild. Abbildung 4.4 zeigt die XML-Datei des entsprechenden Bildes (in Abbildung 4.3).

```

<annotation>
  <folder>train</folder>
  <filename>Image_2018-05-04_14-17-32.jpg</filename>
  <path>C:\Users\nfrioui\Desktop\Bilder\train\Image_2018-05-04_14-17-32.jpg</path>
  <source>
    ...
    <database>Unknown</database>
  </source>
  <size>
    <width>1920</width>
    <height>1080</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Roboter</name>
    <pose>Unspecified</pose>
    <truncated>1</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>170</xmin>
      <ymin>124</ymin>
      <xmax>1059</xmax>
      <ymax>1080</ymax>
    </bndbox>
  </object>
  <object>
    <name>IO</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>449</xmin>
      <ymin>467</ymin>
      <xmax>529</xmax>
      <ymax>587</ymax>
    </bndbox>
  </object>
</annotation>

```

Abbildung 4.4: XML-Dateistruktur

## 4.4 Training

Um die Methoden zu erläutern, die zur Fertigstellung dieses Projekts verwendet wurden, wird es in kleinere Abschnitte unterteilt. Im ersten Schritt mussten die zuvor erwähnten Programme und Pakete installiert werden, dann musste ein Modell beschafft und trainiert werden. Zu diesem Training wurde ein Datensatz in Form von Bildern gesammelt und anschließend in das richtige Format übersetzt. Nach der Annotation der Bilder wurden die Datensätze in zwei Untergruppen unterteilt. 70% der Eingabedaten werden für Training und 30% für Testing verwendet. Der geteilte Datensatz von Bildern wurde als Eingaben im Pipeline-Prozess angegeben. Der Pipeline Prozess beginnt mit der ersten Herausforderung, dass den Bildern der Datensätze überprüft werden. Diese Pipeline wurde in automatisierten Tests umfasst und die großen Mengen von Bildern verarbeitet. Die XML-Dateien übereinstimmender Bilder wurden in CSV konvertiert und anschließend wurden die TFRecord-Datei erstellt. Weil die TensorFlow-Objekterkennungs-API keine CSV-Dateien als Eingabe akzeptiert, aber sie benötigt Records-Dateien zum Trainieren des Modells. In dieser Arbeit wurde ein Faster R-CNN Modell (faster-rcnn-inception-v2-coco) verwendet. Das Modell wählt für dieses Projekt, weil es auf Geschwindigkeit und auf Genauigkeit optimiert. Für den Echtzeit-Teil des Codes ist die Arbeitsgeschwindigkeit sehr wichtig, da sie mehrmals pro Sekunde durchlaufen werden muss.

TensorFlow-Training kann auf drei verschiedene Arten durchgeführt werden, lokal auf einer CPU oder einer GPU oder auf Googles Cloud-Servern. Das Training eines Modells kann sehr lange dauern, besonders wenn die verwendete Hardware nicht optimal ist. Für dieses Projekt wurde das Training lokal auf GPU-Computing durchgeführt. Um rechtzeitig zu sparen, wurde ein Checkpoint aus einem vortrainierten Modell verwendet.

Nachdem die Datensätze vollständig sind, wird eine Pipeline bereitgestellt und eine **labelmap** erstellt. Mit dem vorliegenden Datensatz ist das letzte zu konfigurierende Element eine **Trainingspipeline**, bei der es sich um eine Textdatei mit einer Konfiguration als Dateierweiterung handelt. Die letzte Aufgabe besteht darin, alles, was bisher konfiguriert wurde, zusammenzustellen und das Training durchzuführen.

## Labelmap

Jedem Dataset muss eine Zuordnung zugeordnet werden, die eine Zuordnung von String-Klassennamen zu Integer-Klassen-IDs definiert. Labelmap sollten mit ID 1 beginnen. Die Label-Map für diese Testdatei hat die folgende Struktur:

```
item {
  id: 1
  name: "NIO"
}
```

## PipelineKonfiguration

In diesem Abschnitt werden die Konfiguration der Hyperparameter und der Pfad zu den model checkpoints, Datei.records und der Datei.label-map erläutert. Die Dateien werden verwendet, um den Trainingsprozess zu konfigurieren, bei dem einige Hauptkonfigurationen geändert werden müssen. Mit der vorliegenden Aufzeichnung ist die letzte Konfiguration ein Training Pipeline, die eine Textdatei mit einer Konfiguration als Dateierweiterung ist. Eine Konfiguration wurde verwendete, die eher auf Geschwindigkeit als auf Genauigkeit basierte und die für die Wiedererkennung von Objekten entwickelt wurde, da sie am besten zu dem Ziel des Projekts passte. Für die quantitativen Testfälle wurde die Baseline-Architektur als Grundlage verwendet und verschiedene Hyperparameter wurden während jedes Testfalls geändert, um die Auswirkungen auf Genauigkeit und Laufzeit zu untersuchen. Die wichtigsten Einstellungen für das Experiment sind nachfolgend aufgeführt:

- **Num classes:** wurde Anzahl der Klassen im Dataset verwendet.
- **Number of epochs :** Die Anzahl der Epochen, für die das Netzwerk trainiert wurde. Dies wurde durchgeführt, um eine angemessene Anzahl von Epochen zu bestimmen, damit das Netzwerk korrekt konvergiert. Eine Epoche ist ein vollständiger Durchlauf des Datensatzes.
- **Gradient:** wurde an die Gleichrichter nach den Convolutional schichten angelegt.
- **Learning rate:** wurde von 0,001 bis 1,0 in Schritten um einen Faktor von 3 abgewechselt.
- **Batch size:** Verschiedene Batch Größen wurden getestet, von 1 bis 2000.

- **Image resizer:** Bildgrößen wurden verwendet.

Das Training kann gestartet werden. Es wird so lange ausgeführt. Die Modelle werden an verschiedenen Kontrollpunkten(checkpoints) gespeichert. Wann das Training des Modells abgeschlossen wurde, wurde der nächste Schritt des Pipeline-Prozesses, Evaluation des Modell mit Test-daten, durchgeführt.

## 4.5 Evaluation

Evaluation von Machine Learning-Algorithmus ist ein wesentlicher Teil eines jeden Projekts. Das Modell soll immer bewertet werden, um festzustellen, ob es das Ziel für neue Daten gut vorhersagen kann. Die Genauigkeit des Modells muss auf Test-daten überprüft werden.

Nach dem Training des Modells ist der nächste Schritt herauszufinden, wie effektiv das Modell ist, basierend auf Metrik unter Verwendung von Test-Daten. Verschiedene Leistungsmetriken werden verwendet, um verschiedene maschinelle Lernalgorithmen zu bewerten. Beispiel für eine Metrik zur Bewertung von Algorithmen des maschinellen Lernens ist Genauigkeit und Recall. Die Messwerte, die zur Bewertung Ihres maschinellen Lernmodells ausgewählt werden, sind sehr wichtig. In dieser Arbeit wurde Confusion-matrix verwendet.

