Fachhochschule Westküste
Hochschule für Wirtschaft & Technik

HAW HAMBURG

Masterarbeit

Andreas Bloch

Entwicklung einer mit 3D-Modellen trainierten Multi-Objekt Detektion für kollaborierende Roboter

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Computer Science and Engineering
Department of Information and Electrical Engineering

Andreas Bloch

Entwicklung einer mit 3D-Modellen trainierten Multi-Objekt Detektion für kollaborierende Roboter

Masterarbeit eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Fachbereich Technik
der Fachhochschule Westküste
und
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Lutz Leutelt
Zweitgutachter: Prof. Dr. rer.nat. Kristina Schädler

Eingereicht am: 12. November 2020

Andreas Bloch

Thema der Arbeit

Entwicklung einer mit 3D-Modellen trainierten Multi-Objekt Detektion für kollaborierende Roboter

Stichworte

Objektdetektion, Deep Learning, Kollaborative Arbeit, Vormontagelinie, Mini-Factory, 3D-Modelle, 3DExperience-Plattform, Creative Experience

Kurzzusammenfassung

Die Mini-Factory ist eine Vormontagelinie, die kollaborativ die Umsetzung verschiedener Anwendungsfälle ermöglicht. Für die Mini-Factory wird eine Objektdetektion entwickelt, die auf Deep Learning Ansätzen basiert. Zum Vergleich werden einstufige- und zweistufige Deep Learning Ansätze genutzt. Der benötigte Datensatz wird mit der 3DExcite Creative Experience App basierend auf 3D-Modellen erzeugt.

Andreas Bloch

Title of Thesis

Development of a multi-object detection trained with 3D models for collaborating robots

Keywords

Object detection, Deep learning, collaborative work, pre-assembly line, Mini-Factory, 3D-models, 3DExperience Platform, Creative Experience

Abstract

The Mini-Factory is a pre-assembly line that enables the implementation of various applications in collaborative fashion. For the Mini-Factory, an object detection system based on deep learning approaches is being developed. For comparison, one-step and two-step deep learning approaches are used. The required data set is generated with the 3DExcite Creative Experience App based on 3D models.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AP Average Precision.

API Application Programming Interface.

CAD Computer Aided Design.

CNN Convolutional Neural Network.

FN False Negative.

FP False Positive.

FPS Frames per seconds.

GAN Generative Adversarial Network.

HOG Histogram of Oriented Gradients.

ILSVRC ImageNet Large Scale Visual Recognition.

IoU Intersection over Union.

LBP Local Binary Pattern.

mAP Mean Average Precision.

NMS Non Maximum Suppression.

ReLu Rectified Linear Unit.

ROS Robot Operating System.

RPN Regional Proposal Network.

SBC Single Board Computer.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

UR Universal Robot.

YOLO You Only Look Once.

ZAL Zentrum für Angewandte Luftfahrtforschung.

1 Einleitung

1.1 Motivation

Jedes Lebewesen, das auf diesem Planeten lebt, nutzt unterschiedliche Sensoren seines Körpers, um seine Umwelt wahrzunehmen. Auf diese Weise gelingt es den Lebewesen Herausforderungen zu bewältigen. Dabei können die Herausforderungen unterschiedlicher Art sein. Potentielle Beute, Hindernisse oder andere in der Umwelt befindlichen Objekte werden durch die Lebewesen unterschiedlich aufgenommen und liefern dem Empfänger wichtige Informationen über seine Umwelt. Auf Grundlage der durch diese Weise generierten Daten ist es den Lebewesen möglich Beutetiere zu fangen, sich vor möglichen Fressfeinden zu schützen oder sich zu orientieren.
Lebewesen wie die Fledermaus nutzen zum Generieren der benötigten Daten einen lauten Schrei im Ultrafrequenzbereich. Ein ähnliches Prinzip wie es die Fledermaus nutzt, wird heutzutage in der Schifffahrt angewandt. Mit Hilfe des sogenannten Echolots ist es der Schifffahrt möglich sich zu orientieren und Gefahren, die durch Objekte entstehen frühzeitig zu erkennen und zu vermeiden.
Ein ganz anderes Prinzip zur Informationsgewinnung über seine Umwelt nutzt der Mensch. Für die Wahrnehmung seiner Umwelt nutzt der Mensch seine Augen. Diese liefern ihm Informationen über Hindernisse, über Oberflächen des Untergrundes auf denen sich der Mensch zur Zeit bewegt und Objekte die sich im Sichtfeld des Menschen befinden. Der Mensch ist auf Grund der durch die Augen gewonnen Informationen über seine Umwelt in der Lage mit dieser zu interagieren. So ist es ihm zum Beispiel möglich nach erkannten Objekten zu greifen und sie anzuheben. Auf diese Weise ist es dem Menschen möglich komplexe Bauteile und Werkzeuge zu fertigen und zu nutzen. Eines dieser Werkzeuge sind Roboter, die im Rahmen der Industrie 4.0 eine wichtige Rolle spielen und in unterschiedlichen Formen durch den Menschen geschaffen werden. Ähnlich wie Lebewesen, ist auch ein Roboter darauf angewiesen, Entscheidungen auf Grundlage von Informationen seiner Umwelt zu treffen.

Für die Entwicklung von Robotern orientiert sich der Mensch, ähnlich wie bei der Entwicklung des Echolots, an der Natur. Zum Erkennen von Objekten werden u. a. Kameras genutzt, die es dem Roboter ermöglichen zu sehen. Allein durch die aufgenommenen Bilder ist es dem Roboter jedoch nicht möglich, eine Entscheidung zu treffen. Der Roboter muss lernen zu erkennen, was sich auf den Bildern befindet. Einem erwachsenen Menschen gelingt dies auf Grundlage jahrelanger Erfahrungen und dem Training, welches er zu Beginn seines Lebens erhalten hat. Ständige Nennungen der im Sichtfeld befindlichen Objekte, sowie ein Feedback, bspw. durch die Eltern, ermöglichen es den Menschen Objekte zu detektieren und zu klassifizieren.
Ein ähnliches Vorgehen wird beim Training des Roboters mit Hilfe maschinellen Lernens angewandt. Durch wiederholtes Zeigen von Objekten und einem Feedback, um welches Objekt es sich handelt, lernt der Roboter auf Grundlage der durch die Kamera generierten Bilder, Objekte zu detektieren und zu klassifizieren. Anders als bei einem erwachsenen Menschen, dem es möglich ist auf Grundlage weniger Bilder neue Objekte zu lernen, benötigt ein Roboter eine Vielzahl an Bildern. Entscheidend beim Erzeugen dieser Bilder ist es, eine hohe Varianz der dargestellten Objekte und der Umwelt in den Bildern zu schaffen. Auf diese Weise lernt der Roboter das Objekt unter verschiedenen Umweltbedingungen zu detektieren und zu klassifizieren.
Das Erzeugen der benötigten Bilder ist jedoch zeitaufwändig und dadurch kostenintensiv. Aus diesem Grund haben sich Unternehmen auf das Erzeugen von Datensätzen spezialisiert, die sie verkaufen können. Der Preis für die zu verkaufenden Datensätze ist jedoch hoch [3]. Für die Entwicklung einer Objektdetektion stellt dieser Aspekt ein Hindernis dar. Vor allem in Bereichen in denen häufig neue Objekte durch den Roboter detektiert werden müssen, ist der Kauf von Datensätzen ein großer Kostenfaktor.
Auf Grundlage der beschriebenen Problematik entstehen die folgenden Fragestellungen.

- Wie ist es möglich, Datensätze kostengünstig zu erzeugen, sodass diese für die Entwicklung von Robotern ein geringeres Hindernis darstellen?
- Ist es möglich, den Zeitfakor zum Erstellen des Datensatzes zu verringen, bzw. das Erzeugen der Daten zu automatisieren?

Die gestellten Fragen werden mit der Ausarbeitung der vorliegenden Masterarbeit untersucht.

1.2 Aufgabenstellung

Im Rahmen eines unternehmensübergreifenden Projektes wird eine mobile Fertigungsstraße (Mini-Factory) entwickelt und umgesetzt. Die beteiligten Unternehmen sind die Airbus S.A.S, das Deutsche Zentrum für Luft- und Raumfahrt e. V. und die Dassault Systémes Deutschland GmbH. Realisiert wird die Mini-Factory am Zentrum für Angewandte Luftfahrtforschung (ZAL).
Ziel der Mini-Factory ist es, eine Plattform zu entwickeln, die nicht an einen Ort stationär aufgestellt ist und dadurch die Möglichkeiten bietet in verschiedenen, industriellen Umgebungen eingesetzt zu werden. Zudem soll eine kollaborierende Arbeit zwischen Roboter und Mensch mit Hilfe der Mini-Factory möglich sein.
Das Ziel dieser Masterarbeit ist es, eine Objektdetektion in die Mini-Factory zu implementieren. Als zu erkennende Objekte werden Einzelteile eines Getriebes genutzt, die in dem Arbeitsbereich der Roboter der Mini-Factory platziert werden. Für eine erste Implementierung werden lediglich zwei Einzelteile des Getriebes genutzt, um herauszufinden inwieweit die Umsetzung einer Objektdetektion möglich ist. Des Weiteren wird auf Grundlage der Ergebnisse der ersten Implementierung eine Aussage getroffen, inwiefern weitere Einzelteile des Getriebes der Objektdetektion hinzugefügt werden können.
Zudem wird zwischen zwei Möglichkeiten der Objektdetektion verglichen, um den geeigneten Ansatz für die Mini-Factory zu definieren. Ein Vergleich zwischen den Ansätzen wird auf Grundlage von Ergebnissen getroffen. Nach der ersten Implementierung auf einem leistungsstarkem Computer wird die Objektdetektion auf einem Single-Board-Computer (SBC) implementiert und an der Mini-Factory montiert. Durch die Nutzung eines SBC entsteht ein geringer Einfluss durch zusätzliche Hardware an der Mini-Factory. Für die Detektion sollen vorerst nur 2D-Bilder genutzt werden. Durch die Entwicklung der Objektdetektion wird der Grundstein gelegt, auf denen weitere Entwicklungen folgen.

1.3 Aufbau der Arbeit

Die vorliegende Masterarbeit wird in insgesamt sechs Kapitel unterteilt. Nach der Beschreibung der Motivation, wie auch der Aufgabenstellung im Kapitel 1, folgt in Kapitel 2 die Einordnung der Thematik in den Stand der Technik. Dabei werden verschiedene Ansätze für die Entwicklung einer Objektdetektion vorgestellt, sowie auf eine Methode

zur Bewertung von Machine Learning Ansätzen näher eingegangen. Des Weiteren werden in Kapitel 2 Grundlagen erklärt, die für das Verständnis der Arbeit notwendig sind. Näher erläutert werden vor allem die Machine Learning Ansätze, die einen Fokus auf das Deep Learning legen. Das Berechnen einer Messgröße zur Bewertung der Ergebnisse ist ebenfalls Bestandteil der Grundlagen.
In Kapitel 3 wird eine Anforderungsanalyse erstellt. Diese dient, mit der Einordnung der Thematik in den Stand der Technik, zur Ausarbeitung eines Konzeptes. Im Konzept werden verschiedene Arbeiten zur Erstellung einer Objektdetektion als Referenz genutzt. Zudem werden Methoden diskutiert, die für das Erzeugen eines Datensatzes vielversprechend sind. Mit dem Ende der Konzeptvorstellung werden zwei Ansätze ausgewählt, die eine vielversprechende Umsetzung einer Objektdetektion versprechen.
Im 4. Kapitel werden zwei Programme verglichen, die sich für das Erzeugen des Datensatzes anbieten. Eine Gegenüberstellung der beiden Programme hilft dabei, die Auswahl näher zu erläutern. Der in dieser Masterarbeit behandelte Use-Case beschäftigt sich mit der Detektion von Bauteilen eines Getriebes. Es wird erläutert, weshalb für einen ersten Versuch lediglich zwei der Getriebebauteile genutzt werden. Zudem wird in Kapitel 4 beschrieben, wie der Datensatz erzeugt wird und auf welche Besonderheiten zu achten sind. Die Vorstellung der Mini-Factory erfolgt ebenfalls in Kapitel 4. Eine Auswahl der geforderten Hardware ist ein weiterer Bestandteil des vierten Kapitels. Dafür genutzt wird eine Bewertungstabelle, die eine übersichtliche Darstellung bietet. Im Anschluss wird die Implementierung der Machine Learning Ansätze unter der Verwendung der genutzten Frameworks erläutert.
Das 5. Kapitel beinhaltet eine Darstellung und Diskussion der erzielten Ergebnisse. Dabei wird diskutiert, welche der beiden genutzten Machine Learning Ansätze sich für die Implementierung auf einem Single Board Computer besser eignet. Des Weiteren wird ein Vergleich zwischen synthetisch hergestelltem und realen Datensatz, welcher als Referenz genutzt wird, erzeugt.
Abgeschlossen wird die Masterarbeit mit einem Fazit, in dem die Ergebnisse kritisch bewertet werden. Die Formulierung eines Ausblicks hilft, Ansätze und Verbesserung für künftige Ausarbeitungen zu finden.

2 Stand der Wissenschaft und Technik

2.1 State of the Art

Für die Detektion und Klassifizierung von Objekten gibt es unterschiedliche Ansätze die sich in der Vergangenheit bewährt haben. Zum Zeitpunkt der Ausarbeitung der Masterarbeit werden vor allem die Themengebiete des autonomen Fahrens, wie auch die Industrie 4.0 in diversen Arbeit behandelt. Dabei werden unter der Berücksichtigung von verschiedenen Rahmenbedingung unterschiedlichste Ansätze verwendet, von denen im Folgenden einige Ansätze näher betrachtet werden.
Das Definieren von Mustern zur Erkennung von Strukturen der zu detektierenden Objekte bietet eine Möglichkeit [26] die Objekte zu erkennen. Dabei lässt sich dieser Ansatz in die klassiche Bildverarbeitung einordnen, bei der ein Mensch die Muster für die Detetkion von Objekten definiert. Das Definieren von Mustern durch den Menschen birgt das Risiko, dass nicht ausreichend oder falsche Muster für die Detektion von Objekten genutzt werden und es dadurch dem Algorithmus nicht möglich ist, das Objekt richtig zu detektieren. Aus diesem Grund werden Ansätze entwickelt, bei denen der Mensch lediglich die Rahmenbedingungen definiert. Modernere Ansätze für die Objekterkennung zählen häufig in den Bereich des Machine Learnings. Mit Ansätzen, die dem Machine Learning zuzuordnen sind, werden Methoden entwickelt, die es den Algorithmen ermöglichen, selbständig Bedingungen zum Lösen der Probelmatik zu definieren.
Die Support Vector Machine (SVM) stellt einen häufig genutzten Ansatz im Bereich des Machine Learnings dar. In den Ansätzen der Objektdetektion wird die SVM häufig unter der Verwendung des Histograms of Oriented Gradients (HOG) angewandt.[9], [27], [41] Mit Hilfe des HOGs werden signifikante Merkmale des Bildes extrahiert. Durch die Verwendung der SVM werden die extrahierten Merkmale klassifiziert.
Mizuno beschreibt in der veröffentlichten Arbeit eine Möglichkeit die SVM unter Verwendung des HOGs auf einem FPGA-Prototypenboard zu implementieren [32]. Dabei entwickelt Mizuno ein System, welches sowohl robust gegenüber verändernden Lichtverhältnissen ist, als auch eine geringe Rechenleistung benötigt. Dadurch eignet sich der

Ansatz zur Implementierung auf Systemen, die von mobilen Energiequellen versorgt werden, wie es das FPGA-Prototypenboard darstellt. Des Weiteren erreicht Mizuno mit dem beschriebenen Ansatz auf dem FPGA-Prototypenboard eine Bildfrequenz von 72 Bildern pro Sekunde (fps), bei einer Bildauflösung von 800x600 Pixeln. Mizuno wendet in dem Ansatz eine modifzierte Art des HOGs an, wodurch die Rechenzeit verringert wird.
Eine Erweiterungen des HOG-SVM Ansatzes wird in der von Wang veröffentlichten Arbeit beschrieben.[44] Wang benutzt in dem Ansatz, zusätzlich zu den HOGs zur Merkmalextraktion, Local Binary Patterns (LBP).
Für die Vergleichbarkeit der Ansätze existieren Wettbewerbe. Einer der bekanntesten Wettbewerbe ist die ImageNet Large Scale Visual Recognition Competition (ILSVRC). Der ILSVRC wurde im Jahr 2010 zum ersten Mal ausgeführt und basiert auf dem von Li erstellten Datensatz, dem ImageNet.[11] Mit dem Erfolg des AlexNet, einem Faltungsnetzwerks (engl.: Convolutional Neural Network) das durch das Team von Krizhevsky entwickelt wurde [28], beim ILSVRC im Jahr 2012 begann die Intensität der Erforschung neuer Convolutional Neural Network (CNN) Architekturen zuzunehmen. Das AlexNet gewann den ILSVRC mit einer Fehlerrate von 15,3%.[5] Damit lag das AlexNet mit 10,8 Prozentpunkten vor dem Zweitplatzierten des ILSVRC des selben Jahres (Gunji 26,1%). Das zweitplatzierte Team des ILSVRC 2012 um Gunji nutzte für die Bewältigung der Aufgabe zur Merkmalsextraktion eine Vielzahl von Ansätzen, darunter auch den bereits erwähnten LBP-Ansatz. Für die Klassifizierung nutzte das Team um Gunji einen passive-aggressive Ansatz. Ebenfalls vertreten in dem Wettbewerb ist ein Ansatz, der zur Bewältigung des Problems eine SVM wählt. Erreicht hat das Team um Simonyan mit dem SVM-Ansatz den dritten Platz (26,97%).
Der Erfolg des AlexNet führte dazu, dass sich der Fokus bei der Entwicklung neuer Ansätze zur Objektdetektion auf CNNs richtete. Durch neue Architekturen und weiterer Anpassungen der CNNs erreichten Unternehmen wie Google [40] und Microsoft [20] Genauigkeiten beim ILSVRC in den folgenden Jahren, die die Genauigkeit des AlexNets übertrafen. Der von Google finanzierte Ansatz erreichte beim ILSVRC 2014 eine Fehlerrate von 6,7%. Microsoft erreichte im folgenden Jahr, beim ILSVRC 2015, eine Fehlerrate von 3,57%.
Diverse CNN-Architekturen werden auch zum Lösen von Problemen im industriellen Umfeld angewandt. Ein Problem, das auch für die Entwicklung der Mini-Factory entscheidend ist, ist das Greifen von Objekten durch die Roboter. Ansätze für die Implementierung einer Greiffunktion basierend auf CNNs werden in verschiedenen Arbeiten behandelt.[25], [34], [46] Die Implementierung der Greiffunktion ist kein Bestandteil dieser Masterarbeit. Viel mehr wird sich im Rahmen dieser Masterarbeit auf die Ausar-

beitung des ersten Schrittes zur Implementierung einer Greiffunktion konzentriert. Die Detektion von Objekten repräsentiert den ersten Schritt. Dabei ist seit dem Erfolg des AlexNets eine Vielzahl an CNN-Architekturen entstanden, die sich für die Umsetzung einer Objektdetektion bewährt haben. Unterteilen lassen sich die CNN-Ansätze u. a. in einstufige und zweistufige Ansätze. Eine bekannte einstufige CNN-Architektur ist der You Only Look Once (YOLO) Ansatz.[35] Faster-RCNN repräsentiert eine bekannte, zweistufige CNN-Architektur. In dem Kapitel 2.2.1 wird die prinzipielle Funktionsweise von CNNs vorgestellt. Das Folgende Kapitel 2.2.2 und das Kapitel 2.2.3 stellen in einer detaillierte Darstellung den zweistufigen Faster-RCNN Ansatz und den einstufigen YOLO-Ansatz vor.

2.2 Grundlagen

2.2.1 Convolutional Neural Network

Aufgenommene Bilder werden durch Matrizen dargestellt, die Informationen über die Farbwerte beinhalten. Dabei können die Matrizen unterschiedlich viele Schichten besitzen. Farbige Bilder werden durch dreischichtige Matrizen repräsentiert, wobei jeweils eine Schicht für einen Farbkanal genutzt wird. Bilder, die durch Grauwerte dargestellt werden, benötigen eine Schicht. Dabei repräsentiert die eine Schicht Pixelwerte, die sich in einem Bereich von 0 bis 255 befinden. Den Wert 0 nehmen Pixel an, wenn sie vollständig schwarz sind. Bei dem Wert 255 sind die Pixel vollständig weiß. Mit den Werten, die jedes Pixel besitzt, ist es möglich Konturen und damit Objekte darzustellen. Auf Grundlage der Pixelwerte ist den Algorithmen der Bildverarbeitung möglich Objekte zu detektieren und zu klassifizieren.
Den Grundstein für die Klassifizierung und Detektion von Objekten durch ein neuronales Netz in Bildern legt Yann LeCun mit der Arbeit Gradient-Based Learning Applied to Document Recognition. [29] Darin beschreibt LeCun eine Möglichkeit mit Hilfe der mathematischen Faltungsoperation Eigenschaften von Schriften aus Bildern zu extrahieren und mit Hilfe eines neuronalen Netzwerks zu klassifizieren. Dieser Ansatz wird als Convolutional Neural Network (CNN) bezeichnet. Das CNN ist eine spezielle Form des neuronalen Netzwerks. Der Name des CNNs basiert auf der mathematischen Operation Faltung (engl. Convolution). Das CNN besteht aus verschiedenen Schichten, die unterschiedlich angeordnet werden und dadurch unterschiedliche Architekturen bilden. Die Architekturen der CNNs erzeugen unterschiedliche Ergebnisse in der Genauigkeit und

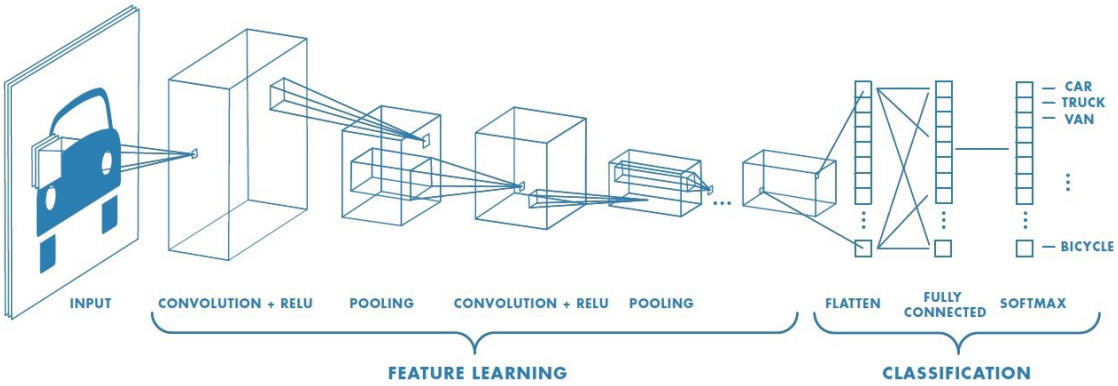


Abbildung 2.1: Aufbau eines Convolutional-Neural-Network [39]

Geschwindigkeit beim Klassifizieren und Detektieren von Objekten. Den Durchbruch erreichten CNNs mit dem von Krizhevsky entwickelten AlexNet beim ILSVRC im Jahr 2012.[5]

Mit Hilfe des in Abbildung 2.1 beispielhaft, dargestellten Aufbaus einer CNN Architektur wird die Funktionsweise der einzelnen Schichten beschrieben. Die für das CNN namensgebende Schicht, die Faltungsschicht (engl. Convolutional-Layer), ist die in Abbildung 2.1 erste dargestellte Schicht, nach dem Eingabebild, welches ein Auto darstellt. Sie besteht aus einer Faltungsmatrix, auch Kernel genannt, die dazu genutzt wird verschiedene Merkmale aus dem Eingabebild zu filtern. Zu den Merkmalen zählen u.a. Kanten und Rundungen, aber auch andere Merkmale, die auf Grund ihrer Vielzahl im Rahmen dieser Arbeit nicht näher aufgelistet werden. Die durch die Convolution-Schicht gefilterten Merkmale werden in einer Feature-Map gespeichert. Die Feature-Map enthält die für die Detektion und Klassifizierung notwendigen Informationen, in reduzierter Darstellunsgweise. Ein Beispiel für eine Feature-Map ist in Abbildung 2.2 dargestellt. Extrahiert werden Eigenschaften einer Katze. Dabei repräsentiert jeweils eine Zeile eine Convolutional-Schicht. Die Ursache für die jeweils acht dargestellten Abbildung der extrahierten Eigenschaften ist die Anwendungen von verschiedenen Filtern, die durch die Convolutional-Schicht auf die Eingabematrix angewandt werden. Mit jeder weiteren Anwendung einer Convolutional-Schicht sehen die extrahierten Merkmale für das menschliche Auge abstrakter aus. Dabei ist zu Berücksichtigen, dass die erzeugten Feature-Maps an die folgende Schicht weitergegeben wird. Im Folgenden wird die Funktionsweise der Convolutional-Schicht näher erläutert.

Der Kernel wird in der Abbildung 2.1 in dem Eingabebild (Input) als weißes Rechteck dargestellt. Eine anschauliche Darstellung, wie mit Hilfe des Kernels Merkmale aus dem

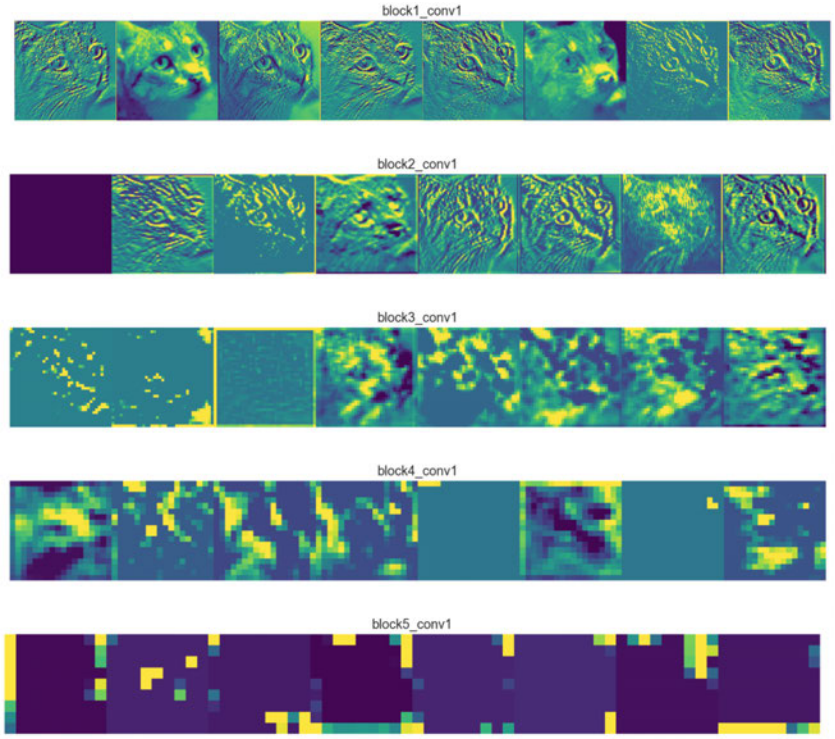


Abbildung 2.2: Feature Map [12]

Eingabebild extrahiert werden, wird in Abbildung 2.3 vorgestellt. Die mit Input bezeichnete Matrix repräsentiert ein Eingabebild, welches lediglich aus Grauwerten besteht. Zu erkennen ist dies auf Grund der nur einen vorhandenen Schicht des Eingabebildes. Der Kernel wird über die Eingabematrix gelegt. Die sich jeweils übereinander liegenden Werte der Matrizen werden miteinander multipliziert. Das Ergebnis der Multiplikation wird in der Matrix Interim Result dargestellt.
Da keine negativen Werte für Pixel dargestellt werden können, der Wertbereich liegt zwischen 0 und 255, wird eine nicht lineare Aktivierungsfunktion verwendet. In dem vorliegenden Beispiel wird die Aktivierungsfunktion Rectified Linear Unit (ReLu) genutzt. Anschließend werden die Teilergebnisse, die in der Interim Result Matrix dargestellt sind, aufsummiert. Das Ergebnis wird in die Result Matrix gespeichert. Der Kernel wird um eine definierte Schrittweite bewegt, wobei in dem vorliegenden Beispiel ein Schritt, einer Spalte oder einer Zeile entspricht. Auf diese Weise wird die komplette Eingabematrix durchlaufen.
Durch das Anwenden der Faltung verliert die Ergebnismatrix Zellen, im Vergleich zur Eingabeschicht. Dies ist hinderlich bei der Verwendung von mehreren Schichten zum Ex-

trahieren von Merkmalen. Architekturen wie das ResNet erhoffen sich durch die Verwendung von mehreren Schichten, dass die Genauigkeit des CNN Ansatzes erhöht wird. Das ResNet-152 verwendet zum Detektieren von Objekten insgesamt 152 Faltungsschichten.[20]
Durch das Hinzufügen weiterer Zellen am Rand der Eingabeschicht wird verhindert, dass die Ergebnismatrix schrumpft. Das Hinzufügen von weiteren Zellen wird als Padding bezeichnet. Dabei besitzen die hinzugefügten Zellen oft den Wert 0.
Dem Convolutional Layer folgt das Pooling Layer. Unterschieden wird beim Pooling Layer in Max-, Average- und Sum-Pooling. Die Funktionsweise des Pooling Layers kann wie folgt beschrieben werden. Die Ergebnismatrix wird durch das Pooling Layer in Untermatrizen unterteilt, wobei die Untermatrizen jeweils die selben Dimensionen besitzen. Vergleichbar mit der Schrittweite des Kernels ist auch die Schrittweite der Untermatrizen definiert. Bei einer 2x2 Untermatrize mit einer Schrittweite von eins überlappen sich die Untermatrizen. Betrachtet man beispielsweise ein Max-Pooling-Layer wird der höchste Wert in der definierten Untermatrix an die nächste Schicht weitergegeben. Auf diese Weise werden die zur Verfügung stehenden Informationen durch das Pooling Layer verallgemeinert. Dies besitzt einen positiven Effekt auf die Berechnungszeit von CNNs, wodurch tiefere Architekturen realisierbar sind.
Die Anordnung und Anzahl der Convolutional Layers und Pooling Layers führt zu einer Vielzahl unterschiedlicher Architekturen. Die Eingabematrizen, die die extrahierten Merkmale beinhalten, werden in Vektoren umgewandelt. Dieser Vorgang wird als Flatten bezeichnet. Durch das Flatten ist es der folgenden Schicht, dem Fully-Connected Layer, möglich die Eingabeinformationen zu verarbeiten. Der Fully Connected Layer besteht aus Neuronen, die auf Grundlage der extrahierten Merkmale eine gewichtete Wahrscheinlichkeit über die Position und die Klasse des dargestellte Objekts trifft. Eine Vorhersage können die Neuronen mit Hilfe einer Aktivierungsfunktion und den einzustellenden Parametern treffen.
Die letzte Schicht des in Abbildung 2.1 dargestellten CNNs stellt das Softmax Layer dar. Sie besteht aus Neuronen, die zu detektierenden Klassen repräsentieren. Durch die Verbindungen des Softmax Layers zu dem Fully-Connected Layer wird eine Zuordnung der jeweiligen Klassen bestimmt. Eine Besonderheit beim Softmax Layer ist die exklusive Zuordnung der Ergebnisse zu einer Klasse.
Bei einem CNN handelt es sich um ein Feedforward-Netzwerk. Die einzelnen Schichten werden ausschließlich mit den darauffolgenden Schichten verbunden. Die an das CNN übergebenen Informationen durchlaufen das Netzwerk in eine Richtung.
Ein weiteres Merkmal des CNNs ist es, dass es sich um ein Überwachtes (engl. Supervised) Netzwerk handelt. Zum Anpassen und Validieren des Netzwerks werden Datensätze

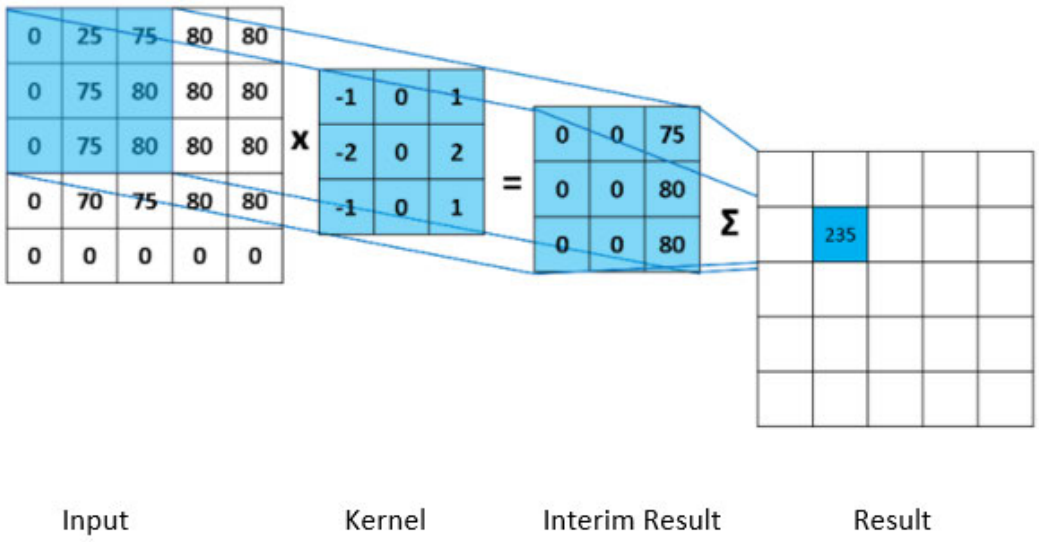


Abbildung 2.3: Erläuterung des mathematischen Faltungsoperators [2]

präpariert. Informationen über die dargestellten Objekte werden in zusätzlichen Dateien gespeichert, auf die das Netzwerk erst mit dem Durchlauf des Bildes zugreifen kann. Das durch das Netzwerk bestimmte Ergebnis wird mit dem erwarteten Ergebnis, welches in den zusätzlichen Dateien gespeichert ist, verglichen. Für den Vergleich wird die sogenannte Loss-Funktion aufgestellt. Eine detaillierte Beschreibung der verwendeten Loss-Funktion erfolgt in dem Abschnitt 2.2.2 und dem Abschnitt 2.2.3. Dabei ist es das Ziel des Ansatzes das Ergebnis der Loss-Funktion zu minimieren. Idealerweise ist das Ergebnis der Loss-Funktion null. Besitzt die Loss-Funktion einen von null abweichenden Wert, werden mit Hilfe des stochastischen Gradientenverfahrens sowohl die Filter (Kernel) für die Merkmalsextratktion, wie auch die Parameter des Fully-Connected Layers angepasst. Die Anpassung der Parameter erfolgt bei der sogenannten Backpropagation. Eine zu geringe Varianz in dem Trainingsdatensatz ist ein Beispiel, weswegen eine sogenannte Überanpassung (engl. Overfitting) des Netzwerkes vorliegt. Dies bedeutet, dass das Netzwerk die Parameter auf die Eigenschaften des Datensatzes anpasst. Dem Netzwerk ist es nicht möglich Objekte in Bildern, die unabhängig vom Traingsdatensatz sind, zu detektieren. Das Netzwerk kann lediglich Objekte in dem gewohnten Trainingsdatensatz erkennen. Mit einer zu schlechten Einstellung der Parameter des Netzwerkes, durch bspw. einem zu kleinen Trainingsdatensatz, besteht die Gefahr der Unteranpassung (engl. Underfitting). Das Netzwerk ist nicht in der Lage sowohl im Trainingsdatensatz, wie auch

bei unbekannten Bildern, Objekte zu detektieren. Das Ziel beim Training eines Netzwerkes ist es, eine ausgeglichene (engl. balanced) Einstellung der Parameter zu erreichen. Wodurch das Netzwerk in der Lage ist Objekte in unbekannten Bildern richtig zu detektieren.
Der im folgendem Abschnitt 2.2.2 vorgestellte Faster-RCNN Ansatz basiert auf der Funktionsweise des CNNs. Jedoch wird durch die Anpassungen der Vorgehensweise ein Ansatz entwickelt, der eine höhere Genauigkeit im Vergleich zu dem ursprünglichen Vorgehen des CNNs besitzt. Dies ist der Grund, weswegen der Faster-RCNN in einer Vielzahl von Arbeiten genutzt wird.

2.2.2 Faster-RCNN

Das in Abschnitt 2.2.1 vorgestellte Netzwerk dient als vereinfachte Darstellung der Vorgehensweise von CNNs zur Objektdetektion. Mit dem Erfolg des AlexNets entwickelten Forscher verbesserte Architekturen und Ansätze, basiernd auf CNNs. Eine der populärsten Ansätze für die Objektdetektion der vergangen Jahre stellt das Faster-RCNN dar, welches von Shaoqing vorgestellt wurde.[38] Das Faster-RCNN ist nach dem Fast-RCNN die zweite Weiterentwicklung des R-CNN , welches von Girshick im Jahre 2014 präsentiert wurde.[14], [15]
Der Faster-RCNN nutzt für die Objekdetektion zwei Stufen. Dabei wird die erste Stufe durch das bereits vorgestellte CNN repräsentiert. Das CNN wird für die Merkmalsextraktion genutzt und erzeugt damit eine Feature Map. Die zweite Stufe des Faster-RCNN dient für eine bessere Eingrenzung der Bereiche des Bildes, in denen sich Objekte befinden können. Genutzt wird für die Eingrenzung ein Regional Proposal Network (RPN), ein weiteres Faltungsnetzwerk. Die durch das CNN erzeugte Feature Map dient als Eingabe für das RPN. Die Vorgehensweise des Faster-RCNN ist in Abbildung 2.4 dargestellt.
Für die Eingrenzung von Bereichen in denen sich Objekte befinden können, nutzt das RPN eine Methode, die im englischen als Sliding Window bezeichnet wird. Die Vorgehensweise des Sliding Windows ähnelt der Merkmalsextraktion durch das CNN, welche bereits beschrieben wurde. Ähnlich wie bei der Erstellung der Feature Maps, wird auch beim RPN eine schrittweise Untersuchung des Bildes getätigt. Jedes Pixel der Feature Map wird untersucht. Um eine Vorhersage über einen potentiellen Bereich zu tätigen, nutzt das RPN Anchor-Boxes. Die Anchor-Boxes repräsentieren Boxen mit unterschiedlichen Verhältnissen, welche für die zu detektierenden Objekte ausgelegt sind. So wird bspw. bei der Detektion von stehenden Menschen eine Anchor-Box verwendet, mit einer

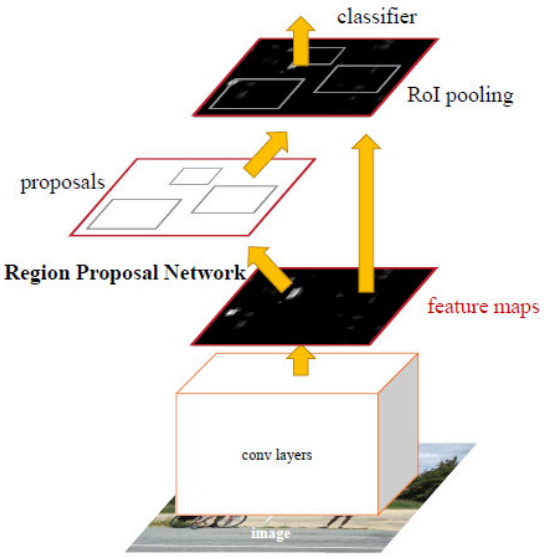


Abbildung 2.4: Aufbau Faster-RCNN [38]

schmalen horizontalen Kante, im Vergleich zu einer langen vertikalen Kante. Bei Autos wird eine breite horizontale Kante, mit einer kurzen vertikalen Kante verwendet. Der Mittelpunkt der Anchor-Boxes befindet sich auf dem definierten Schritt. Insgesamt nutzt das RPN drei Anchor-Boxes, die in drei unterschiedlichen Skalierungen vorliegen. Somit stehen dem RPN insgesamt neun Anchor-Boxes für die Vorhersage zur Verfügung. Eine Darstellung potentieller Anchor-Boxes ist in Abbildung 2.5 erkennbar. Die Vorgehensweise des RPN ähnelt dem der CNNs. Durch die Nutzung der Faltung selektiert das RPN Informationen, die zur Bestimmung eines potentiellen Bereiches führen. Dabei liegt der Vorteil bei Nutzung von RPNs, dass Gewichte zwischen RPNs und CNNs geteilt werden. Eine Reduktion der benötigten Zeit im Vergleich zu Fast-RCNN, welches kein RPN für die Eingrenzung nutzt, ist die Folge.
Das RPN trifft keine Vorhersage über die Klasse, lediglich ob sich in dem Bereich ein Objekt befinden könnte. Das RPN unterscheidet dabei Bereiche die dem Vordergrund zugehörig sind und Bereiche die dem Hintergrund zugeordnet werden. Als Bereiche, die dem Vordergrund zuzuordnen sind werden zu detektierende Objekte definiert. Das RPN unterscheidet zwischen Vorder- und Hintergrund mittels eines zwei Klassen Softmax-Layers, welches der Anchor-Box eine Objektwert zuordnet. Der Objektwert beschreibt die Wahrscheinlichkeit eines vorhandenen Objektes in der Anchor-Box. Das Ergebnis der Vorhersage wird in einer Klassifikations-Schicht (cls-Schicht) gespeichert. Des Weiteren

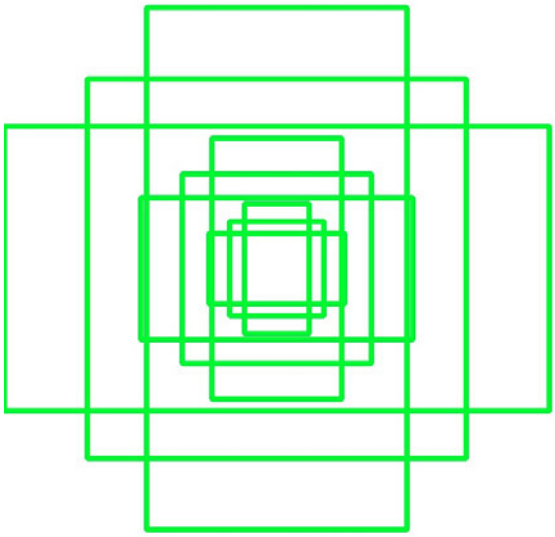


Abbildung 2.5: Anchor-Boxes Faster-RCNN [16]

versucht das RPN durch eine Vorhersage die Anchor-Box an den Bereich anzupassen, der durch das RPN als vielversprechend bewertet wird. Die Position und die Maße der angepassten Anchor-Box werden in einer weiteren Schicht (reg-Schicht) gespeichert.
Für die Klassifizierung der zu untersuchenden Bereiche wird der im Fast-RCNN beschriebene Ansatz verwendet.[14] Mit Hilfe eines Pooling-Layers werden die durch das RPN ermittelten Vorhersagen (cls-Schicht und reg-Schicht) von der Feature-Map des CNN getrennt. Die getrennten Bereiche werden mittels zweier Fully-Connected Layer untersucht. Eine Zuordnung der durch die Fully-Connected Layer erzielten Ergebnisse wird durch eine abschließende Softmax-Layer getroffen.

$$L(p_i, t_i) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*) \tag{2.1}$$

Die Bewertung der Detektion erfolgt mittels Loss-Funktion, die eine Differenz zwischen geschätzter und tatsächlicher Klasse und Position ermittelt. Die Loss-Funktion, die für die Bestimmung der Genauigkeit im Faster-RCNN Ansatz genutzt wird, ist in Gleichung 2.1 dargestellt. Für die Objekdetektion nutzt das RPN mehrere Anchor-Boxes, die an einer Vielzahl von Positionen, Untersuchungen durchführen können. Die Anzahl der genutzten Positionen der Anchor-Boxes liegt bei etwa 2400 Stück. Zur Berechnung unterteilt der Faster-RCNN Ansatz die 2400 Positionen in weitere Untergruppen, die als Mini-Batches bezeichnet werden. Verwendet werden 256 Anchor-Boxes pro Mini-Batch. Auf diese Weise ist es dem Algorithmus möglich die Berechnung durchzuführen.