Confusion-matrix ist eine der meist verwendeten Metriken, um die Korrektheit und Genauigkeit des Modells zu ermitteln. Confusion-matrix, wie der Name schon sagt, gibt eine Matrix als Ausgabe und beschreibt die komplette Leistung des Modells. Es wird für Klassifikationsprobleme verwendet, bei denen die Ausgabe aus zwei oder mehr Klassen bestehen kann.

Die Confusion-matrix ist eine Tabelle mit zwei Dimensionen ("Actual" und "Predicted") und Gruppen von "Klassen" in beiden Dimensionen. Die Klassifikationen sind Spalten und die vorhergesagten sind Zeilen. Abbildung 4.5 zeigt ein Beispiel für Confusion-matrix. Es besteht aus 2 Klassen "Positive" und "Negative".

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

**Abbildung 4.5:** Confusion-matrix

Diese Klassifizierung (oder Vorhersage) erzeugt vier Ergebnisse:

- **TP:** korrekte Positive Vorhersage.
- **FP:** falsche Positive Vorhersage.
- **TN:** korrekte Negative Vorhersage
- **FN:** falsche Negative Vorhersage.

Basismaßnahmen abgeleitet aus Confusion-matrix:

- **Genauigkeit:** wird berechnet als die Anzahl aller korrekten Vorhersagen dividiert durch die Gesamtanzahl der Datensätze. beschreibt, wie relevant die Erkennungsergebnisse sind:

$$\text{Genauigkeit} = \frac{TP + TN}{TP + TN + FN + FP}$$

- **Recall:** beschreibt den Prozentsatz relevanter Objekte, die mit dem Modell erkannt werden:

$$\text{Recall} = \frac{TP}{TP + TN}$$

After Evaluation wird als Letzter schritt von Pipeline Process "Export-Graph" durchgeführt. Der letzte Schritt besteht darin, den "frozen-inference-graph.pb" erzeugen. Diese frozen-inference-graph.pb-Datei enthält den Objekterkennungs-Klassifikator, die in der Produktion bereitgestellt werden kann.

## 4.6 Deployment in Production

Das Objekterkennung Modell wird bereits in der Produktion bereitgestellt.

In dieser Arbeit wurde Roboter als Anwendungsfall verwendet (Abbildung 4.8). Ein Modell Objekterkennung wurde trainiert, um die aufgebauten Fehler in Roboter zu erkennen. Es wurde 3 Klassen verwendet:

- **Roboter:** Zeigt das Roboter.
- **NIO (nicht in Ordnung):** Die Komponente enthält Fehler, ist nicht in Ordnung.
- **IO (In Ordnung):** Die Komponente ist in Ordnung

In dieser Arbeit wurde das Modell mit zwei Methoden in der Produktion bereitgestellt. Es wurde in Android und Web Anwendungen getestet.

### 4.6.1 Mobile Anwendung

Für die Implementierung von CNN-Modell im Android-Gerät, wird die Schnittstelle von "Tensorflow Lite" benutzt. Die Android Anwendung ist mit einem Streaming-Video von der Kamera entworfen und jeder Bildrahmen wird an das CNN-Modell zur Objekterkennung übergeben. Das Modell gibt die erkannten Objekte mit seiner Zuverlässigkeit zurück. Und dann werden die erkannten Ergebnisse in Echtzeit mit Kästchen (boxes) markiert. Es erhält die Objekterkennung und Konfidenz für jedes Objekt. Die gesamte App-Architektur ist wie in Abbildung 4.6 dargestellt.

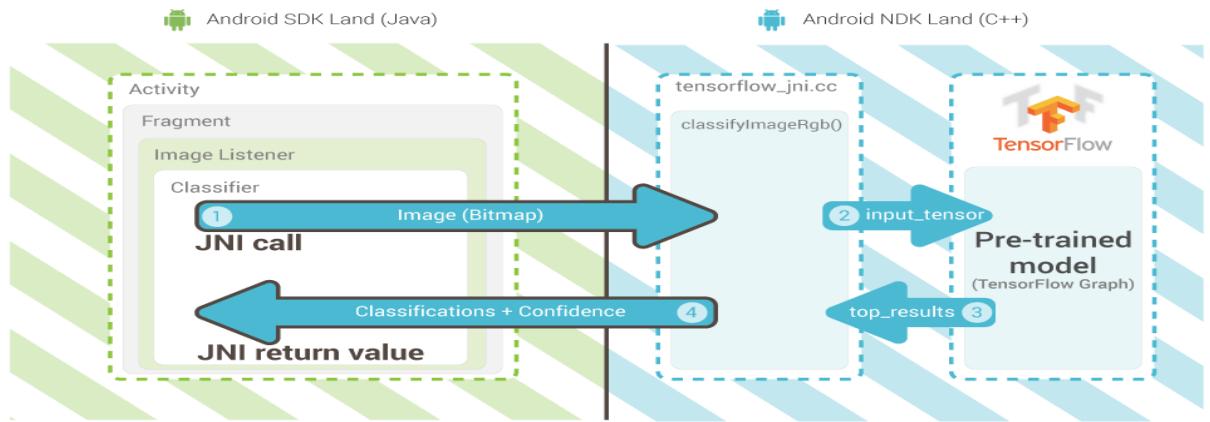


Abbildung 4.6: Android-App-Architektur[50]

Abbildung 4.7 zeigt einen Screenshot der Anwendung, wie es auf diesem Modell ausgeführt wird. Es identifiziert die Roboter mit 99% Wahrscheinlichkeit, und die Kabel Fehler(NIO) auch mit 99% Wahrscheinlichkeit.

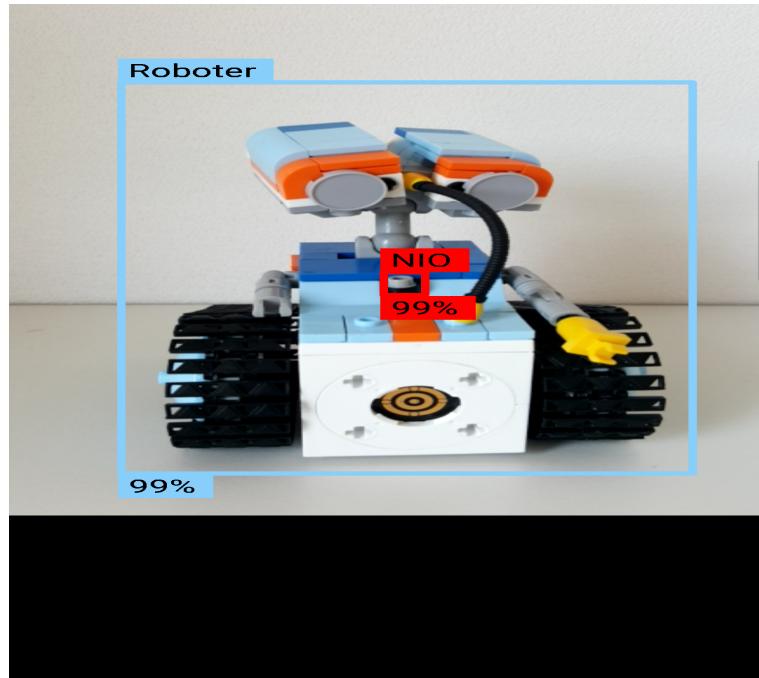
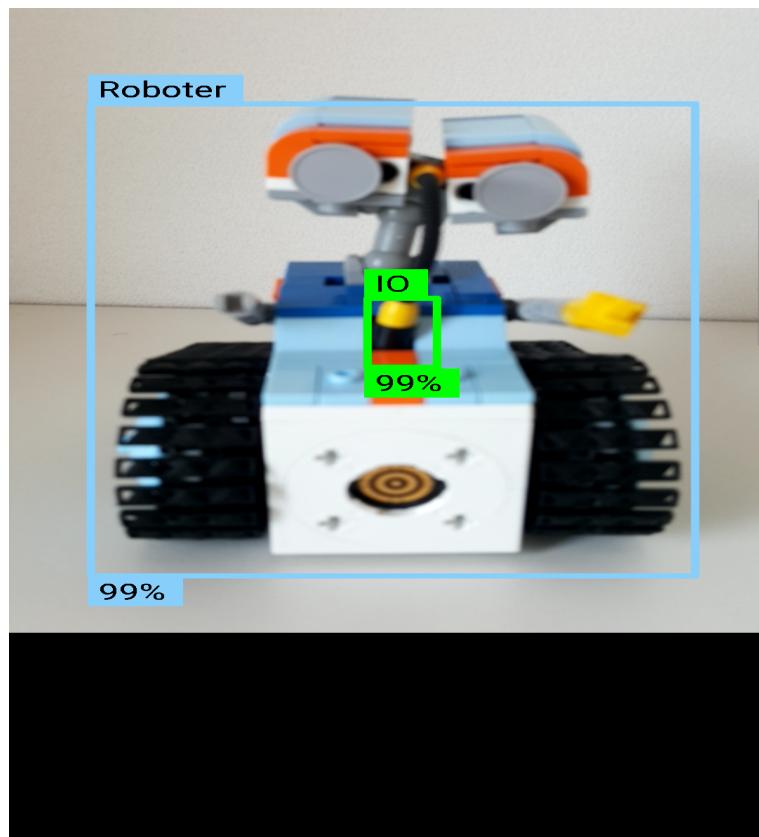


Abbildung 4.7: Android-App:NIO Objekterkennung

**Abbildung 4.8** zeigt, wie es auf diesem Modell ausgeführt wird. Es identifiziert die Roboter mit 99% Wahrscheinlichkeit, und die Kabel ist in Ordnung(IO) auch mit 99% Wahrscheinlichkeit.



**Abbildung 4.8:** Android-App:IO Objekterkennung

#### 4.6.2 Web Anwendung

Das Model wird mit Python durchgeführt. Die test Bilder wurden mit verschiedenen Verfahren wie ausgewählt Bilder, Webcam und Kamera aufgenommen. Um das Modell zu testen, werden Testbilder dem Modell zugeführt und ein Filteralgorithmus wird auf die Ausgabe des Modells angewendet. Der erste Schritt beim Filtern besteht darin, das Vertrauen jeder Erkennung zu berechnen, das durch das Produkt der Objektzuversicht und der Klassifizierungswerte gegeben ist. Dieser Filteralgorithmus gibt die Begrenzungsrahmen, Vertraulichkeiten und Klassen für die endgültigen Erkennungen in jedem Bild zurück. Danach werden die erkannten Ergebnisse mit Kästchen (boxes) markiert. Dann wurden die Updatebilder als Http Request an dem Template Django gesendet. Als Nächstes wurde das Bild in Template Django gezeigt.

**Abbildung 4.9** zeigt einen Screenshot der Anwendung. In diesem Beispiel wurde das Testbild von einer externen Kamera aufgenommen. Die Anwendung wird die original Bilder und die erkannten Objekte gezeigt. Endlich wird das Ergebnis gegeben, ob der Roboter in Ordnung ist oder nicht. In diesem Beispiel werden das Roboter mit 99% Wahrscheinlichkeit und Kabel IO mit 99% Wahrscheinlichkeit erkannt. Das Ergebnis in diesem Beispiel ist IO (in Ordnung) Bild.

**Abbildung 4.10** zeigt einen Screenshot der Anwendung. In diesem Beispiel wurde das Testbild von Webcam. Die Roboter wird mit 99% Wahrscheinlichkeit erkannt und die Kabel Fehler(NIO) wird mit 73% Wahrscheinlichkeit identifiziert. Das Ergebnis in diesem Beispiel ist NIO (nicht in Ordnung) Bild.