Der Koeffizient i in der Gleichung 2.1 gibt die betrachtete Anchor-Box innerhalb des Mini-Batches an. Mit dem Koeffizienten p_i wird die Wahrscheinlichkeit angegeben, ob es sich um das in der Anchor-Box dargestellte Objekt, tatsächlich um eines handelt. Dabei nimmt der Koeffizient p_i^* einen Wert von eins an, wenn die Anchor-Box ein Objekt beinhaltet und einen Wert von null, wenn in der Anchor-Box kein Objekt dargestellt wird. t_i ist ein Vektor, der die vier Koordinaten der Bounding Box beinhaltet. Die Bounding Box ist die kleinste Umschließungsbox für ein Objekt. Sie ist in den Maßen abweichend von den Anchor-Boxes.
Mit dem Koeffizienten t_i^* werden die Koordinaten der Ground-Truth Bounding Box dargestellt und repräsentieren somit die tatsächlichen Koordinaten des Objektes. Mittels L_{cls} und unter Verwendung der binären Kreuzentropie wird bestimmt, ob es sich um ein Objekt handelt oder ob es sich um kein Objekt handelt. Zur Bestimmung des Positionsverlusts wird der Koeffizient $L_{reg}(t_i, t_i^*)$ genutzt. Die Berechnung erfolgt unter Verwendung der smooth L1 Funktion. Zur Normalisierung der Therme werden die Werte N_{cls} und N_{reg} genutzt. Mit λ wird die Anpassung gewichtet. Shaoqing wählt für den Gewichtungsfaktor λ einen Wert von 10. Dadurch besitzen beide Therme etwa den selben Einfluss auf die Loss-Funktion. Vergleichbar zu der in Abschnitt 2.2.1 beschriebenen Minimierung des Fehlers wird auch bei dem Faster-RCNN Ansatz das stochastische Gradientenverfahren genutzt. Eine anschließende Einstellung der Parameter durch die Backpropagation erfolgt.
Der Faster-RCNN ist, wie bereits erwähnt, durch die Nutzung des RPN ein zweistufiger CNN-Ansatz. Die Genauigkeit ist im Vergleich zu anderen CNN-Ansätzen hoch. Jedoch besitzt der Faster-RCNN eine hohe Laufzeit zur Verarbeitung des Bildes. Zum Vergleich wird im Abschnitt 2.2.3 ein einstufiger Ansatz vorgestellt, dessen Fokus auf einer niedrigen Laufzeit beruht.

2.2.3 You Only Look Once - YOLO

Einstufige CNN Architekturen sind darauf ausgelegt eine möglichst geringe Laufzeit bei der Detektion von Objekten zu gewährleisten. Dadurch soll eine echtzeitfähiges System erzeugt werden, die bei der Nutzung in zeitkritischen Bereichen benötigt wird. Bevor eine Erläuterung der einstufigen CNN Archtiktur erfolgt, wird der Begriff Echtzeitfähigkeit näher betrachtet.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Abbildung 2.6: Vergleich zwischen verschiedenen Ansätzen [35]

Eine allgemeingültige Aussage über die Echtzeitfähigkeit eines Ansatzes ist nicht umsetzbar. Nach der DIN-Norm 44300 (Nummer 9.2.11) ist die Echtzeitfähigkeit wie folgt definiert: „Eine Betriebsart eines Rechensystems bei der Programme zur Verarbeitung anfallender Daten ständig ablaufbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfalls nach einer zeitlichen zufälligen Verteilung oder zu vorher bestimmten Zeitpunkten anfallen." [4]
Die Echtzeitfähigkeit eines Systems ist somit auch dann gegeben, wenn alle benötigten Daten zum benötigten Zeitpunkt vorhanden sind. Dadurch kann bspw. auch der Faster-RCNN Ansatz, welcher verhältnismäßig langsam ist, als echtzeitfähig eingestuft werden, insofern der Ansatz in einem dafür passenden System integriert ist. Mit Themengebieten, wie dem autonomen Fahren, sind jedoch Bereiche in den Fokus der Entwicklung gerückt, in denen Ansätze wie der Faster-RCNN nicht mehr als echtzeitfähig eingestuft werden. Die Berechnung die für die Objektdeteketion notwendig ist, liegt nicht zeitgerecht vor und entspricht nicht mehr der Norm zur Echtzeitfähigkeit. Dennoch wird in der Beschreibung der einstufigen Architektur weiterhin die Terminologie von echtzeitfähigen und nicht echtzeitfähigen Systemen beibehalten, ohne einen Bezug auf ein konkrektes Beispiel zu nehmen.
In Abbildung 2.6 werden die Ergebnisse zwischen verschiedenen CNN-Architekturen gezeigt, die mit Hilfe des MS COCO Datensatzes trainiert und getestet werden. Unterschieden wird in echtzeitfähige Systeme und in nicht echtzeitfähige Systeme. Der mean Average Precision Wert (mAP) wird zur Bewertung der Ansätze genutzt. Eine nähere

Erläuterung des mAP-Wertes erfolgt in Abschnitt 2.2.4. Ansätze wie die Faster-RCNN Architektur erreichen bei einer Laufzeit von 0,5 FPS, einen mAP-Wert von 70,0.
Der einstufige CNN-Ansatz mit dem Namen You Only Look Once (YOLO) wurde durch Redmon im Jahre 2016 vorgestellt [35]. YOLO erreicht einen mAP-Wert von 63,4 bei einer Laufzeit von 45 FPS. Durch die Gegenüberstellung ist erkennbar, dass der YOLO-Ansatz wesentlich schneller, aber auch ungenauer, als der Faster-RCNN Ansatz ist. Des Weiteren sind in der Abbildung 2.6 unterschiedliche Versionen der Ansätze dargestellt. YOLO-VGG-16 ist ein Ansatz, der im Vergleich zu YOLO, einen anderen Backbone nutzt. Als Backbone ist der Teil des Netzwerks definiert, der zur Merkmalextraktion genutzt wird. Durch die Nutzung bereits vorhandener Datensätze werden die Parameter des Backbones angepasst.
Im Gegensatz zu Faster-RCNN nutzt YOLO für die Objektdetektion nur ein CNN. Das in Abschnitt 2.2.2 beschriebene RPN wird nicht für die Vorhersage von vielversprechenden Bereichen genutzt. Stattdessen nutzt der YOLO-Ansatz für die Eingrenzung von vielversprechenden Bereichen eine Methode, bei der das zu untersuchende Bild in Zellen aufgeteilt wird. Die Zellen sind in einer gitterartigen Form angeordnet. Die Anzahl der Zellen, in die das Bild unterteilt wird, ist abhängig von der verwendeten YOLO Version. Redmon nutzte in dem ersten Ansatz von YOLO eine 7x7 Aufteilung des Bildes. In den folgenden Versionen von YOLO wird eine Aufteilung von 13x13 genutzt.[8], [36], [37] Unter anderem durch die feinere Aufteilung des Bildes ist eine genauere Vorhersage möglich.
Des Weiteren fügt Redmon mit der zweiten Generationen von YOLO, Anchor Boxes hinzu und orientiert sich damit an dem Faster-RCNN Ansatz. In der dritten Version von YOLO exisitieren drei Arten von Anchor Boxes, mit unterschiedlichen Längen - Breiten Verhältnissen. Vergleichbar mit den Anchor Boxes im Faster-RCNN Ansatz, existieren die drei Anchor Boxes in drei unterschiedlichen Skalierungen. Alle Anchor-Boxes sind im Zentrum jeder Zelle positioniert. Nachfolgend wird eine nähere Erläuterung des YOLO-Ansatzes an Hand der dritten Version von YOLO [37] beschrieben.
Das Eingangsbild wird im ersten YOLO-Ansatz auf eine Auflösung von 448x448 skaliert. In dem zweiten und dritten YOLO-Ansatz wird das Eingangsbild auf eine Auflösung von 416x416 skaliert.
Die Faltungsschichten des Backbones extrahieren besondere Eigenschaften der zu detektierenden Objekte und repräsentieren die Eigenschaften in einer 13x13 Feature Map. Die Zelle, in die der Schwerpunkt des zu detektierenden Objektes fällt, ist für die Vorhersage der Bounding Box zuständig und damit für die Vorhersage der genauen Position und Größe des Objektes. Die Vorhersagen, sowohl über die Existenz einer Klasse in der

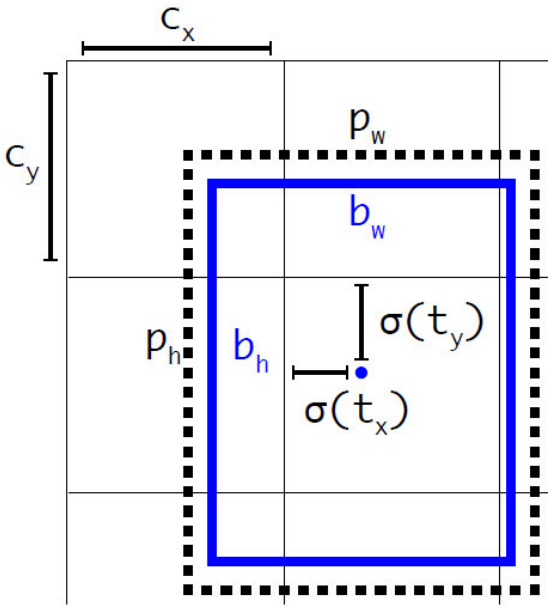


Abbildung 2.7: Anpassung der Position von Anchor Boxes [36]

Zelle, wie auch über die Position, erfolgen durch die Verwendung von Fully Connected Layers. Im Vergleich zu Faster-RCNN nutzt YOLO nicht das Sliding Window Verfahren. Redmon entschied sich die Position der ursprünglich im Zentrum jeder Zelle erzeugten Anchor Boxes, anzupassen. Mit Hilfe des Fully Connected Layers wird somit eine Vorhersage darüber getroffen, inwieweit die Position der Bounding Box von der zentralen Position der Anchor Box abweicht. Des Weiteren wird durch das Fully Connected Layer eine Vorhersage darüber getroffen, wie sich die Maße der Bounding Box von der Anchor Box unterscheiden. Die Anchor Boxes dienen lediglich als Annäherung, um zu große Abweichung bei der Vorhersage der Bounding Box zu vermeiden. In der Abbildung 2.7 wird eine beispielhafte Darstellung, einer vorhergesagten Bounding Box präsentiert. Die Formeln zur Berechnung der veränderten Position, sowie der veränderten Abmaße der Bounding Box sind in den Gleichungen 2.2, 2.3, 2.4 und 2.5 angegeben.

$$b_{\mathrm{x}} = \sigma(t_{\mathrm{x}}) + c_{\mathrm{x}} \tag{2.2}$$

$$b_{\mathrm{y}} = \sigma(t_{\mathrm{y}}) + c_{\mathrm{y}} \tag{2.3}$$

$$b_{\mathrm{w}} = p_{\mathrm{w}} * e^{t_{\mathrm{w}}} \tag{2.4}$$

$$b_{\mathrm{h}} = p_{\mathrm{h}} * e^{t_{\mathrm{h}}} \tag{2.5}$$

Der Ursprung eines Bildes befindet sich in der oberen, linken Ecke des Gesamtbildes. Eine Zelle besitzt die Breite C_x und die Höhe C_y. Sowohl die Breite, als auch die Höhe einer Bounding Box besitzen einen Wert zwischen null und eins. Die Position der Bounding Box wird durch die Variablen b_x und b_y repräsentiert. Die in Abbildung 2.7 dargestellten Gleichungen zur Berechnung von b_x und b_y werden mittels einer Addition berechnet. Mit der Verwendung der logistischen Regression wird der vorhergesagte Wert, auf einen Wert zwischen null und eins begrenzt. Dadurch befindet sich der Mittelpunkt der Bounding Box in der für die Vorhersage zuständigen Zelle. Dargestellt ist die Anwendung der logistischen Regression durch das $\sigma(t_x)$, bzw. $\sigma(t_y)$. Die Gleichung 2.6 gibt die Formel für die Berechnung der logistischen Regression an.

$$y = \frac{1}{1 + e^{-x}} \tag{2.6}$$

Der Wert C_x, bzw. C_y ist abhängig von der Entfernung der Zelle zum Ursprung des Bildes. Zur Bestimmung der Höhe und Breite der Bounding wird die Anchor Box genutzt. Dargstellt wird die Anchor Box in der Abbildung 2.7 durch das gestrichelte Rechteck. Die Breite der Anchor Box ist durch p_w, die Höhe durch p_h dargestellt. Eine Anpassung der Bounding Box erfolgt unter Berücksichtigung der Höhe und Breite der Anchor Box, sowie die Verwendung einer Exponentialfunktion. Mit der Verwendung einer Exponentialfunktion lässt sich die Anpassung in drei Fälle unterscheiden. In der folgenden Auflistung sind t_x, wie auch t_y vereinfacht durch ein t dargestellt.

- $t = 0$, Bounding Box Abmaße entsprechen Anchor Box Abmaße
- $t < 0$, Verringerung der Bounding Box Abmaße
- $t > 0$, Vergrößerung der Bounding Box Abmaße

Die Verwendung einer Exponentialfunktion führt bei Werten von $t < 0$ zu einer geringen Anpassung der Abmaße der Bounding Box. Bei Werten von $t > 0$ ist eine deutliche Vergrößerung der Abmaße der Bounding zu erwarten.
Jede der Zellen erzeugt neun Bounding Boxes für die Vorhersage des Objekts. Für die Detektion eines Objektes ist jedoch nur eine Bounding Box notwendig. Aus diesem Grund verwendet der YOLO-Ansatz Non Maximum Suppression (NMS). NMS vergleicht die Vorhersagen der Bounding Boxes und wählt die Bounding Box mit der höchsten Wahrscheinlichkeit einer Übereinstimmung mit dem zu detektierenden Objekt. Die verbliebenen acht Bounding Boxes werden gelöscht.

Die Klassifizierung des zu detektierenden Objektes erfolgt nach Multi-Label Prinzip. Dabei kann eine Bounding Box ein Objekt darstellen, welches zu mehreren Klassen gehört. Bspw. zählte eine dargestellte Frau sowohl zu der Klasse Frau, wie auch zur der Klasse Mensch. Erreicht wird dieses Vorgehen unter der Verwendung der binären Kreuzentropie. Redmon erhofft sich dadurch einen Einsatz von YOLO bei komplexeren Datensätzen, wie es der Open Image Datensatz darstellt.

$$
\begin{aligned}
Loss = & \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (c_i - \hat{c}_i)^2 \\
& + \lambda_{\text{noobj}} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} (c_i - \hat{c}_i)^2 \\
& + \sum_{i=0}^{s^2} 1_{i}^{obj} \sum_{c \, \epsilon \, classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{2.7}
$$

Das Training des YOLO-Ansatzes ist vergleichbar mit dem Training des Faster-RCNN Ansatzes. YOLO nutzt ebenfalls eine Loss-Funktion zum Vergleich des Soll- und des Ist-Zustandes. Die Gleichung 2.7 stellt die Loss-Funktion des ersten YOLO-Ansatzes vor. Für den dritten YOLO-Ansatz existieren leichte Unterschiede, jedoch wird die Loss-Funktion des dritten YOLO-Ansatzes nicht durch Redmon angegeben. Die Unterschiede werden in dem technischen Bericht lediglich erwähnt.[37] Die Erläuterung der Loss-Funktion erfolgt aus diesem Grund an Hand der Loss-Funktion aus dem ersten YOLO-Ansatz, mit einem Verweis auf die durch Redmon vorgenommene Anpassungen. Unterteilt wird die Loss-Funktion in drei Abschnitte, wobei unterschieden wird in eine Lokalisierungs Abweichung, eine Klassifikations Abweichung und einer Wahrscheinlichkeits Abweichung.
Die Lokalisierungs Abweichung wird durch die ersten beiden Therme der Loss-Funktion repräsentiert. λ_{coord} stellt dabei einen Gewichtungsfaktor dar, der durch Redmon mit einem Wert von 5 definiert wird. S ist die Anzahl der Zellen, in die das Bild unterteilt wird. Im ersten YOLO-Ansatz ist S=7, im dritten YOLO-Ansatz ist S=13. Die zweite Summe repräsentiert dabei die vorherzusagenden Bounding Boxes. Im ersten YOLO-Ansatz werden fünf Bounding Boxes vorhergesagt. Der dritten YOLO-Ansatz verwendet

neun Bounding Boxes. Der Parameter i ist die Bezeichnung für die jeweilige Zelle. Durch den Parameter j werden die Bounding Boxes genannt. 1_{ij}^{obj} nimmt den Wert eins an, sollte die Bounding Box j, in der Zelle i für die Detektion des Objektes verantwortlich sein. Andernfalls nimmt der Faktor den Wert null an. Der Vergleich der Position erfolgt durch die Koeffizienten $\hat{x}_i$ und $\hat{y}_i$, die den tatsächlichen Mittelpunkt der Ground-Truth-Box angeben. x_i und y_i sind die durch den YOLO-Ansatz vorhergesagten Koordianten der Bounding Box. Der Koeffizient w_i gibt die Breite der vorhergesagten Bounding-Box an. $\hat{w}_i$ ist die tatsächliche Breite der Ground-Truth Bounding Box. Mit h_i wird die vorhergesagte Bounding Box angegeben und mit der tatsächlichen Höhe $\hat{h}_i$ verglichen. Der Algorithmus wird abhängig von dem Ergebnis der Loss-Funktion bestraft, bzw. belohnt. Dabei bedeutet dies eine Anpassung der Parameter, die durch die Summe der Quadrate berechnet wird. Bei der Anpassung der Breite und Höhe wird zusätzlich eine Radizierung durchgeführt. Dadurch werden Fehler bei großen, wie auch bei kleinen Bounding Boxes gleichermaßen bestraft.
Mit der Klassifikations Abweichung wird die Wahrscheinlichkeit durch den Algorithmus ermittelt, dass ein Objekt vorhanden ist. Dabei ist c_i der durch den Algorithmus bestimmte Wert. $\hat{c}_i$ ist der Jaccard-Koeffizient, auf dessen Berechnung im Abschnitt 2.2.4 näher eingegangen wird. Der Koeffizient λ_{noobj} ist ein weiterer Gewichtungsfaktor. Redmon definiert den Wert im ersten YOLO-Ansatz mit 0,5 und begründet die Differenz zwischen den Λ Werten mit der zunehmenden Stabilität auf Grund der unterschiedlichen Werte. Im technischen Bericht des dritten YOLO-Ansatzes wird keine abweichende Andeutung zu den Gewichtungsfaktoren getroffen. 1_{ij}^{noobj} nimmt einen Wert von eins an, befindet sich kein Objekt in der betrachteten Zelle. Befindet sich ein Objekt in der Zelle, nimmt der Faktor den Wert null an. Dadurch verhält sich 1_{ij}^{noobj} genau gegenteilig zu 1_{ij}^{obj}.
Der letzte Teil der Loss-Funktion wird repräsentiert durch die Klassifikations-Abweichung. Verglichen wird die vorhergesagte Klasse $\hat{p}$, mit der tatsächlichen Klasse $\hat{p}_i$.
Die dritte Version von YOLO nutzt für die bessere Detektion von Objekten Feature Maps aus vorherigen Schichten. Die Feature Maps werden zusammengeführt. Auf Grund dieser Maßnahme, die sich ebenfalls am Faster-RCNN Ansatz orientiert, ist es dem dritten YOLO-Ansatz, im Vergleich zum ersten YOLO-Ansatz, möglich auch kleinere Objekte im Bild zu detektieren.

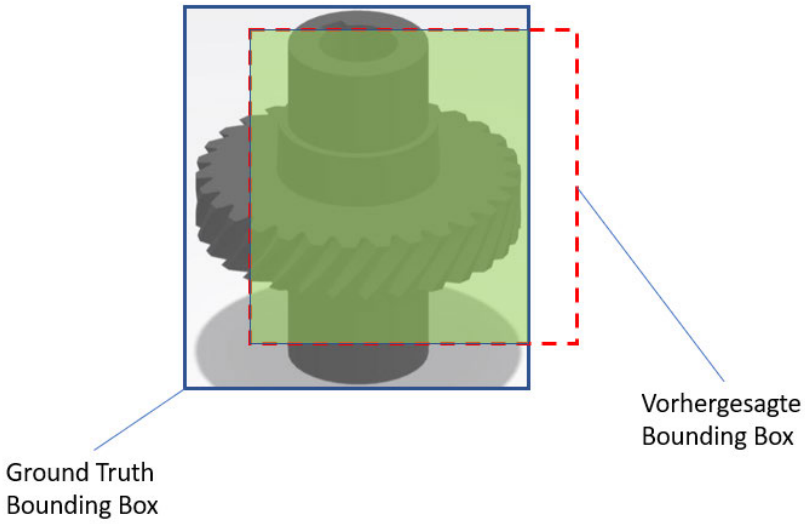


Abbildung 2.8: Jaccard-Koeffizient

2.2.4 Messgrößen

Die Bewertung von Machine Learning Ansätzen beruht auf der Verwendung von Messgrößen. Eine weit verbreitete Messgröße ist die Genauigkeit (engl. accuracy), die u. a. in den Arbeiten verwendet wird.[30], [42], [45] Dabei gibt die Genauigkeit den Wert an, mit dem der Machine Learning Ansatz Objekte richtig klassifiziert und detektiert hat. Die Berechnung der Genauigkeit beruht auf dem Jaccard-Koeffizienten. In englischsprachigen Arbeiten wird der Jaccard-Koeffizient als Intersection over Union (IoU) bezeichnet. Für die Berechnung des Jaccard-Koeffizientens im Bereich der Objektdetektion werden präparierte Daten benötigt, die in Form eines Testdatensatzes vorliegen. Die zu detektierenden Objekte in dem Testdatensatz werden mit einer Ground Truth Bounding Box gekennzeichnet. Der trainierte Machine Learning Ansatz bestimmt eine Bounding Box. In Abbildung 2.8 wird eine beispielhafte Darstellung der Boxen an Hand eines Getriebebauteils dargestellt. Dabei stellt die durchgezogene Linie die Ground Truth Bounding Box dar. Die gestrichelte Linie repräsentiert die durch den Machine Learning Ansatz vorhergesagte Bounding Box. Gemeinsam bilden die Fläche der Ground Truth Bounding Box und die Fläche der vorhergesagten Bounding Box, die Gesamtfläche. Die Schnittfläche der Bounding Boxes wird als transparente Fläche abgebildet.

$$Jaccard = \frac{Schnittflaeche}{Gesamtflaeche} \tag{2.8}$$

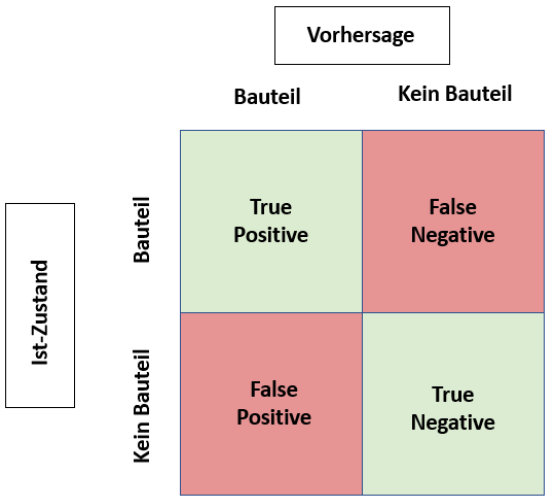


Abbildung 2.9: Konfussionsmatrix - 4 Fälle Jaccard-Koeffizienten

In der Gleichung 2.8 wird die Berechnung des Jaccards-Koeffizienten vorgestellt. Mit der Definition eines Schwellwertes für die Jaccard-Koeffizienten lassen sich vier Fälle unterscheiden. Eine Darstellung der vier Fälle erfolgt in Abbildung 2.9 mit der Verwendung einer Konfusionsmatrix und dem beschriebenen Beispiel zur Berechnung des Jaccards-Koeffizientens.
Der Fall True Positive (TP) tritt bei einer Überschreitung des Schwellwerts durch den Jaccard-Koeffizienten von vorhergesagter Bounding Box und Ground Truth Bounding Box ein. Befindet sich in dem Bild ein zu detektierendes Objekt, welches durch den Machine Learning Ansatz nicht detektiert wird, ist der Fall False Negative (FN) erfüllt. Der FN-Fall tritt u. a. beim Unterschreiten des Schwellwertes ein. Mit dem gegenteiligen Beispiel ist der Fall False Positive (FP) erfüllt. Der letzte Fall wird gebildet durch das korrekte Erkennen, eines nicht vorhandenen Bauteils. Dieser Fall wird als True Negative (TN) bezeichnet. TN-Fälle stellen für die zu entwickelnde Objektdetektion keine realistischen Fälle dar, wodurch sie im Rahmen dieser Masterarbeit eine untergeordnete Rolle einnehmen. Eine entscheidende Rolle nehmen TN-Fälle in binären Klassifikatoren ein. Die Berechnung der Genauigkeit erfolgt unter der Berücksichtigung der vier Fälle und wird in Gleichung 2.9 angegeben.

$$Accuracy = \frac{TP + TN}{TN + TP + FP + FN} \tag{2.9}$$

Für die Bewertung von modernen Machine Learning Ansätze werden andere Messgrößen bevorzugt. Die Genauigkeit birgt das Risiko das auf Grund von einem unzureichenden

Rang	Precision	Recall
1	1.0	0
2	1.0	0.2
3	1.0	0.4
4	0.67	0.4
5	0.5	0.4
6	0.4	0.4
7	0.5	0.6
8	0.57	0.8
9	0.5	0.8
10	0.44	0.8
11	0.5	1.0

Tabelle 2.1: Beispiel Precision und Recall

Testdatensatz unzureichende Ergebnisse mit Hilfe des Machine Learning Ansatzes erzielt werden.[18], [19]
Eine alternative Messgröße zu der Genauigkeit stellt der mean Average Precision Wert (mAP) dar. Als Grundlage dienen ebenfalls die bereits vorgestellten vier Fälle, die durch den Jaccard-Koeffizient entstehen. Unter der Verwendung der vier Fälle werden zunächst die Precision und der Recall bestimmt. Dabei gibt die Precision die Genauigkeit der Vorhersagen an. Mit dem Precision-Wert wird eine Aussage darüber getroffen, wie viele Prozent der Vorhersagen korrekt sind. Berechnet wird die Precision durch die Gleichung 2.10. Der Recall ist ein Wert der die Vorhersage in Relation zu den tatsächlichen dargestellten Objekten bildet. Berechnet wird der Recall durch die Gleichung 2.11. Die Berechnung der Precision und des Recalls erfolgt für jedes vorliegende Testbild.

$$Precision = \frac{TP}{TP + FP} \tag{2.10}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.11}$$

Für eine vereinfachte Erläuterung des mAP-Wertes wird ein Beispiel genutzt, dass sich auf die Detektion des bereits vorgestellten Getriebebauteils bezieht. Somit wird vorerst nur eine Klasse für das Beispiel betrachtet. Die beispielhaft berechneten Werte für Precision und Recall sind in Tabelle 2.1 aufgelistet. Eine Sortierung der Ergebnisse bezogen auf die aufsteigenden Recall Ergebnisse wurde bereits durchgeführt. Die Darstellungen der Ergebnisse erfolgt mit Hilfe der Precision-Recall Kurve. Dabei repräsentiert die Ordinate

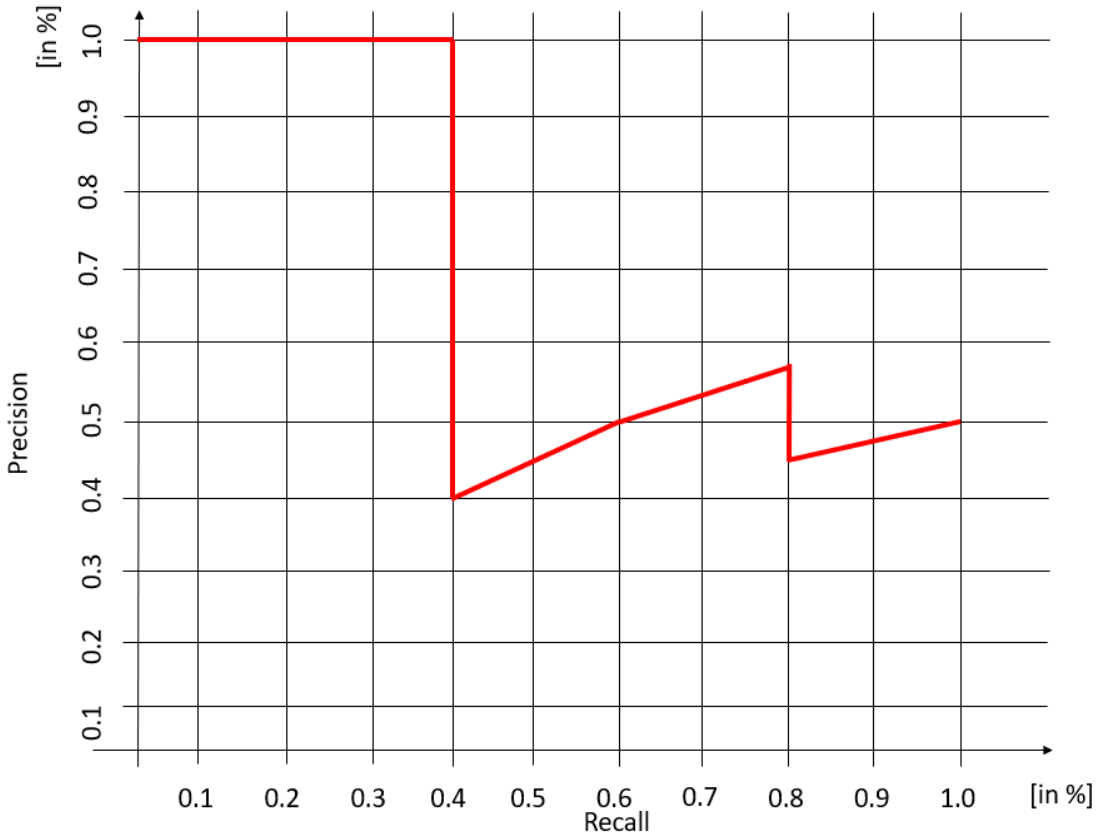


Abbildung 2.10: Precision-Recall Kurve

die ermittelten Werte der Precision, die Abzisse die Werte des Recalls. Abgebildet wird die resultierende Precision-Recall Kurve in Abbildung 2.10 bei einem definierten Schwellwert. Die Fläche unter Kurve stellt die Average Precision (AP) dar. Zum Ausgleich von Schwankungen wird eine Interpolation der Kurve durchgeführt, sodass ein stufenartiger Verlauf entsteht. Mit der Veränderung der Schwellwertes verändern sich die Ergebnisse für Precission und Recall. Dadurch ergibt sich ein anderer Verlauf der Precision-Recall Kurve. Der mAP-Wert ist ein Mittelwert aller betrachteten AP-Werte.

3 Analyse und Anforderung

3.1 Anforderungsanalyse

Eine wesentliche, an die Implementierung der Objektdetektion gestellte Anforderung ist, dass Objekte, die sich im Arbeitsbereich des Roboters befinden, korrekt detektiert werden. Auf Grundlage der Objektdetektion basieren eine Vielzahl weiterer Aufgaben. Die Umsetzung einer Greifoption ist eine Aufgabe der in die Mini-Factory integrierten Roboter. Die Umsetzung des Greifens durch den Roboter ist kein Bestandteil dieser Masterarbeit. Dennoch besitzt die Entwicklung einer Greifoption einen Einfluss auf die Umsetzung der Objektdetektion, da sie als weiterführende Entwicklung betrachtet wird.
Des Weiteren ist für den Zusammenbau von Bauteilengruppen, wie es das bereits erwähnte Getriebe darstellt, eine vordefinierte Reihenfolge notwendig. Aus diesem Grund ist eine Anforderung an die Objektdetektion, dass Bauteile die sich im Arbeitsbereich befinden durch eine Reihenfolge gekennzeichnet werden. Fehlende Bauteile in der Reihenfolge sollen durch die Objektdetektion erkannt und berücksichtigt werden. Das Ende der Objektdetetkion soll signalisiert werden.
Die generierten Informationen über Position und Klasse der detektierten Objekte soll durch die Objektdetektion zur Verfügung gestellt werden, sodass eine Realisierung der Greifoption auf Grundlage der Daten möglich ist. Verschiedene Lichtverhältnisse dürfen keinen Einfluss auf die Objektdetektion besitzen. Die Objektdetektion soll weiterhin in der Lage sein, die detektierten Objekte unter verschiedenen Lichtverhältnissen korrekt zu klassifizieren. Ebenfalls darf kein Einfluss durch Objekte in der industriellen Umgebung auf die Objektdetektion bestehen. Dazu zählen u. a. Fertigungsmaschinen, aber auch metallische Körper wie es Teile eines Flugzeuges darstellen. Der Einsatz der Mini-Factory innerhalb eines Flugszeuges soll dadurch weiterhin gewährleistet sein.
Die Einschränkung der Funktionsweise durch zusätzlich angebrachte Hardware an der Mini-Factory soll vermieden werden. Dadurch soll die Mini-Factory weiterhin mobil bleiben und z. B. mit Hilfe eines Aufzugs transportiert werden können. Des Weiteren wird

durch das Vermeiden zusätzlicher Hardware eine Einschränkung des kollabiereden Menschen unwahrscheinlicher. Die Implementierung der Objektdetektion auf einem SBC ist daher vorzuziehen. Eine detaillierte Beschreibung der Anforderungen, sowie die zu erfüllenden Ziele sind dem Anhang beigefügt.

3.2 Konzept

Die Anforderungsanalyse an die zu implementierende Objektdetektion zeigt auf, dass sich nicht jeder der bereits erwähnten Ansätze eignet. So ist der Ansatz auf Basis von vordefinierten Mustern und Konturen auf Grund der fehlenden Robustheit gegenüber Umwelteinflüssen ausgeschlossen. Zudem ist die Implementierung neuer Muster eine zeitaufwändige Aufgabe, die das Risiko beherbergt, Muster auf Grund menschlichen Versagens nicht zu implementierten.
Der SVM-Ansatz ist ein vielversprechende Alternative zum Muster-Ansatz. Die benötigte Datenmenge zum Trainieren des SVM-Ansatzes ist gering, wodurch eine Umsetzung im Rahmen der Masterarbeit wahrscheinlich ist. Des Weiteren ist es vorstellbar, dass der SVM-Ansatz auf einem SBC implementiert werden kann. Diese Vermutung beweist die bereits im Kapitel 2.1 erwähnte Arbeit von Mizuno. Die Arbeit von Jiang [23] zeigt, dass unter Verwendung einer SVM die Implementierung einer Greiffunktion möglich ist. Dadurch eignet sich der Ansatz für die Implementierung einer Objektdetektion in die Mini-Factory. Jedoch belegt die Platzierung von Simonyan beim ILSVRC 2012, dass CNN-Ansätze zur Objektdetektion besser geeignet sind.
Mit der Verwendung eines CNN-Ansatzes ist ein großer Datensatz erforderlich, wodurch Parameter des Netzwerks eingestellt werden. Des Weiteren ist auf Grund der komplexen Architektur der beim ILSVRC erfolgreichen CNNs zweifelhaft, ob sich ein CNN auf einem SBC implementieren lässt. Die im ILSVRC verwendeten Ansätze werden auf leistungsstarken Systemen angewandt. Es existieren jedoch alternative CNN-Architekturen, die auf leistungsschwächeren Systemen genutzt werden können. Izidio beschreibt [22] die Implementierung einer Detektion von Nummernschildern unter Verwendung eines CNN Ansatzes auf einem Raspberry Pi 3. Dadurch ist eine grundlegende Implementierung einer Objektdetektion, basierend auf einer CNN-Architektur, auf einem SBC bewiesen. Die Anforderung einen SBC für die Objektdetektion zu nutzen ist bei erfolgreicher Implementierung des CNNs erfüllt.
Mit der Verwendung eines Machine Learning Ansatzes für die Umsetzung einer Objektdetektion ist ein Datensatz erforderlich. Die frei verfügbaren Datensätze, wie es bspw.

der ImageNet Datensatz darstellt, eignen sich nur bedingt für die Detektion von Bauteilen, des in der Masterarbeit behandelten Use-Cases. Der ImageNet Datensatz basiert u. a. auf Bildern von Tieren. Abbildung von Getriebebauteilen sind in dem Datensatz nicht vorhanden. Verwendet werden die frei verfügbaren Datensätze zum Trainieren eines Netzwerkes, das grundlegende Eigenschaften von Objekten selektiert. Eine nähere Erläuterung der Problematik erfolgt im Abschnitt 4.1.2. Das Erzeugen des Datensatzes, basierend auf echten Objekten vor echten Hintergründen ist eine zeitaufwändige Aufgabe, die im Rahmen der Masterarbeit nicht bewältigt werden kann. Aus diesem Grund sind alternative Lösungsansätze zum Kreieren des Datensatzes erforderlich.
Eine Alternative stellt das Generative Adversarial Network (GAN) dar. Dabei lässt sich die Funktionsweise der GANs wie folgt zusammenfassen: Für das GAN werden zwei, separate Netzwerke genutzt, um den Datensatz zu erzeugen. Das eine Netzwerk erzeugt die Daten. Das zweite Netzwerk kontrolliert, basierend auf einem bereits vorhandenen Datensatz, das Ergebnis der erzeugten Daten. Verwendetet werden GANs zum Erzeugen des Datensatzes in der Arbeit von Frid-Adar [13], die sich mit der Erkennung von Leberschäden beschäftigt. Frid-Adar beschreibt eine signifikante Verbesserung des genutzten Ansatzes zur Erkennung von Leberschäden unter der Verwendung des erzeugten Datensatzes. Die Verwendung eines GANs für das Erzeugen eines Datensatzes im Rahmen der Masterarbeit ist jedoch ausgeschlossen. Das Getriebe, welches für den behandelten Use-Case genutzt wird, steht erst unmittelbar vor dem Abschluss der Arbeit zur Verfügung. Ein Datensatz, der für das Trainieren des einen GAN-Netzwerks benötigt wird, kann aus diesem Grund nicht erzeugt werden.
Das Erzeugen eines Datensatzes auf Basis von 3D-Modellen stellt eine weitere Alternative dar. Beim Verfolgen dieser Strategie wird das reale Getriebe für das Trainieren des Machine Learning Ansatzes nicht benötigt. Verwendet werden die 3D-Modelle des Getriebes, die als computergenerierte Modelle vorliegen. Auf diese Weise kann ein Netzwerk trainiert werden, ohne dass das reale Getriebe vorliegt. Somit wäre die Mini-Factory in der Lage, Bauteile des Getriebes zu erkennen, bevor das reale Getriebe vorhanden ist. Der Einsatz der Mini-Factory ist unmittelbar mit dem Vorhandenseins der Getriebebauteile gewährleistet. Ein paraller Arbeitsprozess von der Fertigung der Getriebebauteile, wie auch das trainieren des Netzwerkes ist vorstellbar.
Der mit Hilfe von 3D-Modellen erzeugte Datensatz wird in einer idealen Umgebung erzeugt. Dies birgt das Risiko der Sim-to-Real-Gap. Unter dem Begriff der Sim-to-Real-Gap wird die Vernachlässigung realer Umwelteinflüsse in der Simulation verstanden, die beim Transferieren der simulierten Aktionen in die reale Welt zu einem unvorhergesehenen Verhalten des Systems führen. Dabei ist der unter idealen Bedingungen erstelle Datensatz

nicht geeignet, ein Netzwerk zu trainieren, welches reale Objekte erkennen soll. Das Netzwerk ist nicht in der Lage zu abstrahieren und Veränderungen in der realen Umgebungen zu berücksichtigen. Diese Problematik wird in unterschiedlichen Arbeiten behandelt [10], [17], [33]. Tremblay [43] beschreibt, wie unter Berücksichtigung der Sim-to-Real-Gap, ein Datensatz auf Grundlage von 3D-Modelle erzeugt werden kann, der sich eignet ein Netzwerk zu trainieren, dass auch reale Objekte erkennt. Des Weiteren beschreibt Tremblay die automatisierte Generierung des Datensatzes. Damit ist das Erzeugen eines Datensatzes im Rahmen der Masterarbeit möglich.
In Abschnitt 2.2.4 wird die Berechnung des mAP-Wertes vorgestellt. Der mAP-Wert wird dazu genutzt, die trainierten Netzwerke zu bewerten und dadurch eine Vergleichbarkeit herzustellen. Die Vergleichbarkeit bezieht sich jedoch auf die Verwendung des selben Datensatzes. Datensätze wie der ImageNet Datensatz sind darauf ausgelegt die Architekturen miteinander zu vergleichen. Verdeckte Objekte, unterschiedliche Skalierungen der Objekte, sowie andere Eigenschaften der Bilder erschweren es dem Netzwerk die Objekte zu erkennen. Bochkovskiy beschreibt die Entwicklung der vierten Generation des YOLO-Ansatzes [8]. Das Netzwerk wird mit dem MS COCO Datensatz trainiert und getestet. Erreicht wird ein mAP-Wert von 65,7%, bei einer Bildrate von 65 fps. Bochkovskiy bezeichnet die vierte Version von YOLO als "State of the Art" [8]. Zidek [21] generierte einen eigenen Datensatz und erreichte einen mAP-Wert von 99% bei einer Bildrate von 2 fps unter Verwendung des eigenen Datensatzes. Eine Vergleichbarkeit zwischen den Ergebnissen ist schwierig, da kein linearer Verlauf der Objektdetektion vorliegt.
Die in Kapitel 3.1 beschriebene Anforderungsanalyse gibt keine Vorgabe bzgl. der Bildrate. Eine Einschätzung über die benötigte Bildrate ist nicht möglich, da die Mini-Factory bis zur Fertigstellung der Masterarbeit nicht vollständig einsatzfähig ist. Damit eine Beurteilung der im Rahmen dieser Masterarbeit umgesetzten Objektdetektion getroffen werden kann, wird ein Datensatz auf Grundlage von 3D-Modellen erzeugt. Der erzeugte Datensatz wird dafür verwendet zwei CNN-Architekturen zu trainieren und zu testen. Der mAP-Wert wird als Beurteilungsgrundlage verwendet.
Der YOLO-Ansatz repräsentiert die erste zu verwendende CNN-Architektur. Auf Grund der einstufigen Auslegung der Architektur bietet YOLO eine hohe Bildrate. Zudem existieren an leistungsschwache Hardware angepasste Versionen der YOLO-Architektur, wodurch die Implementierung auf einem SBC vielversprechend ist.
Faster-RCNN ist die zweite verwendete CNN-Architektur. Mit der zweistufigen Ausrichtung der Faster-RCNN Architektur ist eine höhere Genauigkeit zu erwarten, jedoch bei einer geringeren Bildrate. Faster-RCNN wird in unterschiedlichen Arbeiten verwendet [7], [24].

4 Umsetzung

4.1 3DExperience

4.1.1 3DExperience-Plattform

Für die Detektion von Objekten mittels eines CNNs ist es notwendig Datensätze anzufertigen, die dazu genutzt werden die Parameter des Netzwerkes anzupassen und das Netzwerk zu überprüfen. In den Wettbewerben, die dazu dienen sollen neue Ansätze im Bereich der Objektdetektion miteinander zu vergleichen, werden Datensätze zur Verfügung gestellt. Der ImageNet-Datensatz repräsentiert einen der zur Verfügung gestellten Datensätze, der im Rahmen des ILSVRC genutzt wurde. Der Datensatz beinhaltet mehr als 15 Millionen Bildern von mehr als 22.000 Klassen. Die Ground-Truth Bounding Boxes des ImageNet-Datensatzes wurden durch Menschen händisch definiert. Fast 49.000 Menschen aus 167 Ländern definierten die Ground-Truth Bounding Boxes [31]. Das händische definieren von Ground-Truth Bounding Boxes wird als labeln bezeichnet. Die aufgenommenen Bilder repräsentieren dabei reale Objekte in realen Umgebungen. Der Datensatz ist öffentlich zugänglich und kann von jedem Anwender eines Machine-Learning Ansatzes genutzt werden.
Besitzt der Anwender jedoch ein Interesse darin Objekte zu erkennen, die nicht durch die Klassen eines öffentlich zur Verfügung gestellten Netzwerkes abgedeckt werden, ist der Anwender dazu gezwungen einen eigenen Datensatz anzufertigen. Das Aufnehmen und das Präparieren von echten Objekten in realen Umgebungen stellt dabei eine zeitintensive Tätigkeit dar. Durch die Verwendung von künstlich erzeugten Objekten, wie auch künstlich erzeugten Umgebungen besteht ein Ansatz den Zeitfaktor zum Erzeugen von Datensätzen zu verringern. Die Objekte und die Umgebungen müssen nicht durch Menschen präpariert werden. Für eine guten Datensatz sind eine Vielzahl von Umgebungen und Szeneariern notwendig. Eine nähere Erläuterung zum Erstellen eines Datensatz erfolgt im Abschnitt 4.1.2.