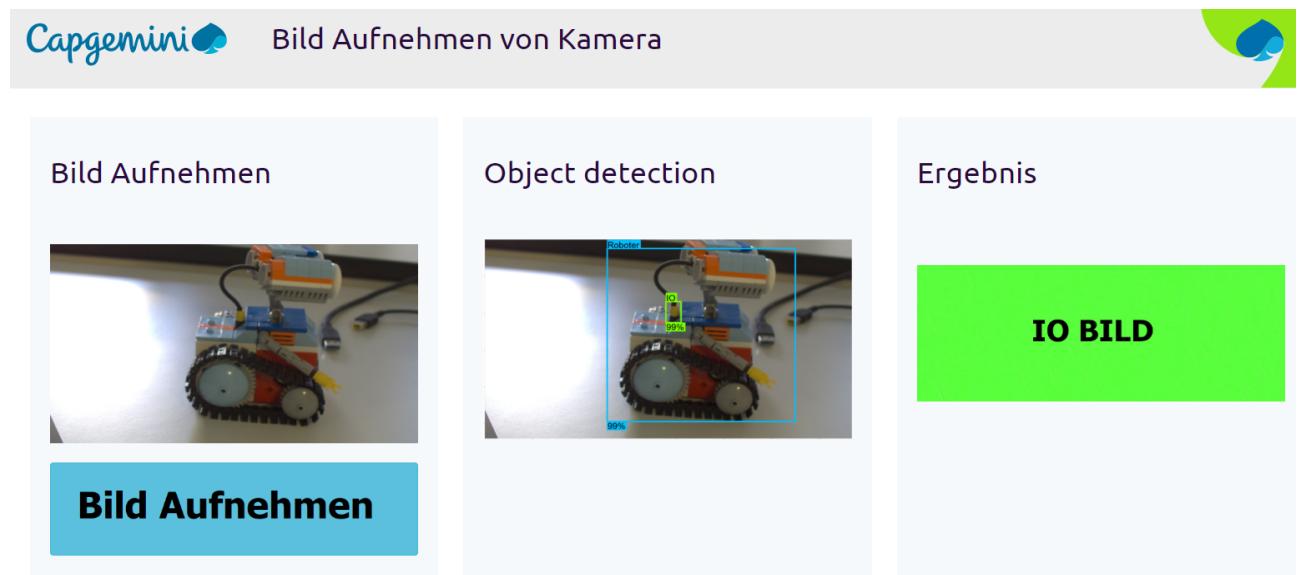


Abbildung 4.9: Screenshot der Anwendung: Bild aufnehmen von Kamera



Abbildung 4.10: Screenshot der Anwendung: Bild aufnehmen von webcam

## 4.7 Zusammenfasung

Um mit einem CNN-Modell in der Praxis zu experimentieren, wurde eine funktionierende Tensorflow-Implementierung von Faster R-CNN erstellt. Es wurde bemerkt, dass der schwierigste Teil der Implementierung eines Deep-Learning-Systems darin besteht, die Trainingsdaten zu sammeln und das Training selbst durchzuführen. Das Training wurde mit Pipeline Process durchgeführt. Deployment Pipeline wurde verwendet, um den Trainingsprozess zu automatisieren und Datensätze zu analysieren. Die Trainingszeit kann durch Verwendung eines vortrainierten Modells verkürzt werden. Selbst wenn das endgültige Modell nicht die gleichen Objektklassen wie die Benchmark-Daten aufweist, sind visuelle Probleme universell genug, um von Detektoren zu profitieren, die für ein anderes Problem ausgebildet sind. Die Tools sind recht unberechenbar hinsichtlich der Interoperabilität von Softwareversionen und Hardware, insbesondere wenn die Implementierung auf einer GPU versucht wird.

# Kapitel 5

## Analyse der Ergebnisse und Diskussion

In diesem Kapitel werden die Ergebnisse der Experimente vorgestellt. Zuerst wird die allgemeine Training Ergebnisse des Modells Faster-R-CNN diskutiert. Als nächstes wird die Konfusionsmatrix der Schrittauswertung analysiert. Anschließend wird Diskussion erstellt.

### 5.1 Analyse der Ergebnisse

#### 5.1.1 Ergebnisse des Trainings

Die Ergebnisse der Trainingsphasen können mit TensorFlows Visualisierungsplattform TensorBoard eingesehen werden. Die TensorBoard-Seite bietet Informationen und Grafiken, die zeigen, wie das Training voranschreitet.

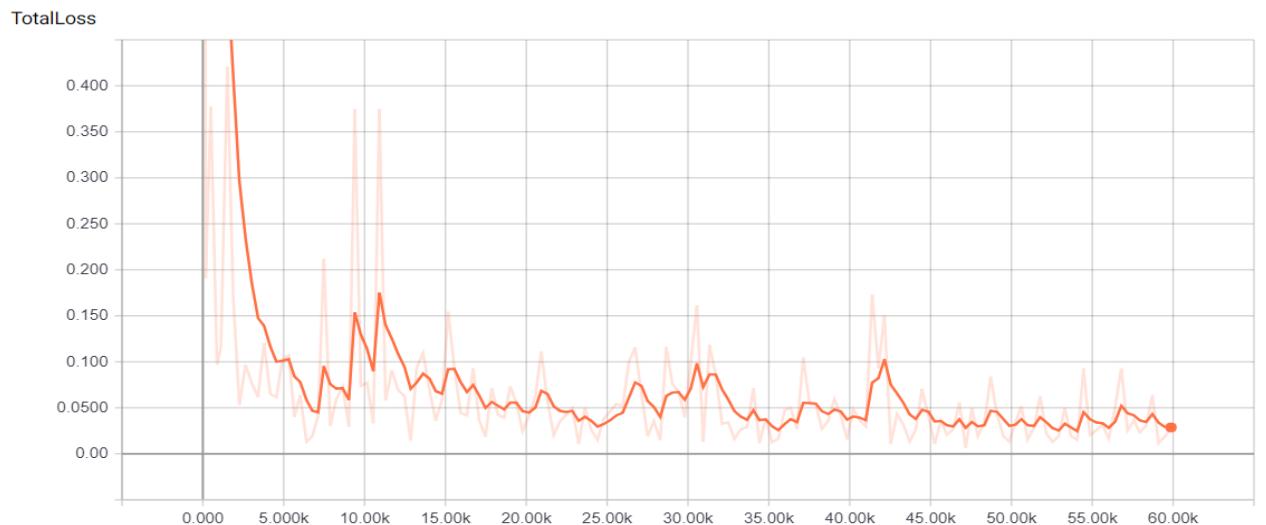
Die Bewertung der Trainingsschritte erfolgt über das TensorBoard, das Diagramme über das Training und seine Leistung liefert, durch Betrachten des Diagramms über den Gesamtverlust (Total Loss) und Präzision(Precision). Ein wichtiger Graph ist der Verlustgraph, der den Gesamtverlust des Klassifikators über die Zeit zeigt.

TensorBoard bietet die Möglichkeit, die Testphase auch auf den Bildern zu beobachten, um einen besseren Überblick über die Genauigkeit des Modells zu erhalten und wie wirklich das Modell lernt.

Das Modell wurde mit 200 Bildern trainiert. Das Training wurde nach 60000 Zeitschritten gestoppt, wenn die mittlere durchschnittliche Präzision etwas ausgeglichen war. Das Training dauert über 5 Stunden mit GPU. **Abbildung 5.1**