Die Mini-Factory wird in Produktionshallen, in zu fertigenden Flugzeugen und vergleichbaren Umfeldern eingesetzt. Diese Umfelder werden dazu genutzt Produktgruppen zusammenzubauen oder zu verbauen. Die Umgebungen werden daher im Folgenden als industrielles Umfeld bezeichnet. Im industriellen Umfeld werden künstliche Daten der Objekte bereits bei der Konstruktion entwickelt. Dafür genutzt werden spezielle Computer-Programme, die unter dem Begriff Computer-Aided Design (CAD) geführt werden. Mit Hilfe der CAD-Programme ist es möglich exakte Modelle der Bauteile zu erstellen. Im Folgenden werden die erzeugten Modelle als 3D-Modelle bezeichnet.
Zum Erzeugen eines Datensatzes eignen sich CAD-Programme nicht. Die Varianz der aufgenommenen Bilder ist gering. Die Ursache hierfür ist, dass der Schwerpunkt der CAD-Programme auf der Konstruktion liegt. Unterschiedliche Umgebungsbedingungen sind nicht umsetzbar. Die Qualität des Datensatz ist gering, die Gefahr des Overfittings hoch.
Spezielle Programme zum Erzeugen von Datensätzen, welche auf 3D-Modellen basieren, existieren nicht. In den Ansätzen wird zum Erzeugen des Datensatzes die frei verfügbare Software Blender genutzt. [7] [21] Blender ist eine Software mit dessen Hilfe sich Körper modellieren, texturieren und animieren lassen. Eingesetzt wird die Software bspw. in der Produktion von Animationsfilmen. Integeriert in Blender ist eine Python Konsole, mit dessen Hilfe sich Skripte schreiben und ausführen lassen. Des Weiteren ist es mit Hilfe von Blender möglich die Umgebungsbedingungen anzupassen und 3D-Modelle zu importieren. Die Automatisierung zur Erstellung eines Datensatzes ist auf Grund der Python Konsole mit Blender möglich.
Eine Alternative zu Blender ist die 3DExperience-Plattform. Mit der 3DExperience Plattform von Dassault Systémes existiert eine Umgebung, mit der unterschiedlichste Bereiche der Entwicklung abgedeckt werden. So ist es bspw. möglich Bauteile zu konsturieren oder Simulationen mit Hilfe der konstruierten 3D-Modelle zu erzeugen. Dadurch ergibt sich das Potential vorhandene Fehler frühzeitig zu erkennen und zu beheben. Des Weiteren bietet die 3D-Experience-Plattform die Möglichkeit industrielle Umgebung vor der realen Umsetzung vollständig zu modellieren. Dazu zählen auch Simulationen anzufertigen, die spätere Anwendungsfälle darstellen.
In der Abbildung 4.1 ist das zentrale Element der 3DExperience Plattform dargestellt, der Kompass. Aufgeteilt ist der Kompass in vier Quadranten, wobei jeder Quadrant einen Bereich der Plattform abdeckt. Die Nutzung von Programmen erfolgt durch die Zuweisung von Rollen, denen Applikationen zugeteilt sind.
Für die Konstruktion von 3D-Modellen bietet die Plattform unterschiedlichste Applikationen, die im West-Quadranten des Kompass eingeordnet sind. Dazu zählen z. B. Ap-

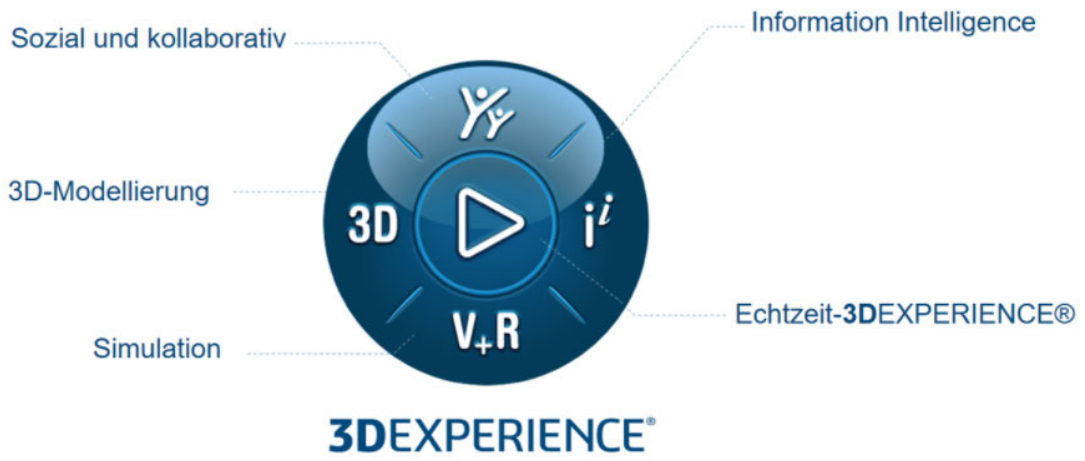


Abbildung 4.1: Kompass der 3DExperience Plattform [1]

plikationen, die zum Programm Catia gezählt werden. Simulationsaufgaben lassen sich mit Hilfe von Programmen aus dem Süd-Quadranten bewältigen. Dafür genutzt werden bspw. Applikationen aus dem Programm Delmia. Der Ost-Quadrant fasst Applikationen zur Informationsbündelung zusammen. Im Nord-Quadranten sind Applikationen einsortiert, die für die Erstellung von Bildern und Videos für das Marketing genutzt werden. Ein Programm des Nordquadranten ist 3D-Excite.
Die 3DExperience-Plattform bietet den Vorteil, dass erstellte Dateien auch mit anderen Applikationen aus anderen Quadranten verwendet werden können. So ist bspw. das Erzeugen eines 3D-Modells mit Hilfe von Applikationen aus dem Ost-Quadranten möglich, welches dann im Rahmen einer Simulation im Süd-Quadranten verwendet wird. Mit der Nutzung der erzeugten Modelle innerhalb der 3DExperience-Plattform werden Barrieren beim transferieren der Daten verringert.
Das Programm 3DExcite-CreativeExperience (Creative Experience) ist im
Nord-Quadranten des Kompasses untergebracht. Es wird für das Erstellen von Bildern und Videos für das Marketing genutzt. Es bietet die Möglichkeit Umgebungen anzupassen und 3D-Modelle hinzuzufügen. Des Weiteren ist eine Konsole in das Programm integriert, welche sich mittels Javascript nutzen lässt. Mit der Konsole ist ein Zugriff auf eine Programmier-Schnittstelle (API) möglich, in der Funktionen zum Verändern der Umgebung definiert sind. Eine Automatisierung zum Verändern der Umgebung ist realisierbar, wodurch das Erzeugen eines ausreichend großen Datensatzes zum Trainieren eines CNNs durch einen geringen Aufwand möglich ist. Jedoch ist der erstellte Datensatz nicht vollständig geeignet, um ein Netzwerk zu traineren. Damit der Datensatz die Voll-

ständigkeit erreicht, müssen die Bilder gelabelt werden. Eine Beschreibung zur Erstellung eines vollständigen Datensatzes ist im Abschnitt 4.1.2 gegeben.
Die Creative Experience ist in die 3DExperience Plattform integriert und eignet sich Barrieren beim Transferieren der Daten zu verringern. Auch andere Anwendungen im Rahmen der 3DExperience Plattform sind umsetzbar. Mit der Nutzung der Creative Experience ist eine Integration in die Arbeitsprozesse mit Hilfe der 3DExperience Plattform, zur vollständigen Simulation einer Anwendung erdenklich. Dadurch ist es möglich Anwendung vor der reellen Umsetzung zu testen.
Für die Erzeugung des Datensatzes wird aus den genannten Gründen die Verwendung der Creative Experience ihm Rahmen dieser Masterarbeit genutzt. Im folgenden Abschnitt 4.1.2 werden auf die Besonderheiten sowie das Vorgehen beim Erzeugen des Datensatzes eingegangen.

4.1.2 Erzeugen des Datensatzes

Im Rahmen dieser Masterarbeit wird ein Datensatz auf Grundlage von 3D-Modellen erzeugt. Dieser Datensatz wird zum Trainieren und Validieren eines CNNs genutzt. Das trainierte CNN wird bei der Klassifizierung und der Detektion realer Bauteile verwendet. Erkannt werden Bauteile eines Getriebes. Das 3D-Modell des Getriebes ist in Abbildung 4.2 dargestellt. Für den ersten Ansatz werden zwei Bauteile des Getriebes ausgewählt, auf dessen Grundlage ein Datensatz erzeugt wird. Zwei Bauteile für die erste Implementierung der Objektdetektion werden als ausreichend bewertet, da eine prinzipielle Multi-Objektdetektion verschiedener Bauteile auf diese Weise überprüft werden kann.
Für die erste Implentierung der Objektdetektion wird eine Welle und die Abdeckung eines Kugellagers gewählt. Diese Bauteile besitzen im Verhältnis zu anderen Bauteilen des Getriebes große Abmaße, wodurch sie in der realen Umgebung leichter zu erkennen sind. Dadurch ist eine Evaluierung des prinzipiellen Vorgehens der Objektdetektion, die in dieser Masterarbeit beschrieben wird, leichter umsetzbar. In der Abbildung 4.3 sind die genannten Bauteile des Getriebes als 3D-Modelle abgebildet.
Die Größe des Datensatzes, der zum Trainieren des CNNs genutzt wird, ist bei der Anfertigung des eigenen Datensatzes eine entscheidende Rolle. Mit der Verwendung der öffentlichen Datensätze, wie es der ImageNet Datensatz darstellt, kann die Größe des Datensatzes mit den individuellen Klassen verringert werden. Der öffentlich Datensatz wird dazu genutzt die Parameter des CNNs einzustellen. Mit eine Methode, die als Transfer Learning bezeichnet wird, wird das bereits trainierte CNN für das individuelle Problem

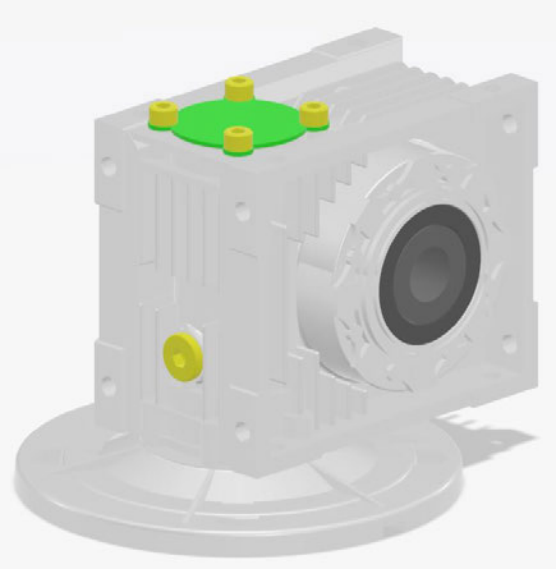
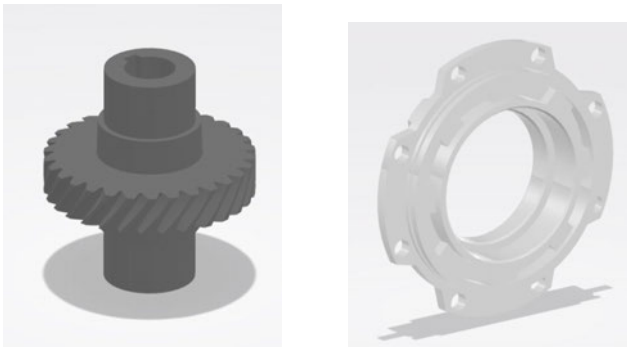


Abbildung 4.2: 3D-Modell eines Getriebes

genutzt. Unter Verwendung des eigenen, im Verhältnis zum ImageNet Datensatz, deutlich kleineren Datensatzes, werden die Parameter des vortrainierten CNNs auf die zu detektierenden Objekte angepasst.
Eine Definition über die Größe des individuellen Datensatzes pro Klasse existiert nicht. Die Werte zu der Größe des Datensatzes basieren auf Erfahrungswerten. Als Erfahrungswert gilt, dass pro Klasse etwa 1000 Bilder zum Trainieren erzeugt werden müssen, um einen ausgeglichenen (balanced) Datensatz zu erzeugen. Dabei ist entscheidend, dass eine hohe Varianz im Trainingsdatensatz besteht. Die Varianz wird durch das Verändern der Parameter, zu denen auch die Umgebung zählt, erreicht. Tremblay beschreibt das Erzeugen eines Datensatz auf Grundlage von 3D-Modellen, welcher dazu verwendet wird echte Objekte zu erkennen.[43] Dabei hat Tremblay herausgefunden, dass künstliche Bilder nicht fotorealistische Nachbildungen der realen Welt darstellen müssen. Vielmehr ist es hilfreich das künstliche erzeugte Bilder möglichst abstrakte Darstellungen wiedergeben. Auf diese Weise sollen die Parameter des Netzwerkes so eingestellt werden, dass das Netzwerk in der Lage ist, unabhängig von der Umgebung die Objekte zu erkennen. Eine Detektion von Objekten durch die Mini-Factory, außerhalb der industriellen Umgebung ist vorstellbar. Dies führt zu einer robusteren Objektdetektion. Das Hinzufügen von Geometrien hilft dem CNN dabei für das Objekt spezifische Merkmale herauszufiltern. Gleiches gilt für die Verwendung von unterschiedlichen Hintergründen und der Zuweisung unterschiedlicher Materialien zu den Bauteilen. Mit dem Erstellen abstrakter Bilder ist



(a) (b)

Abbildung 4.3: 3D-Modell der Bauteile (a) Welle, (b) Kugellager-Abdeckung

es Trembley gelungen ein Netzwerk zu traineren, welches in der Lage ist Objekte besser zu differenzieren. Mit Anwendung einer weiteren Methode, dem Fine Tuning, und Verwendung eines verhältnismäßigen kleines Datensatzes von echten Objekten, vor echten Umgebungen ist es Trembley gelungen Ergebnisse zu erzielen die besser sind, als die Ergebnisse von Netzwerken, die auf Basis eines reellen Labordatensatzes trainiert wurden. Zum Erzeugen des Datensatzes für die Objektdetektion orientiert sich diese Masterarbeit an Tremblay.
Beim Erstellen des automatisierten Prozesses zur Erstellung eines Datensatzes unter Verwendung der Creative Experience sind grundlegende Parameter in das Programm zu integrieren. Zu den Parametern zählen die unterschiedlichen Hintergründe, die in Form von 360 Grad Welten bereits in der 3DExperience Plattform hinterlegt sind. Eine 360 Grad Welt ist ein Bild, dass die Aufzeichnung einer Umgebung darstellt. Dabei wird das Bild um die vertikale Achse von 360 Grad aufgenommen. Unter der Berücksichtigung des Bodens, auf dem sich die vertikale Achse befindet, ensteht dadurch eine Kuppel. Die Verwendung von 360 Grad Welten birgt den Vorteil, dass durch verändern der Position der Objekte unterschiedliche Hintergründe, bei gleicher 360 Grad Welt, entstehen. Für die Erzeugung des Datensatz werden insgesamt 20, 360 Grad Welten in die Creative Experience integriert. Dabei stellen die 360 Grad Bilder auch industriefremde Umgebungen dar. Zu den 360 Grad Welten zählen u. a. Wüsten, Inseln, Lagerhallen, Küchen oder auch Parkhäuser. Mit der Verwendung von industriefremden Umgebungen wird eine abstrakte Umgebung erzeugt. Weitere Parameter, die in die Umgebung integriert werden und dadurch ein höhere Abstraktion erzeugen sind:

- RGB-Lichtquellen
- zusätzliche Geometrien
- Materialien

Mit dem Einfügen der RGB-Lichtquellen ist es möglich, dass unterschiedliche Lichtverhältnisse erzeugt werden. Dadurch werden bspw. Szenarien gebildet, die einer Abenddämmerung oder eingeschaltetem Deckenlicht entsprechen. Der Trainingsdatensatz wird auf unterschiedliche Szenarieren ausgelegt, mit dem ein CNN trainiert wird, das robuster auf wechselnde Lichtverhältnisse angepasst ist. Die vom Getriebe unabhängigen Geometrien stellen Körper wie, Kugeln, Pyramiden, Würfel oder auch Tori dar, welche im Bild positioniert und rotiert werden. Die Rotation bezieht sich in diesem Zusammenhang auf die Achsen der Geometrien. Mit dem Hinzufügen von Geometrien wird ein abstrakteres Bild erzeugt, wodurch das CNN Merkmale der zu detektierenden Objekte von den Geometrien besser differenzieren kann. Ein weiterer Parameter sind Materialien, die sowohl den Geometrien, als auch den 3D-Modellen zugeordnet werden. Besitzen die Objekte und die Geometrien unterschiedliche Materialien, wird das CNN nicht darauf trainiert Objekte an Hand der Materialien zu erkennen. Auch diese Maßnahme führt zu einem robusteren Verhalten der Objektdetektion.
Neben den hinzugefügten Parametern werden 3D-Aktoren in die Creative Experience integriert. Zu den 3D-Aktoren zählen die Kamera und die beiden, zu detektierenden Bauteile des Getriebes. Auch die 3D-Aktoren werden zufallsbasiert positioniert und rotiert.
In der Abbildung 4.4 ist das Erstellen eines Bildes mit Hilfe der Creative Experience dargestellt. In der oberen linken Ecke ist der in Abbildung 4.1 vorgestellte Kompass zur Nutzung der Creative Experience App präsentiert. Auf der linken Seite sind die verwendeten Aktoren aufgelistet. In der Mitte ist das generierte Bild dargestellt, welches die Getriebewelle abbildet. Die Getriebewelle ist in der 3D-Welt einer Landstraße positioniert und besitzt das Erscheinungsbild von Marmor, wodurch es sich von der Getriebewelle in der Abbildung 4.3 unterscheidet. Des Weiteren ist eine Welle aus Marmor auf Grund des schlechten dynamischen Verhaltens ein abstraktes Material für eine Welle. Dies begünstigt die Tatsache, Bilder möglichst abstrakt zu gestalten, wodurch das CNN nicht auf das Material der Objekte trainiert wird. Auf der rechten Seite ist die Konsole dargestellt, welche unter Verwendung der Programmiersprache Javascript die Möglichkeit bietet ein Programm anzufertigen. Am unteren Rand der Abbildung 4.4 sind verschiedene, bereits integrierte Funktionen der Creative Experience App zu erkennen, die durch Bediener

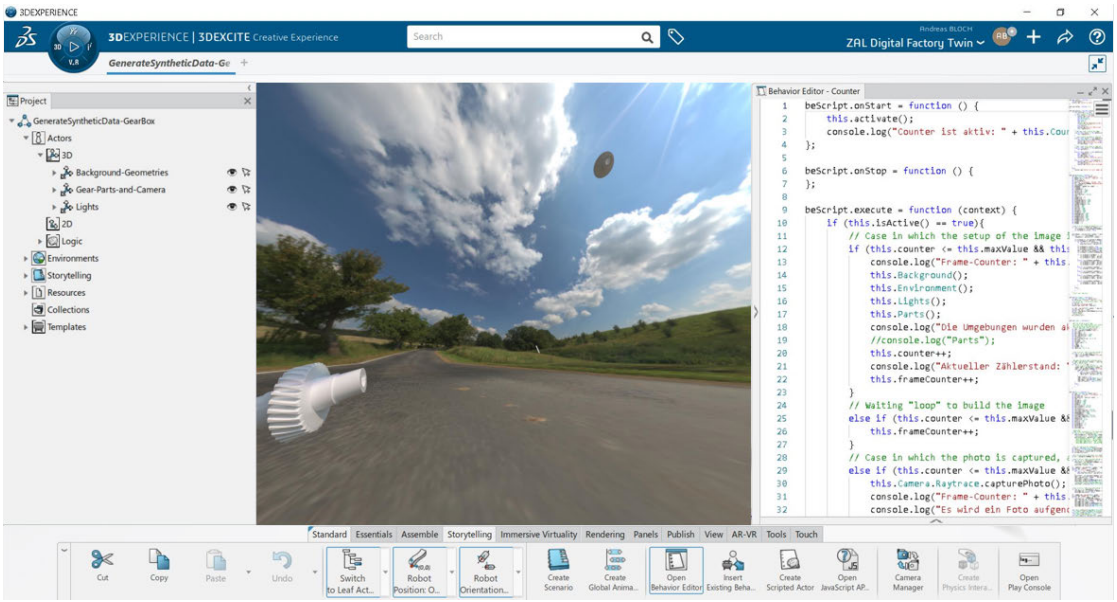


Abbildung 4.4: 3DExcite - Creative Experience

genutzt werden können.
Die Varianz des Datensatz wird durch eine zufallsbasierte Bestimmungen aller Parameter und Aktoren erzeugt. Erreicht wird die zufallsbasierte Zuweisung und Anordnung durch die Erstellung eines Skripts mit Hilfe der in die Creative Experience integrierten Javascript Konsole. Der Aufbau des Skripts ist in einem Aktivitätsdiagram in Abbildung 4.5 dargestellt.
Die Aktivierung des Skripts führt zu einer Abfrage der Menge an Bildern, die erzeugt werden soll. Ist die Menge noch nicht erreicht, setzt das Skript die Umgebung auf einen Ursprungszustand. Dies bedeutet, dass alle zugewiesenen Materialien gelöscht werden und die Positionen der Geometrien, wie auch der RGB-Lichtquellen in den definierten Ursprungszustand versetzt werden. Der verwendete Hintergrund wird gelöscht. Anschließend werden die Parameter neu zugewiesen, sodass eine abstrakte Umgebung entsteht. Auf diese Weise unterscheiden sich die direkt aufeinander folgenden Bilder voneinander, wodurch eine höhere Varianz im Datensatz entsteht. Mit der Positionierung der Kamera wird ein Timer aktiviert, der es der Hardware ermöglicht die Umgebung zu rendern. Der Wert des Timers wird empirisch ermittelt. Erst mit Ablauf des Timers wird mit Hilfe der Kamera eine Aufnahme des Bildes getätigt. Das Bild wird als 2D-Datei gespeichert. Wird die Menge der definierten, aufzunehmenden Menge an Bildern erreicht, endet das Skript.

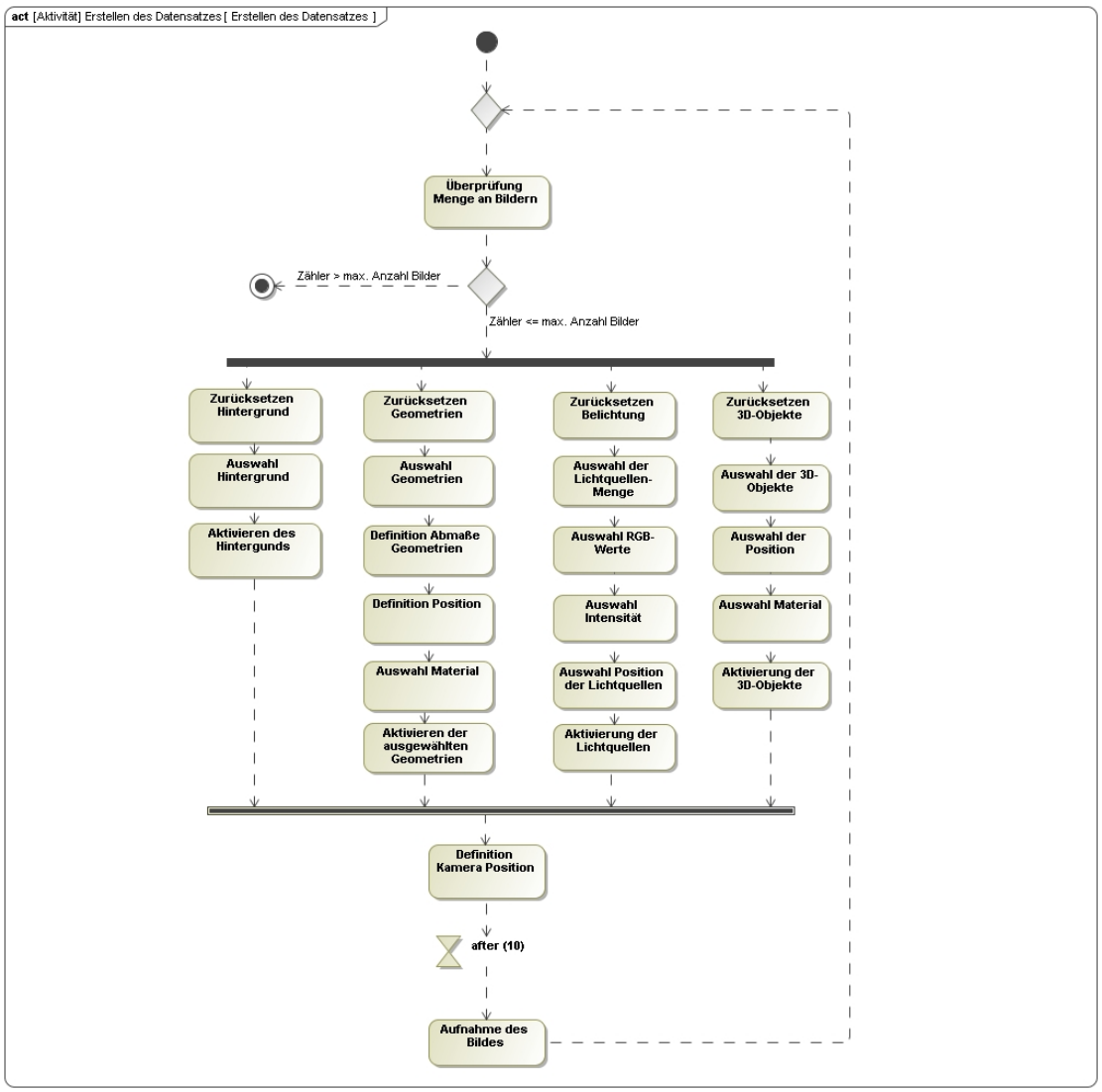


Abbildung 4.5: Aktivitätsdiagramm zum Erstellen eines Datensatzes

```
<annotation verified="yes">
        <folder>test</folder>
        <filename>Generiert_0001.jpg</filename>
        <path>C:\Users\andre\Desktop\test\Generiert_0001.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>1920</width>
                <height>1080</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>shaft</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>1548</xmin>
                        <ymin>262</ymin>
                        <xmax>1746</xmax>
                        <ymax>382</ymax>
                </bndbox>
        </object>
</annotation>
```

Abbildung 4.6: PascalVoc-Format genutzt für den Faster-RCNN Ansatz

Wie in dem Abschnitt 4.1.1 erwähnt, handelt es sich bei den erzeugten Bildern um einen nicht vollständigen Datensatz. Die Creative Experience bietet keine Funktion die Daten für das Traineren eines CNNs zu labeln. Das Labeln des Datensatz wird mit Hilfe der Open Source Software LabelImg händisch durchgeführt. Dabei wird durch einen Menschen jedes erzeugte Bild begutachtet. Die erzeugten Label können sowohl im PascalVoc-, wie auch im YOLO-Format gespeichert werden. Werden die Daten im PascalVoc-Format gespeichert, wird eine xml-Datei, mit dem selben Namen wie das Bild, erzeugt. Genutzt wird das PascalVoc-Format durch den Faster-RCNN Ansatz. Der Aufbau der xml-Datei wird an Hand eines Beispiels in der Abbildung 4.6 dargestellt.
Unterteilt wird die xml-Datei in Informationen zum Speicherort des Bildes, so wie der Größe des Bildes, welches in dem genutzten Beispiel 1920x1080 Pixeln besitzt. Des Weiteren ist die Anzahl der genutzen Farbkanäle in der xml-Datei gespeichert. Da der erzeugte

```
<Objekt-Klasse> <x-Koordinate> <y-Koordinate> <Höhe> <Breite>
```

Abbildung 4.7: Format YOLO-Textdatei

Datensatz auf RGB-Bildern basiert, besitzen die Bilder insgesamt drei Farbkanäle. Der zweite Teil der xml-Datei beschreibt die dargestellte Klasse und die erzeugte Ground-Truth Bounding Box. In dem Beispiel wird ein Label genutzt, dass für Getriebewelle genutzt wird. Die Ground-Truth Bounding Box wird durch zwei Koordinatenpunkte definiert, die durch insgesamt vier Punkte beschrieben werden. Dabei beschreiben xmin und ymin einen Punkt, xmax und ymax den zweiten Punkt.
Beim Abspeichern der Label im YOLO-Format wird eine txt-Datei mit dem selben Namen, wie das Bild erzeugt. Der allgemeine Aufbau der txt-Datei ist in Abbildung 4.7 dargestellt. Die in dieser Masterarbeit entwickelte Objektdetektion ist für die Detektion von zwei Klassen ausgelegt. Die erste Klasse ist die Welle, die zweite Klasse ist die Kugellagerabdeckung. Den Klassen wird eine Nummer zugewiesen, beginnend mit der Nummer 0. Somit besitzt die Welle die Klasse 0 und die Kugellagerabdeckung die Klasse 1. Die x-Koordinate und y-Koordinate beziehen sich auf den Mittelpunkt der Ground-Truth Bounding Box. Mit der Höhe und Breite werden die Abmaße der Ground-Truth Bounding Box angegeben. Für jedes Objekt, dass durch eine Ground-Truth Bounding Box gelabelt wird, wird eine neue Zeile in der txt-Datei mit dem vorgestellten Format angelegt.
Mit der Verwendung von LabelImg wird es möglich den Datensatz zu präparieren. Der erzeugte und präparierte Datensatz eignet sich für das Trainieren und Evaluieren des Faster-RCNN-Ansatzes, wie auch für das Trainieren und Evaluieren des YOLO-Ansatzes. Im Rahmen der Master-Arbeit wurde einen Datensatz mit dem Umfang von etwa 2500 Bildern erzeugt und präpariert. Dabei sind gemäß der üblichen Norm 90% des Datensatzes für das Trainieren des Netzwerkes bestimmt. Die Übrigen 10% des Datensatz werden zum Testen, bzw. zum Evaluieren der trainierten Netzwerke genutzt.

4.2 Mini-Factory

Baugruppen, wie es das Getriebe darstellen bestehen aus mehreren Einzelteilen. Der Zusammenbau der Baugruppe erfolgt in komplexen Montagelinien. Die Montagelinien werden speziell auf eine Baugruppe ausgerichtet, um dadurch die benötigte Zeit zum

Abbildung 4.8: 3D-Modell der Mini-Factory

Montieren einer Baugruppe zu verringern. Mit wechselnder Baugruppe ist ein zeitintensiver Prozess verbunden die Montagelinie umzurüsten. Für den Aufbau von Montagelinien wird oft ein stationärer Ort gewählt. Die Ursache hierfür ist, dass spezielle Maschine für die Montage der Baugruppen benötigt werden. Bei der Montage des Getriebes wäre dies bspw. eine Presse, die für die Fixierung des Kugellagers genutzt wird. Das Kugellager wird unter einer großen Krafteinwirkung, die durch die Presse verursacht wird, in das Gehäuse gepresst. Der Einsatz von Montagelinien in sich schnell ändernden Umgebungen, wie es bspw. ein zu fertigendes Flugzeug darstellt, ist bisher nicht vorgesehen. Montagen im Flugzeug werden überwiegend händisch durch Menschen übernommen. Baugruppen, wie es das Getriebe darstellen, können nicht durch den Menschen im Flugzeug montiert werden. Das Getriebe muss bereits montiert in das Flugzeug gebracht werden. An dieser Stelle sei noch einmal erwähnt, dass das Getriebe, in diesem Kontext, nur als ein abstraktes Beispiel genutzt wird, um die Vorgänge besser beschreiben zu können.
Die Mini-Factory stellt eine mobile Montagelinie dar, die sich zur Kollaboration zwischen Mensch und Roboter anbietet. Sie wird dafür genutzt Baugruppen ortsunabhängig fertigen zu können. Der Einsatz der Mini-Factory in einem Flugzeug ist auf Grund der Felxibilität und der Größe der Mini-Factory vorstellbar. Dadurch ergibt sich die Möglichkeit, auch Baugruppen wie das Getriebe, in einem Flugzeug zu fertigen.
In Abbildung 4.8 ist das 3D-Modell der Mini-Factory dargestellt. Die Mini-Factory besteht aus drei, voneinander unabhängigen Rollwägen, die über einen Mechanismus miteinander gekoppelt werden können. Jeder der Rollwägen besitzt eine Länge von 2 m und eine

Breite von 1 m. Auf zwei der Rollwägen befinden sich auf einem beweglichen Schlitten Roboter. Für die kollaborative Arbeit werden spezielle Roboter verwendet. Zum Einen der Universal Robot-10e (UR10e). Zum Anderen der Bosch Apas, welcher den Kuka KR6 als Basis nutzt. Die genannten Roboter besitzen Sensoren, die es ermöglichen Berührungen zu erkennen. Zur kollaborativen Fertigung einer Baugruppe stehen den Robotern Menschen gegenüber. An dem Rollwagen ohne montierten Roboter werden Arbeiten durch Menschen durchgeführt.
Die für die Objektdetektion benötigte Kamera wird am Roboter UR10e montiert. Verwendet wird die Handgelenkkamera von dem Unternehmen Robotiqs. Diese bietet den Vorteil, dass keine zusätzliche Vorrichtungen an dem Roboter angebracht werden müssen. Die Position der montierten Handgelenkkamera befindet zwischen Roboter und Endeffektor. Die Handgelenkkamera nimmt 2D-Bilder auf, die für die Objektdetektion geeignet sind. Für die Tiefeninformationen, die für die Implementierung einer Greiffunktion benötigt werden, wird eine Roboception rc_visard 160 colour Kamera verwendet. Montiert wird die Kamera zur Erzeugung der Tiefeninformationen an einem Stativ, welches stationär, unabhängig vom Roboter an der Mini-Factory montiert wird.
In der Abbildung 4.8 sind vor dem Roboterarm UR10e Bauteile des Getriebes abgebildet. Diese vermitteln ein besseres Verständnis des in dieser Masterarbeit behandelten Use-Cases. Einzelne Bauteile des Getriebes werden durch Menschen auf der Arbeitsfläche der Mini-Factory positioniert. Die Objektdetektion detektiert und klassifiziert die einzelnen Bauteile des Getriebes. Die durch die Objektdetektion erzeugten Informationen dienen als Grundlage für weiterführende Funktionen, wie es bspw. die Greiffunktion darstellt.

4.3 Single-Board-Computer

Die Mini-Factory ist eine mobile Fertigungsstraße, durch die es ermöglicht wird, Montagen an unterschiedlichen Orten im industriellen Umfeld durchzuführen. Auf Grund der Nutzung von integrierten Rollen ist ein Transport der Mini-Factory möglich. Die Nutzung von engen Räumen, wie es bspw. Aufzüge darstellen, können für den Transport der Mini-Factory genutzt werden. Durch das Hinzufügen von größeren Objekten, wie es bspw. ein leistungsstarker Computer ist, wird der Transport der Mini-Factory erschwert. Dies widerspricht dem Sinn der Mini-Factory. Für die Berechnung der Objekdetetkion wird weitere Hardware benötigt. Zum Trainieren der Machine-Learning Ansätze wird leistungsstarke Hardware genutzt. Die Montage der leistungsstarken Hardware an der Mini-Factory ist basierend auf den genannten Gründe ausgeschlossen. Des Weiteren ist

eine Nutzung von externen Servern für die Berechnung auf Grund von möglichen Signalstörungen, verursacht durch die industrielle Umgebung, nicht umsetzbar. Mit der Verwendung von Einplatinencomputern (engl. Single-Board Computer) steht eine, im Vergleich zur leistungsstarken Hardware, leistungsschwächere Alternative zur Verfügung, die ohne Einschränkungen der Mobilität an der Mini-Factory montiert werden kann. Bei der Verwendung von Single-Board Computern (SBC) ist eine im Vergleich zu einer leistungsstarken Hardware, deutlich schwächere Hardware zu berücksichtigen. Allerdings ist auch zu berücksichtigen, dass das SBC nur für die Ausführung des Machine Learning Ansatzes genutzt wird. Für die Ausführung der trainierten Netzwerke ist eine leistungsschwache Hardware ausreichend. Dennoch muss die Funktionsfähigkeit der Objektdetektion auf einem SBC gewährleistet bleiben. Aus diesem Grund wird im Rahmen der Masterarbeit ein Verfahren genutzt, welches die Auswahl des SBC darstellt. Verglichen werden fünf potentielle SBCs, die mit der Hilfe von acht Punkten bewertet werden. Die fünf miteinander zu vergleichenden SBCs sind im Folgenden aufgelistet.

- ODROID-N2 AMLOGIC S922X (S922X)
- NVIDIA Jetson TX2-Entwicklerkit (NJ TX2)
- Raspberry Pi 4 (RP4)
- Google Coral Dev Board (Coral Board)
- NVIDIA Jetson Nano (NJ Nano)

Die Auswahl der SBCs beruht u. a. auf den bereits in anderen Arbeiten vorgestellten Ansätzen, in denen Machine Learning Algorithmen auf SBCs implementiert werden. So beschäftigt sich bspw. Arva mit der Implementierung einer Gesichtsdetektion auf einem Raspberry Pi [6]. Des Weiteren werden für den Vergleich, SBCs von NVIDIA und das Coral Board genutzt. Die erwähnten SBCs sind für Machine Learning Ansätze ausgelegt. Das ODROID ist im Vergleich zum Raspberry Pi ein leistungsstarkes SBC, wodurch es für die Implementierung einer Objektdetektion denkbar ist.
Die Kriterien zur Bewertung der SBCs werden unter Berücksichtigung verschiedener Aspektpunkte ausgewählt. So ist das Kriterium der GPU/TPU für die Machine Learning Ansätze ein Entscheidendes. Erst durch die Nutzung von leistungsstärkeren GPUs für Machine Learning Ansätze entstand ein erneutes Interesse an der Weiterentwicklung von Machine Learning Ansätzen. GPUs ermöglichen es, eine schnellere Berechnungen der Algorithmen zu realisieren. Ein weiteres, wichtiges Kriterium für die Auswahl des

Kriterium	S922X	NJ TX2	RP4	Coral Board	NJ Nano	Priorität
Arbeitsspeicher	3	4	3	0	3	3
CPU	4	4	2	3	3	4
GPU/TPU	2	4	0	4	3	5
Support	1	3	4	1	3	7
Speicher	2	4	2	2	2	2
Kompatiblität	1	4	4	1	4	6
Betriebssystem	2	2	3	2	2	1
Preis	3	0	4	3	3	8
Ergebnis	78	103	108	75	**111**	

Tabelle 4.1: Auswertungstabelle SBC

SBCs ist die Unterstützung durch Dokumente und Foren. Mit der Verwendung von ausreichenden Dokumenten und Foren ist eine eigenständige Inbetriebnahme des SBCs und die Implementierung der Objektdetektion möglich. Das Kriterium der Kompatibilität beschreibt die Integration und Nutzung von externen Bibliotheken, die die Umsetzung einer Objektdetektion auf einem SBC vereinfachen.
In der Tabelle 4.1 ist der Vergleich der SBCs dargestellt. Aus Gründen der Übersichtlichkeit werden für die SBCs die bereits vorgestellten Abkürzungen verwendet. Eine detaillierte Beschreibung der Hardware ist im Anhang A.1 hinterlegt. Für die Bewertung werden fünf Noten in einem Bereich zwischen 0 und 4 verwendet. Dabei stellt die Note 0 das schlechteste zu erzielende Ergebnis dar. Mit der Note 4 wird das beste, zu erzielende Ergebnis erreicht. Jedem der SBC wird basierend auf den Eigenschaften und unter Berücksichtigung der Kriterien Noten zugewiesen. Des Weiteren werden die Kriterien nach Prioritäten eingestuft. Dafür genutzt werden insgesamt acht Stufen. Die Stufe 1 ist die niedrigste, die Stufe 8 die höchste Priorität. Der Preis des SBCs besitzt im Rahmen dieser Masterarbeit die höchste Priorität auf Grund der Pandemie, welche die Ausgaben des Unternehmens beeinflusst. Das Betriebssystem besitzt die niedrigste Priorität, da Lösungsansätze für Implementierung einer Objektdetektion sowohl für windows- als auch für linuxbasierte Betriebssysteme existieren. Die den SBC zugeteilten Noten, abhängig von den Kritieren, werden mit den jeweiligen Prioritäten multipliziert. Das Ergebnis der Multiplikationen wird pro SBC aufsummiert und in der letzten Zeile der Tabelle 4.1 dargestellt. Das SBC mit dem höchsten Ergebnis eignet sich nach der Auswertung am ehesten für die Implementierung einer Objektdetektion.
Das NVIDIA Jetson Nano ist ein SBC, welches speziell für die Ausführung von Machine Learning Ansätzen entwickelt wurde. Aus diesem Grund sind Hardware-Komponenten für die Erfüllung von Aufgaben aus diesem Themenbereich ausgelegt. Der NJ Nano verfügt

über einen Grafikprozessor, sowie die Möglichkeit externe Bibliotheken und Frameworks, wie Tensorflow, PyTorch oder das Robot Operating System (ROS) zu nutzen. Vor allem ist ROS für die spätere Implementierung einer Greifoption ein Faktor, den es zu berücksichtigen gilt. Die in der Mini-Factory verwendete Tiefenkamera bietet die Möglichkeit aufgenommene Informationen über ROS zur Verfügung zu stellen. Auffällig ist die Platzierung des NJ TX2, die eine Hardware starke Variante des NJ Nano darstellt. Dennoch ist unter der Berücksichtigung des Preises die Nutzung des NJ TX2 im Rahmen dieser Masterarbeit nicht vorgesehen.