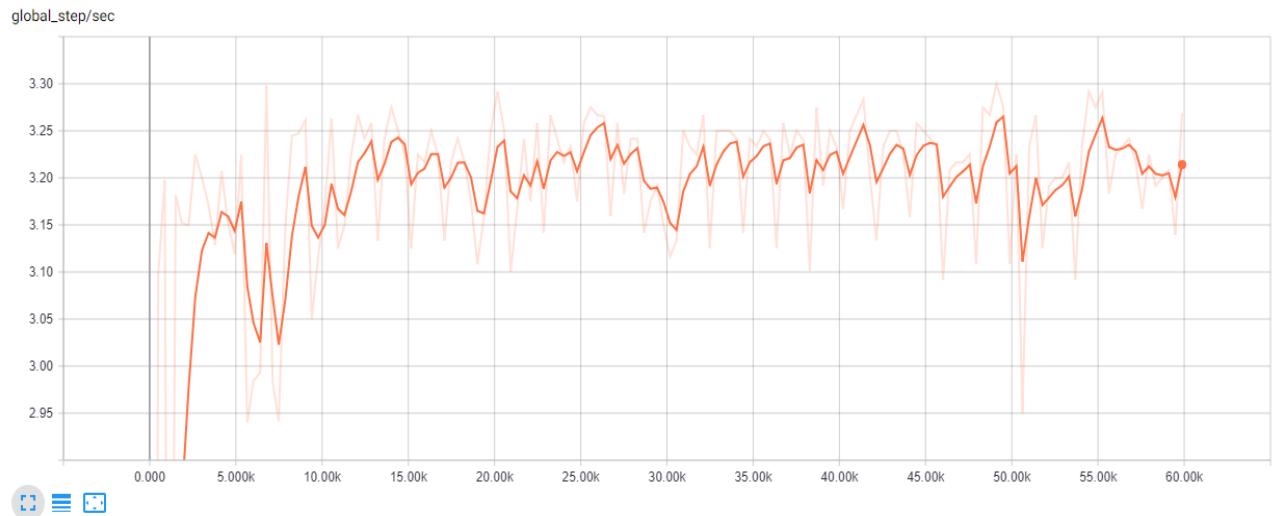
zeigt die mittlere durchschnittliche Präzision über die Zeit (steps). Anhand von Grafiken wie diesen kann entschieden werden, wann das Training ausreichend ist. Die Präzision ist nicht so steil, da es auf den Bildern getestet wird, mit denen das vortrainierte Modell keine Erfahrung hat.

Der Gesamtverlust stabilisiert sich schnell aufgrund der Verwendung eines vortrainierten Modells. **Abbildung 5.2** zeigt den Gesamtverlust über die Zeit (steps) und ist einer von mehreren Graphen, die in TensorBoard zu sehen sind.



**Abbildung 5.1:** Gesamtverlust beim Trainingsmodell

Die Modelle wurden bis zu 60000 Zeitschritten trainiert. Die Präzisionswerte für das Modell zeigten einen zunehmenden Trend bis etwa 7000 Zeitschritte und ebbten sich danach wieder ab (Abbildung 5.2).



**Abbildung 5.2:** Präzision beim Trainingsmodell

### 5.1.2 Ergebnisse der Evaluation

Die relevantesten Evaluierungsmetriken für diese Anwendung sind die Genauigkeit und der Abruf(recall).

- **Genauigkeit:** beschreibt, wie relevant die Erkennungsergebnisse sind:

$$\text{Genauigkeit} = \frac{TP}{\text{Gesamtanzahl}}$$

- **Recall:** beschreibt den Prozentsatz relevanter Objekte, die mit dem Modell erkannt werden:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Die Erkennungsergebnisse wurden überprüft, indem jedes Bild durchlaufen und die **true positive**, **false positive** und **false negative** Bewertungen gezählt wurden.

Um das Ziel dieser Überprüfung abzuschätzen, ob ein Objekt überhaupt erkannt wurde, wurde eine sehr liberale Definition von **TP** verwendet: Die richtige Erkennung wurde als **TP** klassifiziert, wenn mindestens 50% des Objekts als innerhalb der vorhergesagten Begrenzungsbox liegend geschätzt wurde. Die falsche Erkennung wurde als **FP** klassifiziert. Ein Objekt ohne vorhergesagte Bounding-Boxen

wird als **FN** betrachtet. Das Evaluierungs-Protokoll ignoriert nicht erkannte Objekte.

Das Modell wurde mit 75 Bildern Evaluiert. Die Bewertung der Evaluationsschritte wurde in **Abbildung 5.3** gezeigt.



**Abbildung 5.3:** Confusion Matrix der Evaluationsschritte

75 Bilder von Roboter (45 **NIO** Bilder und 30 **IO** Bilder) wurden verwendet, um das Modell zu bewerten.

**NIO** (das Roboter ist nicht in Ordnung)

**IO** (Das Roboter ist in Ordnung).

Das Ergebnis der Bewertung war nicht perfekt. Das Modell hat einige Fehler gemacht:

- Das Modell hat nur 70 Bilder erkannt.
- 4 “NIO“ Bilder wurden als “IO“ klassifiziert

- 3 IO-Bilder wurden falsch identifiziert.
- 6 NIO und 2 IO wurden nicht erkannt.
- Nur 60 Bilder von insgesamt 75 Bildern wurden richtig klassifiziert.
- **Genauigkeit:**

$$\text{Genauigkeit} = \frac{70 + 35 + 25}{137} = 0.94$$

- **Recall:**

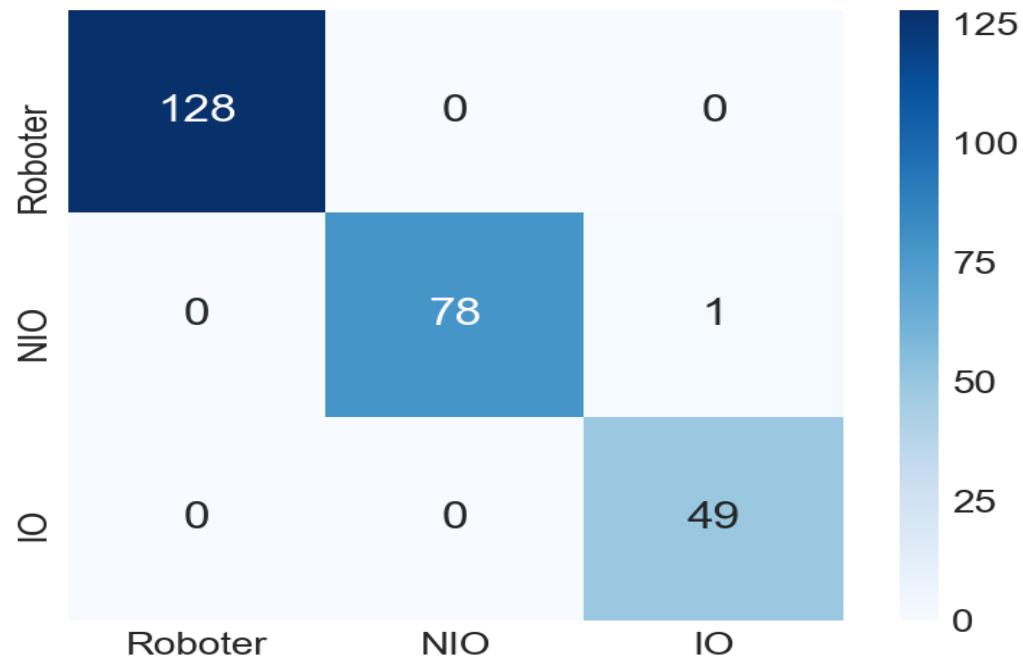
$$\text{Recall NIO} = \frac{35}{39} = 0.89$$

$$\text{Recall IO} = \frac{25}{28} = 0.89$$

Aus der Analyse der Ergebnisse ergab sich, dass das Modell nicht optimal war. Daher wurde das Modell weiter trainiert. 150 neue Bilder für das Training und 55 neue Bilder für die Auswertung wurden zu Pipeline hinzugefügt. Das Modell wurde mit insgesamt 350 Bildern und bis zu 90 000 Zeit schritten weiter trainiert, um die Leistung in der Objekterkennung zu verbessern.