4.4 Implementierung

Für die Implementierung der vorgestellten Machine Learning Ansätze werden zwei unterschiedliche Frameworks genutzt. Zum einen das Framework Tensorflow, der für die Impelementierung des Faster-RCNN Ansatzes genutzt wird. Zum anderen der Framework Darknet, der für die Implementierung des YOLO-Ansatzes genutzt wird. Die genutzten Frameworks bieten die Möglichkeit, das bereits unterschiedliche Funktionen integriert sind. Die Betrachtung der entwickelnden Loss-Funktion ist ein Beispiel einer integrierten Funktion, die in beiden Frameworks hinterlegt sind. Des Weiteren sind bereits Skripte vorhanden, mit denen das Training und Testen der Ansätze durchgeführt wird. Dadurch wird die Entwicklung einer Objektdetektion für die Mini-Factory beschleunigt.
Das Training der Machine-Learning Ansätze wird unabhängig vom genutzten Framework auf leistungsstarker Hardware durchgeführt. Als Betriebssystem wird Windows 10 genutzt. Die Nutzung eines anderen Betriebssystems ist auf Grund der Unternehmensvorgaben ausgeschlossen. Verbaut sind in der Workstation zwei GPUs des Typs NVIDIA QUADRO RTX 6000 . Des Weiteren wird eine CPU des Typs Intel Xeon Gold 5220 mit einer Taktfrequenz von 2.20GHz verwendet. Insgesamt stellt die Workstation 96 GB Arbeitsspeicher zur Verfügung.
Tensorflow ist ein Framework, welches ursprünglich von dem Google-Brain-Team für den internen Bedarf entwickelt wurde. Später wurde es unter einer Apache-2.0-OpenSource-Lizenz für die allgemeinene Verwendung freigestellt. Unter Tensorflow ist die Nutzung bereits vorhandener Skripte möglich, wie auch die Implementierung neuer Skripts. Zum Programmieren neuer Skripts werden die Programmiersprachen Python und C++ genutzt. Mit der Verwendung des integrierten Werkzeugs Tensorboard ist es dem Nutzer möglich verschiedenste Graphen und Ergebnisse während und nach dem Training zu beobachten. Durch Tensorflow werden während des Trainings in regelmäßigen, definierten

Abständen Daten gespeichert, auf dessen Grundlage die Graphen erzeugt werden.
Für die Anwendung von Tensorflow wird die frei verfügbare Distribution Anaconda genutzt. Anaconda bietet die Möglichkeit einen abgeschlossenen Raum zu initialisieren, in dem verschiedenste Versionen der verwendeten Software genutzt werden, ohne dabei einen Einfluss auf die Einstellung des Windows Benutzers zu nehmen. Im Rahmen dieser Masterarbeit wird Python 3 genutzt, sowie die Version Tensorflow 1.18. Es existiert bereits eine neuere Tensorflow-Version mit der Bezeichnung Tensorflow 2.2, jedoch kann diese im Rahmen der Masterarbeit nicht genutzt werden. Für das Training des Faster-RCNN Ansatzes werden die bereits vorhandenen Skripte verwendet. Neue Versionen der Skripte vereinfachen die Anwendung, sind jedoch nicht kompatibel mit dem Betriebssystem Windows 10. Aus diesem Grund werden ältere Versionen der Skripte genutzt, die die gleichen Funktionen wie die neuen Skripte bieten und unter dem Betriebssystem Windows 10 angewandt werden können.
Des Weiteren wird die durch NVIDIA zur Verfügung gestellten Programmier-Technik CUDA verwendet. Die Verwendung der Beiden verbauten GPUs ist unter Tensorflow und Windows nicht möglich. Somit wird für das Training des Faster-RCNN Ansatzes lediglich eine GPU genutzt. Tensorflow bietet für die unterschiedlichen Ansätze vortrainierte Netzwerke. Durch den eigenen Datensatz wird das bereits trainierte Netzwerk an das Problem der Objektdetektion angepasst.
Eine weitere Eigenschaft bei der Verwendung von Tensorflow ist die Annotation des erzeugten Datensatzes. Durch das Verändern der Sättigung, des Farbraums, der Helligkeit und anderer Eigenschaften des Bildes werden neue Bilder basierend auf denen durch Creative Expierence erzeugten Bildern erstellt, wodurch der Datensatz vergrößert wird. Ein Beispiel für ein annotiertes Bild ist in der Abbildung 4.9 dargestellt. Für den Leser schwer erkenntlich stellt das Bild die 3D-Welt der Küche dar, in dessen Raum die Getriebewelle zufällig positioniert wird. Auf Grundlage der Annotation wird der Datensatz von ursprünglich ca. 2.200 Bildern auf etwa 500.000 Bilder vergrößert.
Damit das Overfitting verhindert wird, wird in einem Abstand von 2.000 Anpassungen der Parameter das Netzwerk mit Hilfe des Testdatensatzes getest. Das Training wird kurzzeitig unterbrochen. Ist eine deutliche und konstante Verschlechterung des mAP-Wertes erkennbar, ist davon auszugehen, dass ein Overfitting vorliegt. Die durch das Netzwerk eingestellten Parameter werden als Gewichte bezeichnet. Die Gewichte mit dem besten mAP-Wert werden gesondert gespeichert, sodass nach dem Beenden des Trainings die gespeicherten Gewichte genutzt werden können. Die Gewichte stellen einzelne Dateien dar, die mit Hilfe von Datenträgern oder Servern auf andere Systeme transferiert werden können. Dadurch können die erzeugten Gewichte auch bspw. auf einem SBC angewandt

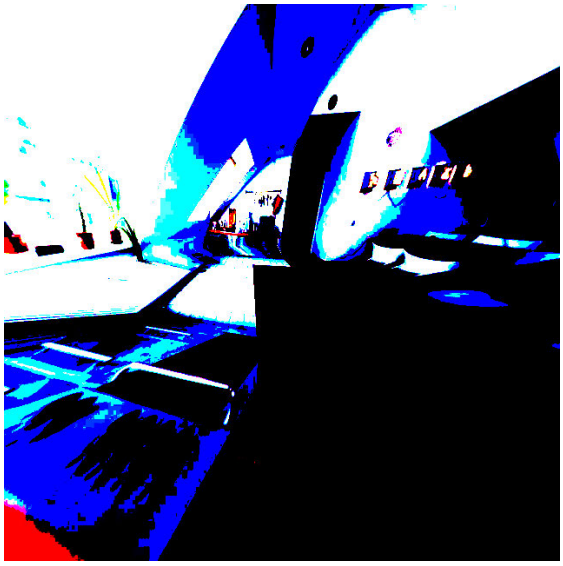


Abbildung 4.9: Annotiertes Bild

werden.
Das Darknet Framework ist ein von Redmon entwickeltes Framework für die Ausführung aller zur Zeit verfügbaren YOLO-Versionen. Für die Objektdetektion der Mini-Factory wird die vierte Version von YOLO verwendet. Die vierte Version von YOLO basiert auf der im Abschnitt 2.2.3 beschriebenen dritten YOLO Version. Allerdings verwendet es als Backbone das CSPDarknet53, eine modifizierte Architektur des in der dritten YOLO Version vorgestellten Darknet53 Architektur. Die Darknet53 Architektur besteht aus ingsesamt 53 Schichten, zum Extrahieren von Merkmalen. Weitere Modifikationen an der Architektur der vierten YOLOv4 Versionen wurden durchgeführt, auf die im Rahmen dieser Masterarbeit nicht näher eingegangen wird. Dadurch ergibt sich ein Ansatz, der im Vergleich zur dritten YOLO Version genauer und schneller ist. Die vierte YOLO-Version ist damit ein vielverpsrechender Ansatz für die Objektdetektion. Mit der Verwendung des Darknets ist es, im Vergleich zum Tensorflow, möglich beide GPUs unter Windows zu nutzen. Das Training wird im Vergleich deutlich beschleunigt.
Vergleichbar mit dem Tensorflow Framework bietet auch das Darknet Framework Skripte die für das Training und das Testen des YOLO-Ansatzes geeignet sind. Mit der Anpassung der Konfigurations-Datei ist es möglich einzelne Parameter, wie bspw. die Lernrate oder die Auflösung auf die das Bild skaliert wird, einzustellen. Des Weiteren existiert eine Option, die die Loss-Funktion in einem Graphen plottet. Auch Darknet nutzt Annotationen für die Vergrößerungen des Trainingsdatensatz. Dadurch ergibt sich ein Trainingsda-

tensatz, der etwa 500.000 Bilder umfasst. Darknet nutzt eine ähnliche Methode, um das Overfitting zu verhindern. Während des Trainings wird dieses in regelmäßigen Abständen unterbrochen. Das Netzwerk wird mit Hilfe des Testdatensatzes getestet und an Hand des mAP-Wertes bewertet. Die Gewichte werden in regelmäßigen Abständen gespeichert. Des Weiteren werden die Gewichte mit dem höchsten mAP-Wert und die Gewichte der letzten Ermittlung des mAP-Wertes gespeichert. Die Gewichte können ebenfalls auf andere Systeme transferiert werden.
Für eine erfolgreiche Implementierung und für einen besseren Vergleich der Leistung auf dem SBC wird neben dem Faster-RCNN Ansatz und dem YOLOv4-Ansatz der Tiny-YOLOv4 Ansatz präpariert. Dieser ist wie der YOLOv4-Ansatz in dem Framework Darknet vorhanden. Tiny-YOLOv4 ist speziell für den Einsatz auf SBCs vorgesehen. Tiny-YOLOv4 besitzt eine Architektur, mit lediglich 27 Schichten zur Merkmalsextraktion. Dies hat einen negativen Einfluss auf die Genauigkeit, jedoch einen positven Einfluss auf die Geschwindigkeit. Die schnellere Berechnung ist vor allem bei der Anwendung auf leistungsschwachen Systemen, wie es der NJ Nano ist, vorteilhaft. Die im Abschnitt 2.2.3 beschriebene Methode zur Detektion von Objekten wird ebenfalls bei Tiny YOLOv4 angewandt.

5 Ergebnisse und Diskussion

5.1 Evaluierung Ansätze - synthetischer Datensatz

Die in diesem Kapitel beschriebene Auswertung der Ergebnisse bezieht sich auf die Nutzung des Testdatensatzes, welcher auf 3D-Modellen basiert. Eine Anwendung des trainierten Netzwerks auf einen Datensatz, der auf echten Bildern basiert, wird in Kapitel 5.2 näher betrachtet. Vorgestellt werden die Ergebnisse der gewählten Ansätze zur Implementierung einer Objektdetektion. Als Grundlage für die Bewertung werden die Loss-Funktion, der mAP-Wert, sowie die durch den Ansatz bentötigte Berechnungsdauer genutzt. Betrachtet wird, soweit möglich, die Performanz auf einem leistungsstarken Computer, wie auch auf einem SBC.
In Abbildung 5.1 ist die Loss-Funktion des Faster-RCNN Ansatzes dargestellt. Berechnet wird die Loss-Funktion mittels der Gleichung 2.1. Die Ordinate repräsentiert die Abweichung des Netzwerks zu dem tatsächlichen Ergebnis, basierend auf dem Ergebnis der Loss-Funktion. Bei einem Wert von 0, würde keine Abweichung vorliegen. Auf der Abzisse wird eine Größe mit dem Namen Epoch aufgetragen. Dabei ist ein Epoch ein sogenannter Hyperparameter, der angibt wie häufig der vollständige Datensatz für das Training durchlaufen wird. Die in Abbildung 5.1 dargestellte Loss-Funktion wird

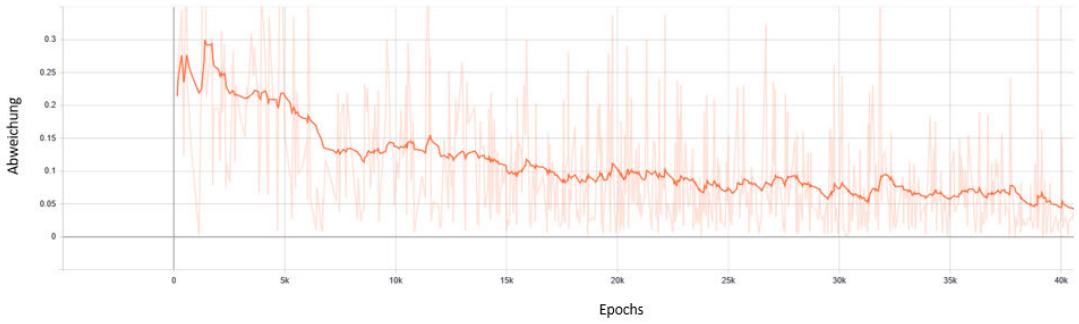


Abbildung 5.1: Graph Loss-Funktion Faster-RCNN

Abbildung 5.2: Faster-RCNN angewandt

bis zu einem Epoch von 40.000 dargestellt. Somit ist der Trainingsdatensatz 40.000 mal durchlaufen wurden. Der Graph wird mit Hilfe des in Tensorflow integrierten Werkzeugs, Tensorboard erzeugt. Durch Anwendung der Interpolation werden Ausreißer im Graphen geglättet. Der interpolierte Graph ist als markante Linie dargestellt. Ausreißer des Graphen werden als schemenhafte Linie in den Graph gezeichnet. Mit der Berechnung der ersten Abweichung nach einem Epoch besitzt das Netzwerk eine Abweichung von 0,22. Die Anpassung der Parameter führt zu einer geringern Abweichung. Mit dem 40.000 Epoch erreicht das Netzwerk eine Abweichung von etwa 0,048. Das Training hat über einen Zeitraum von 72 Stunden stattgefunden. Verwendet wird eine GPU der Workstation. Erreicht wird ein mAP-Wert von 0,81. Für die Auswertung eines Bildes benötigte der Faster-RCNN Ansatz etwa sieben Sekunden, dabei beträgt die Auflösung der genutzten Bilder 1920x1080 Pixel. Der Schwellwert, mit dem Objekte durch das Faster-RCNN Ansatz als erkannt betrachtet werden liegt bei 50 %.
Die Abbildung 5.2 zeigt die Anwendung des trainerten Faster-RCNN Ansatzes auf eines der Testbilder, die mit Hilfe der Creative Experience App erstellt werden. In dem Testbild sind insgesamt fünf Getriebewellen und zwei Kugellager-Abdeckungen zufällig in der 3D-Welt eines Büros positioniert. Dabei ist die Ausrichtung, das Material und der Abstand zwischen der Kamera und der Getriebebauteile unterschiedlich. Des Weiteren

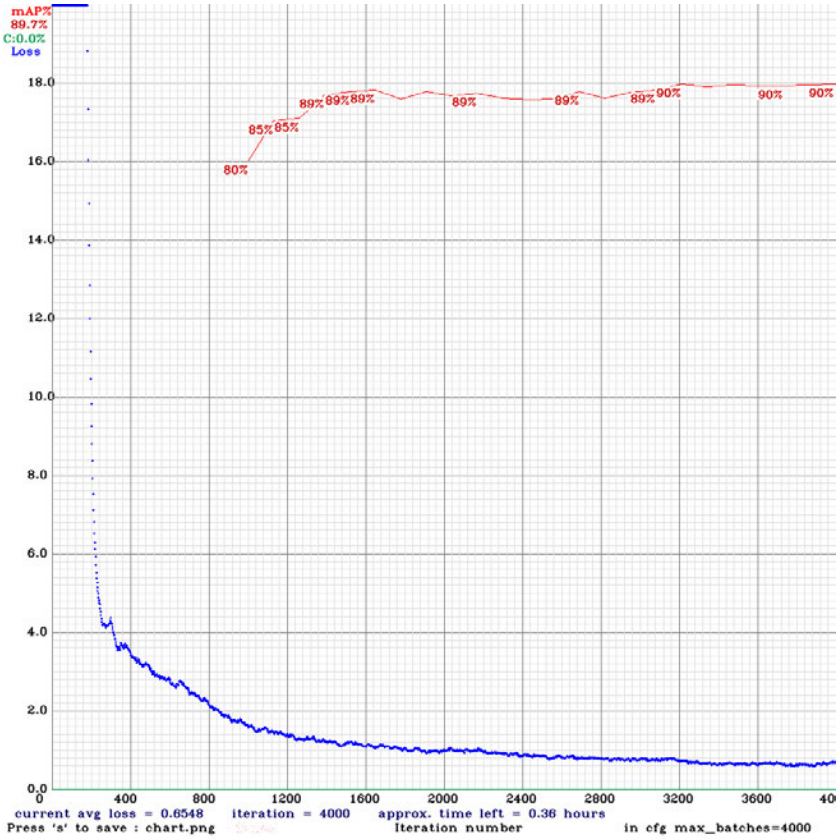


Abbildung 5.3: Graph Loss-Funktion YOLOv4

sind geometrische Formen, wie der Torus im Raum platziert. Das Testbild ist lediglich eins der verwendeten Bilder des Testdatensatzes. Die durch den Faster-RCNN Ansatz erkannten Objekte sind mit einer Bounding Box und der Genauigkeit des Faster-RCNN Ansatzes, dass es sich um das benannte Objekt handelt, gekennzeichnet.
In der Abbildung 5.2 ist zu erkennen, dass der trainierte Faster-RCNN nicht in der Lage ist alle Objekte richtig zu klassifizieren. So klassifiziert der Faster-RCNN Ansatz den Torus als Shaft, mit einer Wahrscheinlichkeit von 74 %. Darüber hinaus sind keine fehlerhaften Klassifizierungen zu erkennen. Positiv hervorzuheben ist die Genauigkeit der positionierten Bounding Boxes. Die erzeugten Bounding Boxes umschließen die zu detektierenden Objekte. Des Weiteren besitzt der Faster-RCNN Ansatz eine hohe Sicherheit bei den korrekt detektierten Objekten. Objekte mit einer größeren Distanz zu der Kamera werden ebenso klassifiziert und detektiert, wie Objekte mit einer geringeren Distanz zu der Kamera. Die Rotation der Getriebebauteile, wie auch die unterschiedlichen Materialien scheinen keinen Einfluss auf die Objektdetektion zu besitzen.

Abbildung 5.4: YOLOv4 angewendet

In der Abbildung 5.3 ist die Loss-Funktion des YOLOv4 Ansatzes, sowie der zur Überprüfung erzeugte mAP-Wert abgebildet. Die Graphen werden mit Hilfe der Funktion, welche in das Darknet-Framework integriert wurde, erzeugt. Der blaue Graph repräsentiert die Loss-Funktion, der rote Graph repräsentiert den ermittelten mAP-Wert. Auf der Ordinate ist die Abweichung des Netzwerkes zu dem tatsächlichen Wert dargestellt. Vergleichbar mit dem Faster-RCNN Ansatz, ist der Idealwert der Abweichung ebenfalls bei 0. Auf die Abzisse werden die Batches eingetragen. Batches sind Hyperparameter, die Trainingsdaten in vordefinierten, kleineren Gruppen repräsentieren. Erst mit dem vollständigen Durchlauf des Batches werden die Parameter des Netzwerkes angepasst. Für die Nutzung des YOLO-Ansatzes werden durch Redmon eine maximale Anzahl an Batches vorgeschlagen. Dabei basiert der Vorschlag auf den zu detektierenden Klassen. Für die in der Objektdetektion, zwei zu detektierenden Klassen schlägt Redmon 4.000 Batches vor. Die erste berechnete Abweichung liegt bei einem Wert, der außerhalb der Skala liegt. Jedoch besitzt dieser Teil der Kurve keine endgültige Aussagekraft über das Lernverhalten des Netzwerkes und wird daher nicht weiter betrachtet. Die für die Beurteilung des Trainings entscheidende Entwicklung der Kurve wird durch den Graphen dargestellt. Der erste in dem Graphen abzulesende Wert liegt bei etwa 180 Batches und weist eine Abweichung von 18,8 auf. Das Training wird nach 4.000 Batches beendet. Die Lernkurve besitzt einen Wert von 0,6548. Die erste Berechnung des mAP-Wertes

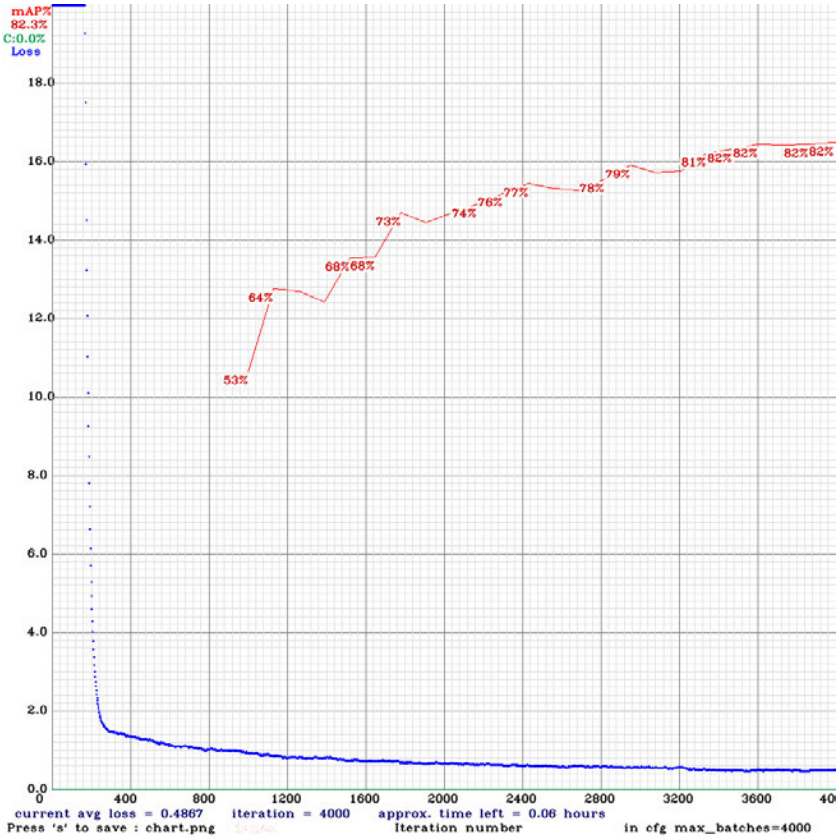


Abbildung 5.5: Graph Loss-Funktion Tiny-YOLOv4

erfolgt nach 1.000 Batches. Bei etwa 360 Batches ist ein Anstieg der Lernkurve zu erkennen. Die Ursache für dieses Verhalten der Lernkurve liegt in der falschen Einstellung der Parameter durch den Algorithmus. Der mAP-Wert wird in Prozent angegeben. Das Netzwerk erreicht nach 4.000 Batches einen mAP-Wert von 89,7 % und erreicht damit einen höheren mAP-Wert als der Faster-RCNN Ansatz. Der mAP-Wert wird aufgerundet in den Graphen eingezeichnet. Für die Berechnung benötigt der YOLOv4-Ansatz etwa 0,05 Sekunden pro Bild und ist damit im Vergleich zum Faster-RCNN Ansatz wesentlich schneller.

Für einen besseren Vergleich wird das trainierte YOLOv4-Netzwerk auf das selbe Testbild wie in Abbildung 5.2 angewandt. Das Ergebnis des angewandten YOLOv4-Ansatzes auf das Testbild wird in Abbildung 5.4 dargestellt. Im Vergleich zum Faster-RCNN Netzwerk ist es dem YOLOv4-Netzwerk möglich alle Objekte von dem geometrischen Figuren zu differenzieren. Der durch den Faster-RCNN detektierte Torus, wird durch das YOLOv4-Netzwerk nicht als Getriebewelle klassifiziert. Allerdings klassifiziert das

Abbildung 5.6: Tiny YOLOv4 angewendet

YOLOv4-Netzwerk vier der fünf Getriebewellen. Eine der durch den YOLOv4-Ansatz klassifizierte Kugellagerabdeckung ist eine der Getriebewellen. Dennoch ist positiv hervorzuheben, dass auch Objekte mit einer größeren Distanz durch das YOLOv4-Netzwerk erkannt werden. Ebenfalls positiv hervorzuheben sind die durch den YOLOv4-Ansatzes erzeugten Bounding Boxes.
In der Abbildung 5.5 ist die Lernkurve des Tiny-YOLOv4 Ansatzes dargestellt. Die Achsenbeschriftung entspricht der bereits vorgestellten Achsenbeschriftung des YOLOv4-Ansatzes. Nach 180 Batches besitzt der Tiny-YOLOv4 Ansatzes eine Abweichung von 19,2. Nach 4.000 Batches besitzt Tiny-YOLOv4 eine Abweichung von 0,4867. Der mAP-Wert beträgt nach 4.000 Batches 82,3 %. Für die Auswertung jedes Testbildes benötigt der Tiny-YOLOv4 Ansatz eine Zeit von 0,034 Sekunden. Im Vergleich zum YOLOv4-Ansatz benötigt der Tiny-YOLOv4-Ansatz weniger Zeit, um ein Bild zu verarbeiten, jedoch wird durch die Testergebnisse bestätigt, dass der Tiny-YOLOv4-Ansatz eine schlechtere Genauigkeit aufweist. Der Faster-RCNN-Ansatz ist sowohl in der Genauigkeit, als in der Geschwindigkeit dem Tiny-YOLOv4-Ansatz unterlegen.