After das Training wurde das Modell mit neuen Test-Datensätzen Evaluierter. Die Bewertung der neuen Evaluationsschritte wurde in **Abbildung 5.4** gezeigt.

Das Modell wurde mit 130 Bildern(80 **NIO** Bilder und 50 **IO** Bilder) wieder Evaluierter



**Abbildung 5.4:** Confusion Matrix der neuen Evaluationsschritte

Das Ergebnis der Bewertung:

- Das Modell hat 128 Bilder erkannt.
- Nur ein “NIO“ Bild wurde als “IO“ klassifiziert
- Alle erkannten IO-Bilder wurden korrekt identifiziert.
- **Genauigkeit:**

$$\text{Genauigkeit} = \frac{128 + 78 + 49}{256} = 0.99$$

- **Recall:**

$$\text{Recall NIO} = \frac{78}{79} = 0.98$$

$$\text{Recall IO} = \frac{49}{49} = 1.00$$

### 5.1.3 Zusammenfassung

Hinsichtlich der Präzision waren die Ergebnisse vielversprechend. Es wurde gezeigt, wie ein Modell, das auf allgemeine Bilddaten trainiert wurde, verwendet werden kann, um Objekte in einer bestimmten Aufgabe zu erkennen, was die Anpassungsfähigkeit der Methoden demonstriert. In vielen Fällen erkannte Faster R-CNN mehr Objekte als die Annotation der Originaldaten markiert hatten. Diese wurden als falsch-positiv bezeichnet, obwohl sie durch visuelle Inspektion eindeutig die richtige Objektklasse hatten.

Auf der anderen Seite hat das Modell auch einige tatsächliche Fehler gemacht. Außerdem demonstrierte das Modell die Wichtigkeit der Nachbearbeitung der Datensätze. Datensätze hatten einen signifikanten Einfluss auf die durchschnittliche Genauigkeit. Die Wahl des Modells beeinflusst sowohl Geschwindigkeit als auch Genauigkeit des Objekterkennungssystems. Im Allgemeinen verursachen überlappende Objekte und Begrenzungsrahmen Probleme für Objekterkennung.

## 5.2 Diskussion

Deep Learning ist ein sehr guter Ansatz, um kognitiven Systemen die Fähigkeit zu geben, Objekte in einer Umgebung zu verstehen und zu interpretieren. Dennoch, die Anwendung von Deep Learning bietet auch Nachteile und Einschränkungen.

Obwohl die gesammelten Datensätze zu den wichtigsten Voraussetzungen für das Faster-RCNN-Modell gehören, verkörpern sie auch einen limitierenden Faktor. Für den Anwendungsfall in dieser Arbeit ist der limitierende Faktor nicht primär die Anzahl der Trainingsbilder, sondern die Vielfalt verschiedener Objekte mit unverwechselbaren und einzigartigen Attributen, die für das Training genutzt wurden. Wenn ein Attribut oder eine Bezeichnung in dem anfänglichen Datensetze nicht vorhanden ist, ist es nicht möglich, diese Klasse für unbekannte Objekte vorherzusagen. In einem Datensetze muss jede dargestellte Klasse eines Objekts oder Attributs mehrere verschiedene Beispiele enthalten, um sicherzustellen, dass die richtigen Beziehungen erlernt werden können.

### 5.2.1 Datensätze

Die Qualität der Trainingsdaten ist gut angesichts der allgemeinen Lichtverhältnisse und der Auflösung der Bilder. Die Trainingsdaten bestehen aus Bildern Notizen unter verschiedenen Lichtverhältnissen, die den Bedürfnissen und Anforderungen entspricht.

Die Herausforderung der Trainingsdaten ist das Fehlen von Bildern aus einer Umgebung, die mit dem Endanwendungsfall des Modells in Zusammenhang steht, der sich in einer Umgebung mit mehreren Noten und nicht perfekter Beleuchtung befindet. Das Sammeln einer erheblichen Anzahl von Bildern in solchen Umgebungen könnte daher die Leistung des Modells verbessern. Dies würde andere Arten der Datenerfassung erfordern, da Bilder mit solchen Merkmalen schwierig zu finden waren. Allerdings könnte die geometrische Einfachheit der Note bereits nahe an die aktuellen Daten optimal angepasst werden und eine signifikant erhöhte Leistung kann nur durch Erhöhung der Komplexität des Modells erreicht werden.

### 5.2.2 Mehrere Erkennungen

Überlappende Objekte und Begrenzungsboxen verursachen eindeutig Probleme für die getestete Objekterkennungs-Pipeline. Zu allererst macht es die Überlappung der wahren Objekte schwierig, die Begrenzungsboxen eindeutig zu platzieren. Die Begrenzungsrahmen werden durch die Region-Generierungsmethode ausgewählt und sollten typischerweise das gesamte Objekt umgeben. Wenn das Objekt jedoch verdeckt ist, können die Dimensionen des Objekts nicht bekannt werden, bevor es klassifiziert wurde.

Der getestete Faster-R-CNN ist oft gut genug, um ein verdecktes Objekt korrekt von einem Teilobjekt zu klassifizieren. Das Erkennen von Teilobjekten kann jedoch Probleme verursachen. Nicht-maximale Unterdrückung wurde in den Experimenten verwendet, um Mehrfachdetektionen desselben Objekts zu entfernen. In **Abbildung 5.5** wird ein Beispiel für Mehrfachobjekt-Erkennungen gezeigt, das nicht entfernt wurde. Faster-R-CNN gibt diesen Erkennungen einen hohen Wahrscheinlichkeitswert, was bedeutet, dass die Partialtöne ganze Objekte übertreffen können, die schwieriger zu erkennen sind.

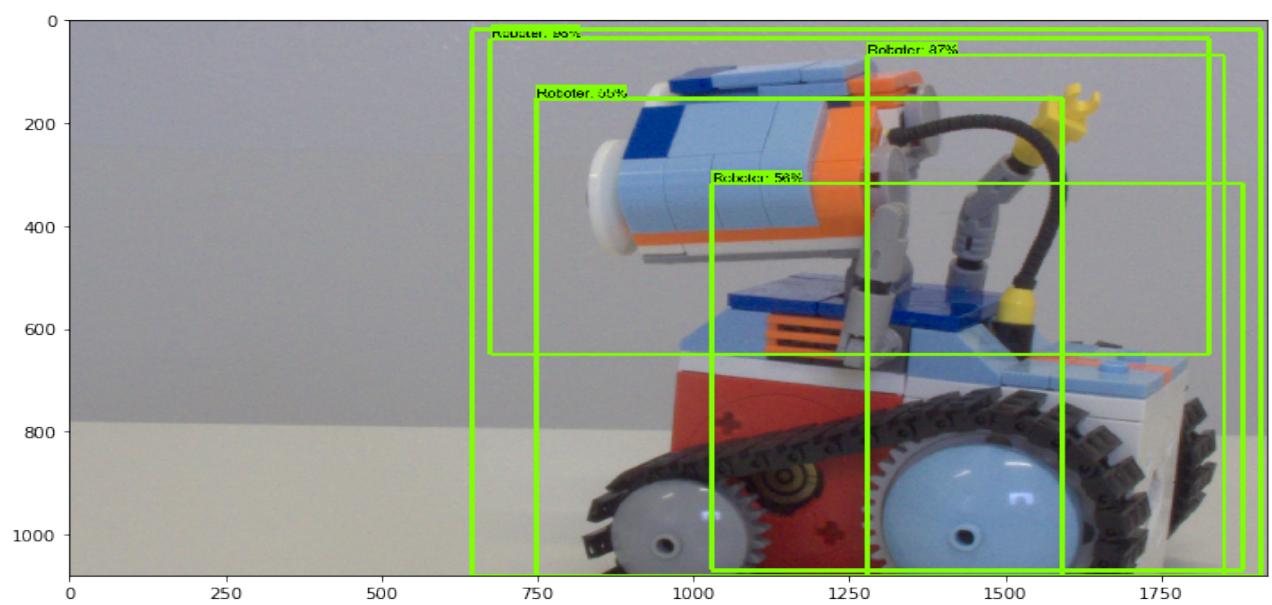


Abbildung 5.5: Beispiele für Mehrfachobjekt-Erkennungen

# Kapitel 6

## Zusammenfassung

Eine sehr hohe Leistung der Objekte trainierten Modelle wurde erwartet und erreicht, weil Deep-Learning-Algorithmen für viele Objekterkennungsaufgaben nahezu optimale Ergebnisse liefern. Eine schwierige Aufgabe besteht darin, Attribute aus dem relativ kleinen Datensatz zu lernen und die erlernten Beziehungen auf unbekannte Daten zu verallgemeinern. Auf dem Test-Set erreichten das Faster-R-CNN Modell, das an Objekten und Kontexten trainiert wurden, Genauigkeiten zwischen 68 und 99 Prozent.

Ein vortrainiertes Faster-R-CNN Objekterkennungsmodell wurde zur Erkennung von dem aufgebauten Fehler in Robotern mit einem manuell markierten Datensatz getestet. Das Modell wurde unter Verwendung der von der TensorFlow Object Detection API bereitgestellten Funktionen erstellt, einem frei verfügbaren Framework zur Objekterkennung. Deployment Pipeline wurde verwendet, um den Trainingsprozess zu automatisieren und Datensätze zu analysieren. Aus den Ergebnissen wurden folgende Schlussfolgerungen gezogen: Zuerst funktionierte das Modell einigermaßen gut. Zweitens, die Modellleistung, die als Genauigkeit bewertet wurde, verbesserte sich, als das Modell von 20 000 auf 100 000 Zeitschritte trainiert wurde. Die optimale Anzahl von Zeitschritten wird wahrscheinlich bei 100 000 Zeitschritten liegen. Drittens: Das Modell Faster R-CNN könnte besser zur Erkennung kleiner Objekte geeignet sein.

Die Objekterkennung Verfahren wurden vorgeschlagen und verglichen, die die Vorteile von maschinellen Lernalgorithmen mit semantischen Schlusstechniken kombinieren. Die Ergebnisse bestätigen, dass die Systeme die allgemeine Erkennungsge-

nauigkeit und Lernfähigkeit von Faster-RCNN-Modell verbessern. Bei bekannten und unbekannten Objekten verbesserte sich die Erkennungsleistung auf 99 % bzw. auf 85%. Selbst wenn die Vorhersagen der Klassifikatoren falsch sind, ermöglicht das System, das gezeigte Objekt zu erkennen.

Schließlich erfordert das gleichzeitige Erkennen von Objekten und kontextuellen Informationen mit Deep Learning Rechenleistung. Aufgrund der technischen Ausstattung war dieses Thema für diese Forschungsexperimente kein Problem. Für Computer mit sehr begrenzten Energieressourcen oder leistungsschwachen CPUs / GPUs ist es jedoch nicht möglich, Deep Learning zu verwenden. Spezialisierte Hardware und aktuelle Fortschritte in effizienten Computerarchitekturen für deep learning werden dieses Problem in naher Zukunft lösen.

# Quellenverzeichnis

## Literatur

- [1] I. Sutskever A. Krizhevsky und Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Advances in Neural Information Processing Systems 25*. 2012, S. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf> (siehe S. 27).
- [2] A. Nguyen, J. Yosinski, and J. Clune. *Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images*. 2014. URL: <https://arxiv.org/abs/1412.1897/> (siehe S. 10, 28).
- [3] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, (siehe S. 6).
- [4] B.A. Opelt, A. Pinz, and A. Zisserman. *Learning an alphabet of shape and appearance for multi-class object detection*. International Journal of Computer Vision, 2008 (siehe S. 30, 31).
- [5] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2011 (siehe S. 9).
- [6] C. Szegedy, S. Reed, P. Sermanet, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. 2015 IEEE Conference on Computer Vision und Pattern Recognition (CVPR), S. 1–12 (siehe S. 27).
- [7] Ch. H. Lampert, H. Nickisch, and S. Harmeling. *Learning to detect unseen object classes by betweenclass attribute transfer*. IEEE, 2009 (siehe S. 28).
- [8] M. Copeland. *What is the difference between artificial intelligence, machine learning, and deep learning?* July 2016. URL: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/> (siehe S. 29).
- [9] D. K. Prasad. *Survey of the problem of object detection in real images*. International Journal of Image Processing (IJIP), 2012 (siehe S. 29).

- [10] *D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search.* 2016. URL: <http://dx.doi.org/10.1038/nature16961/> (siehe S. 28).
- [11] *D.E. Rumelhart and J.L. McClelland. Parallel Distributed Processing.* MIT Press Cambridge, 1986. URL: <https://dl.acm.org/citation.cfm?id=104279/> (siehe S. 26).
- [12] P. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press, 2017 (siehe S. 6).
- [13] *H. W. Lin and M. Tegmark. Why does deep and cheap learning work so well ? ArXiv e-prints.* August 2016. URL: <https://arxiv.org/abs/1608.08225/> (siehe S. 28).
- [14] J. Humble. *Continuous delivery vs continuous deployment.* 2014. URL: <http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/> (siehe S. 22).
- [15] J. Humble und D. Farley. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Addison-Wesley Signature Series.* 2010 (siehe S. 22, 23).
- [16] *I. Awaad, G. K. Kraetzschmar, and J. Hertzberg. Finding Ways to Get the Job Done: An Affordance-Based Approach.* International Conference on Automated Planning und Scheduling (ICAPS), 2014. URL: <https://pdfs.semanticscholar.org/6f07/cc069cabad52ffb1faf663f38319585a7d23.pdf> (siehe S. 28).
- [17] *I. Goodfellow, Y. Bengio and A. Courville. Deep Learning.* MIT Press, 2016. URL: <http://www.deeplearningbook.org> (siehe S. 6, 8, 12, 27).
- [18] *I. Sheshadri, D. Hoiem, A. Farhadi. Describing Objects by their Attributes.* IEEE Conference on Computer Vision und Pattern Recognition, 2012 (siehe S. 28).
- [19] *J. Shotton. Contour and texture for visual recognition of object categories.* 2007. URL: <https://www.microsoft.com/en-us/research/publication/contour-and-texture-for-visual-recognition-of-object-categories-2/> (siehe S. 30).
- [20] *J. Wang, Q. Tao. Machine Learning: The State of the Art.* IEEE Intelligent Systems, 2008 (siehe S. 27).
- [21] *K. He, X. Zhang, S. Ren, and J. Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.* IEEE International Conference on Computer Vision (ICCV), 2015. URL: <https://arxiv.org/abs/1502.01852/> (siehe S. 17).

Masterarbeit,

TH

Brandenburg

- Naoufel  
Frioui
- Machine  
Learning
- [22] *K.Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position.* Springer-Verlag, URL: <https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf/> (siehe S. 13).
  - [23] *L. Breiman. Statistical Modeling: The Two Cultures.* The Institute of Mathematical Statistics, 2001. URL: <https://projecteuclid.org/euclid.ss/1009213726> (siehe S. 27).
  - [24] *L. N. Long, and A. Gupta. Scalable massively parallel artificial neural networks. Journal of Aerospace Computing, Information, and Communication.* 2008 (siehe S. 8).
  - [25] *Li Fei Fei. If we want machines to think, we need to teach them to see.* 2015. URL: <https://www.wired.com/brandlab/2015/04/fei-fei-li-want-machines-think-need-teach-see/> (siehe S. 28).
  - [26] *NVIDIA. CUDA C Best Practices Guide .* 2015. URL: <https://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html> (siehe S. 34).
  - [27] *O. Russakovsky and Li Fei-fei. Attribute learning in large-scale datasets.* URL: <http://ai.stanford.edu/~olga/papers/eccv10workshop-Attributes.pdf> (siehe S. 28).
  - [28] *O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator.* IEEE Computer Society Conference on Computer Vision und Pattern Recognition, 2015 (siehe S. 28).
  - [29] *Q.Z. Wu, Y. LeCun, L. D. Jackel, and B.S. Jeng. on-line recognition of limited vocabulary chinese character using multiple convolutional neural networks.* In Proc. of the 1993 IEEE International Symposium on circuits and systems, volume 4, pages 2435–2438. IEEE. 1993 (siehe S. 1).
  - [30] *R. Girshick. Fast R-CNN.* IEEE International Conference on Computer Vision (ICCV). 2015 (siehe S. 18–20).
  - [31] *R. Girshick, J. Donahue, T. Darrell and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation.* IEEE conference on computer vision und pattern recognition, 2014. URL: <https://arxiv.org/abs/1311.2524/> (siehe S. 18).
  - [32] *R. O. Duda, P. E. Hart, and D. G. Stork. Pattern classification.* Wiley, (siehe S. 5).
  - [33] *R. Rojas. Neural networks: a systematic introduction.* Springer-Verlag Berlin, Heidelberg, 1996 (siehe S. 8).

- Naoufel Frioui
- Machine Learning
- [34] *R. Szeliski. Computer vision: algorithms and applications.* Springer Science Business Media, 2010 (siehe S. 10, 17).
- [35] *S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* In Advances in neural information processing systems., 2015. URL: <https://arxiv.org/pdf/1506.01497.pdf/> (siehe S. 20).
- [36] *TensorFlow.* February 2017. URL: <https://www.tensorflow.org/> (siehe S. 34, 35).
- [37] *Tensorflow Lite.* 2018. URL: <https://www.tensorflow.org/%20mobile/tflite/> (siehe S. 36).
- [38] *V. Escorcia, J. C. Niebles, and B. Ghanem. On the relationship between visual attributes and convolutional networks.* IEEE Conference on Computer Vision und Pattern Recognition, 2015 (siehe S. 28).
- [39] *V. Ferrari and A. Zisserman. Learning visual attributes.* Curran Associates, Inc., 2008. URL: <http://papers.nips.cc/paper/3217-learning-visual-attributes.pdf> (siehe S. 28).
- [40] *W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A. C. Berg. SSD: Single shot multibox detector.* In European Conference on Computer Vision, 2016. URL: <https://arxiv.org/abs/1512.02325/> (siehe S. 21).
- [41] *W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity.* Bulletin of mathematical biophysics. Cambridge University Press, (siehe S. 8).
- [42] *Y. Bengio. Learning Deep Architectures for AI.* January 2009, S. 1–127. URL: <https://www.iro.umontreal.ca/~lisa/pointeurs/TR1312.pdf/> (siehe S. 28).
- [43] *Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition.* 1998. URL: [http://vision.stanford.edu/cs598\\_spring07/papers/Lecun98.pdf/](http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf/) (siehe S. 12, 13).
- [44] *Y. Lecun, L.D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning.* In E. Sanchez-Sinencio, and C. Lau (Eds.), Artificial neural networks (pp. 463-468). 1992 (siehe S. 1).
- [45] *Z. Si, H. Gong, Y. N. Wu, and S. C. Zhu. Learning mixed templates for object recognition.* IEEE Conference on Computer Vision und Pattern Recognition, 2009 (siehe S. 30).

Masterarbeit,

TH

Brandenburg

64

## Online-Quellen

- [46] . URL: <https://www.datacamp.com/community/tutorials/deep-learning-python/> (siehe S. 9).
- [47] . URL: <https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html/> (siehe S. 10).
- [48] . URL: <https://www.pyimagesearch.com/2014/06/09/get-deep-learning-bandwagon-get-perspective/> (siehe S. 12).
- [49] . URL: <https://pipeline.ai/> (siehe S. 33).
- [50] . URL: <http://jalammar.github.io/> (siehe S. 45).
- [51] *The TensorFlow Authors (2017) TensorFlow Object Detection API.* 2017. URL: [https://github.com/tensorflow/models/tree/master/research/object\\_detection/](https://github.com/tensorflow/models/tree/master/research/object_detection/) (siehe S. 36, 37).

**Ehrenwörtliche Erklärung**

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Ort, Datum

Unterschrift

Naoufel Frioui