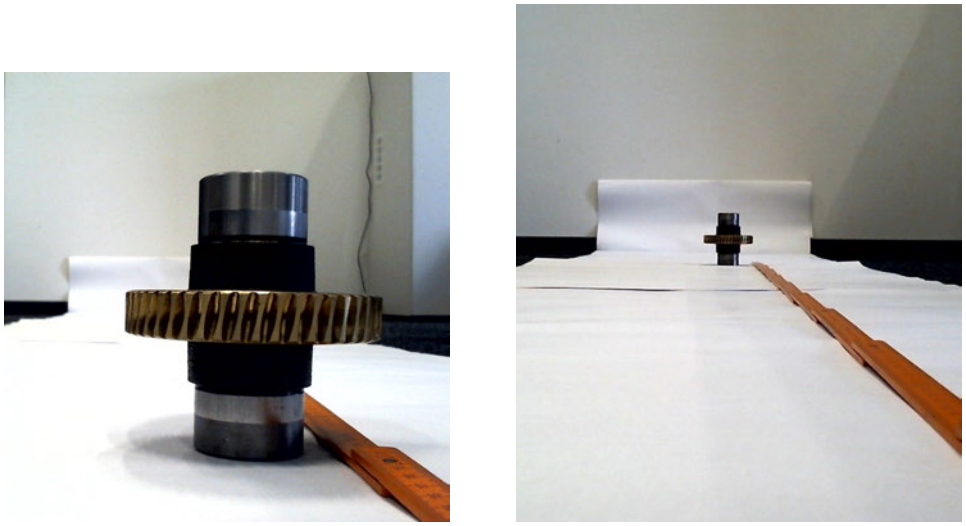
In der Abildung 5.6 ist das Ergebnis des trainierten Tiny-YOLOv4 Ansatzes auf das Testbild angewandt. Tiny-YOLOv4 ist in der Lage zwei der Getriebewellen und eine Kugellager-Abdeckung zu detektieren und richtig zu klassifizieren. Weitere Objekte wurden durch den Tiny-YOLOv4 Ansatz nicht erkannt. Aufällig bei diesem Ergebnis ist, dass

die Objekte mit einer größeren Distanz zur Kamera nicht detektiert werden. Kleine Objekte werden somit durch den Tiny-YOLOv4 Ansatz nicht erkannt. Der Tiny-YOLOv4 Ansatz ist in der Lage die geometrischen Figuren nicht als Objekte zu klassifizieren.
Bei einem Vergleich der drei vorgestellten Ansätze ist eine Implementierung des Faster-RCNN Ansatzes auf dem SBC nicht weiter zielführend. Die Geschwindigkeit zur Auswertung eines Bildes auf einem leistungsstarken Rechner beträgt das 140 fache der Geschwindigkeit des YOLOv4-Ansatzes. Mit der Implementierung des Faster-RCNN Ansatzes auf einem SBC ist die erwartete Geschwindigkeit des Faster-RCNN Ansatzes zur Auswertung des Bildes um ein vielfaches geringer. Des Weiteren ist der mAP-Wert des Faster-RCNN Ansatzes nierdriger als die vermeindlich schwächere Architektur des Tiny-YOLOv4 Ansatzes. Auch wenn der Vergleich eines einzelnen Testbildes einen vermeintlich positiven Eindruck des Faster-RCNN Ansatzes erzeugt. Jedoch dient die Darstellung der Anwendung des Ansatzes auf ein Testbild lediglich für das bessere Verständnis und das Aufzeigen von potentiellen Schwachstellen der einzelnen Ansätze. Der Faster-RCNN Ansatz ermöglicht es eine bessere Einschätzung über die Performanz der vorgestellten Ansätze zu treffen. Aus diesen Gründen wird die Nutzung des Faster-RCNN Ansatzes für die zu implementierende Objektdetektion nicht weiter berücksichtigt. Der Tiny YOLOv4-Ansatz besitzt im Vergleich zum Faster-RCNN Ansatz einen beseren mAP-Wert und wertet die Bilder schneller aus als der YOLOv4 Ansatz. Auch wenn der Tiny-YOLOv4 Ansatz nicht in der Lage ist, alle Objekte des Testbildes zu detektieren, versprechen der mAP-Wert und die Geschwindigkeit eine erfolgreiche Implementierung der Objektdetektion auf einem SBC. Für die weitere Implementierung der Objektdetektion auf dem SBC werden daher nur noch der YOLOv4 Ansatz und der Tiny YOLOv4 Ansatz berücksichtigt.
Auf dem verwendeten SBC, dem NJ Nano, wird das Betriebssystem Ubuntu 18.04 installiert, sowie CUDA 10.2 und OpenCV 4.3. Des Weiteren wird für die Nutzung der YOLO-Ansätze auch das Framework Darknet eingerichtet. Die auf dem leistungsstarken Rechner erzeugten Gewichte für die YOLO-Ansätze werden auf den NJ Nano übertragen. Für den Vergleich der unterschiedlichen Plattformen zur Ausführung der YOLO-Ansätze wird der gleiche Testdatensatz, basierend auf den 3D-Modellen, auch auf dem NJ Nano angewandt. Sowohl der YOLOv4 Ansatz, als auch der Tiny YOLOv4-Ansatz besitzen auf dem NJ Nano dieselben mAP-Werte, wie auf dem leistungsstarken Rechner. Lediglich die Zeit zur Auswertung der Bilder unterscheidet sich zwischen den Plattformen. Der YOLOv4 Ansatz benötigt für die Auswertung eines Bildes auf dem SBC eine Zeit von etwa 3 Sekunden und ist damit um das 60 fache langsamer, als auf der leistungststarken Hardware. Der Tiny-YOLOv4 Ansatz benötigt für die Auswertung eines Bildes 0,3 Sekunden und ist damit um das 10 fache schneller als der YOLOv4-Ansatz, auf dem NJ

Nano. Des Weiteren benötigt der Tiny-YOLOv4 Ansatz auf dem NJ Nano das 10 fache der benötigten Zeit, als auf der leistungsstarken Hardware.
Die Implementierung beider Ansätze auf einem SBC ist erfolgreich. Somit eignen sich sowohl der YOLOv4-Ansatz, wie auch der Tiny-YOLOv4 Ansatz für die Objekdetektion. Eine endgültige Beurteilung über den zu nutzenden Ansatz kann auf Grund der Ergebnisse jedoch nicht getroffen werden. Es konnte bis zum Abschluss dieser Masterarbeit nicht geklärt werden, in welcher Geschwindigkeit die Bilder ausgwertet werden müssen, um ein funktionsfähiges System zu gewährleisten. Wie in Abschnitt 2.2.3 bereits erwähnt, gibt es keine allgemeingültige Aussage, ob ein Ansatz echtzeitfähig ist. Auch wenn der YOLOv4 Ansatz langsamer, als der Tiny-YOLOv4 Ansatz ist, besitzt dieser jedoch eine höhere Genauigkeit, was sich positiv auf die folgende Entwicklung einer Greiffunktion und dem damit vielversprechenden Anheben des Objektes auswirkt. Auch das Detektieren von Objekten, die kleiner sind, scheint auf Grund der Anwendung des YOLOv4 Ansatzes auf ein Testbild vielversprechender zu sein. Für die Nutzung des Tiny-YOLOv4 Ansatzes spricht die durch das Netzwerk benötigte Zeit zur Auswertung des Bildes. Mit der Definition der benötigten Geschwindigkeit der Mini-Factory könnte dies ein entscheidender Faktor bei der Auswahl des genutzten Ansatzes sein.
Ein weiterer Faktor, der bei der Auswahl des für die Objektdetektion genutzten Ansatzes berücksichtigt werden muss, ist ob die Ansätze in der Lage sind auch reale Objekte zu erkennen. Mit der im Abschnitt 5.2 beschriebenen Evaluierung der Netzwerke wird untersucht, ob die im Abschnitt 3.2 erwähnte Sim-to-Real Gap durch die entwickelte Objektdetektion geschlossen wird. Des Weiteren wird untersucht, welche Grenzen sich durch die Objektdetektion ergeben. Dabei werden Faktoren, wie bspw. die Helligkeit berücksichtigt.

5.2 Evaluierung Ansätze - realer Datensatz

Die Entwicklung der Objektdetektion für die Mini-Factory basiert auf der Verwendung von 3D-Modellen zum Erzeugen des Datensatzes. Die im Abschnitt 3.2 angesprochene Sim-to-Real Gap ist ein Faktor der bei der Entwicklung der Objektdetektion berücksichtig wird. Dieses Kapitel beschreibt unterschiedliche Tests, die für die Evaluierung der Objektdetektion durchgeführt werden und für die Untersuchung der Objektdetektion bei Anwendung auf reale Daten dienen. Die durchgeführten Tests werden nicht unter Laborbedingung angewandt. Genutzt werden die trainierten Ansätze von YOLOv4 und Tiny YOLOv4 auf Grund der im Abschnitt 5.1 erwähnten Möglichkeiten zur Implementierung



(a) (b)

Abbildung 5.7: Distanztest (a) Distanz Kamera-Objekt 30cm, (b) Distanz Kamera-Objekt 190cm

auf einem SBC. Untersucht wird des Weiteren, ob mit der entwickelten Objektdetektion die Sim-to-Real Gap verringert wird und somit eine funktionsfähige Objektdetektion in die Mini-Factory implementiert wird.
Für die Ermittlung des mAP-Wertes wird in dem Abschnitt 5.1 ein Testdatensatz genutzt, um die vorgestellten Ansätze miteinander zu vergleichen. Der Testdatensatz besteht aus 233 Bildern, die durch die Nutzung der Creative Experience App künstlich erzeugt werden. Auf Grund des Zeitaufwands, den das Erzeugen eines Datensatzes auf realen Bildern in der selben Größe benötigt, besitzt der Testdatensatz basierend auf realen Daten einen Umfang von 150 Bildern. Der reale Datensatz wird in einer industriefremden Umgebung erzeugt. Dabei stellt die industriefremde Umgebung das Büro von Dassault Systèmes im ZAL dar. Verwendet werden für die Erstellung des realen Datensatzes die bereits vorgestellten Getriebebauteile, die jeweils in doppelter Ausführung vorliegen. Somit werden für jedes Bild zwischen einem und vier Objekten im Raum positioniert. Die Lichtverhältnisse werden zum einem von der Deckenbeleuchtung, zum Anderen von der natürlichen Tagesbeleuchtung beeinflusst. Der reale Datensatz wird an zwei unterschiedlichen Tagen erzeugt. Auf den Bildern werden neben den Getriebebauteilen auch für ein Büro übliche Objekte abgelichtet. Dazu zählen z. B. Stühle, Tische oder auch Bildschirme. Für die Aufnahme der Bilder wird die Logitech B525 HD Webcam genutzt. Zum Labeln der Daten wird die in Abschnitt 4.1.2 vorgestellte Software LabelImg verwendet.

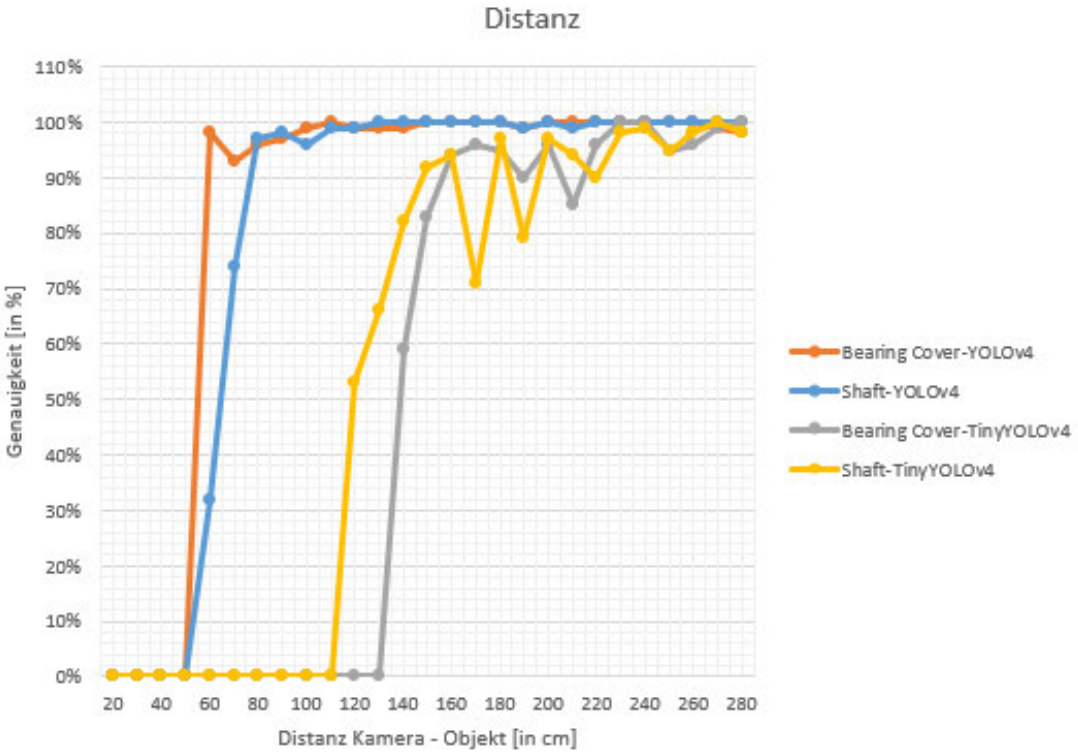


Abbildung 5.8: Distanz Kamera-Objekt

Für die Bestimmung des mAP-Wertes wird das in das Darknet integrierte Skript verwendet. Der YOLOv4-Ansatz erreicht unter Verwendung des realen Datensatzes einen mAP-Wert von 72,6 %. Der Tiny-YOLOv4 Ansatz erreicht einen mAP-Wert von 31,3 %. Im Vergleich zu den mAP-Werten der Ansätze, die mit Hilfe des Datensatzes, welcher auf 3D-Modellen basiert, getestet werden, ist eine deutliche Differenz erkennbar. Die im Folgenden vorgestellten und durchgeführten Tests dienen der detaillierteren Betrachtung der Objektdetektion, zur Bestimmung von Grenzen und Aufdeckung von möglichen Ursachen für die Differenz der mAP-Werte zwischen den erzeugten Testdatensätzen.
Mit dem ersten Test werden die Grenzen untersucht, bis zu welcher Distanz zwischen Kamera und Objekt die Objektdetektion die Objekte detektiert. Dabei wird sich an dem realen Aufbau der Mini-Factory orientiert. Die für die spätere Objetdetektion der Mini-Factory verwendete Kamera wird zwischen dem Endeffektor und dem UR10e montiert. Aus diesem Grund wird für den Test eine minimale Distanz zwischen Kamera und Objekt von 20 cm gewählt. Als Messgröße für die Distanz wird die äußerste Kante des zu untersuchenden Objektes gewählt. Die Position und die Ausrichtung des zu untersuchenden

(a) (b)

Abbildung 5.9: Objekt verdeckt (a) Objekt zu 0% verdeckt, (b) Objekt zu 50% verdeckt

Objektes wird bei der Durchführung des Testes nicht verändert. Das zu untersuchende Objekt wird beginnend mit einer Distanz von 20 cm, in 10 cm Schritten von der Kamera entfernt. Als maximale Grenze wird eine Distanz von 280 cm gewählt. Ermittelt wird die Genauigkeit, mit der die Objektdetektion die Objekte detektiert. Mit jedem Schritt wird ein 2D-Bild aufgenommen. Die Bilder werden sowohl von der Getriebewelle, als auch von Kugellagerabdeckung erzeugt. In der Abbildung 5.7 ist der Testaufbau an Hand der Getriebewelle abgebildet. Dargestellt werden zwei unterschiedliche Distanzen zwischen der Kamera und dem Objekt.
Der für die Evaluierung der Distanz erzeugte Datensatz wird sowohl auf dem Tiny-YOLOv4 Ansatz, wie auch auf dem YOLOv4 Ansatz angewandt. Die Messergebnisse sind in der Abbildung 5.8 in einem Koordinatensystem eingezeichnet. Dabei ist auf der Ordinate die ermittelte Genauigkeit in Prozent aufgetragen. Auf der Abzisse ist die Distanz zwischen Kamera und Objekt in 10 cm Schritten aufgetragen, beginnend bei 20 cm. Insgesamt sind vier Graphen eingezeichnet. Dabei ist ein Graph das Messergebnis bei der Anwendung des Tiny-YOLOv4 Ansatzes auf den Distanztest bei der Getriebewelle. Ein weiterer Graph ist das Messergebnis bei Anwendung des Tiny-YOLOv4 Ansatzes auf den Distanztest bei Verwendung der Kugellagerabedeckung. Gleiches wird für den YOLOv4 Ansatz wiederholt, wodurch weitere zwei Graphen in das Koordinatensystem eingezeichnet sind.
Die grafische Darstellung zeigt, dass sowohl der Tiny-YOLOv4 Ansatz, wie auch der YOLOv4-Ansatz Objekte die nah an der Kamera sind nicht detektiert. Wobei eine deutliche Differenz zwischen den Ansätzen zu erkennen ist. Der Tiny-YOLOv4 Ansatz detektiert zum ersten Mal ein Objekt bei einer Distanz von 110 cm und erreicht dabei eine Genauigkeit von etwa 54 %. Der YOLOv4 Ansatz detektiert zum ersten Mal ein Objekt

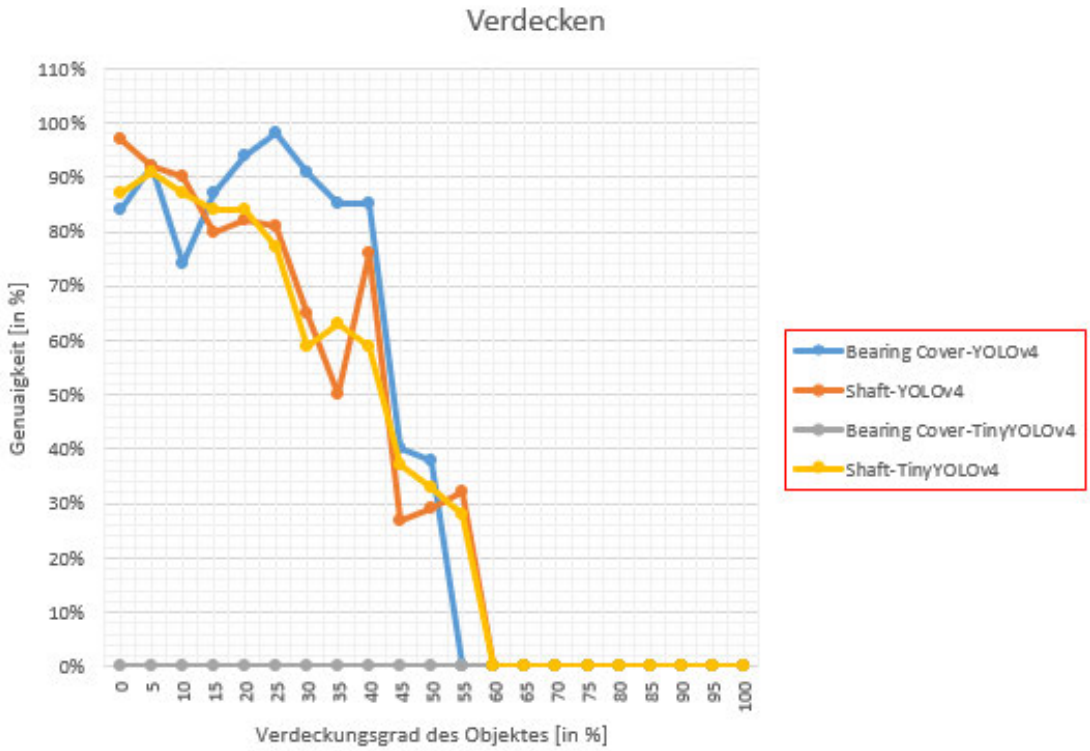
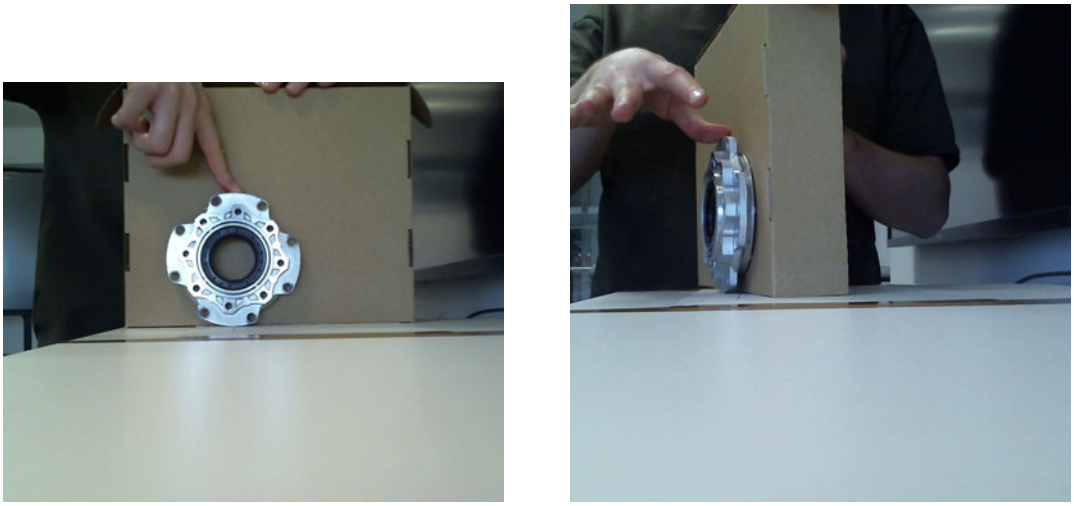


Abbildung 5.10: Verdecktes Objekt

bei einer Distanz von 60 cm und erreicht dabei eine Genauigkeit von nahezu 100 %. Zwischen den zu detektierenden Bauteilen ist bei der Verwendung des YOLOv4 Ansatzes ein minimaler Unterschied zu erkennen. Jedoch werden beide Objekte das erste Mal bei einer Distanz von 60 cm detektiert. Beim Tiny-YOLOv4 Ansatz ist im Vergleich ein deutlicherer Unterschied zu erkennen. Die Getriebewelle und die Kugellaberabdeckung werden mit einer Differenz von 20 cm zum ersten Mal detektiert. Des Weiteren ist auffällig, das beim YOLOv4 Ansatz die Genauigkeit beim Detektieren der Objekte nach der ersten Detektion stetig ansteigen, bis zu einem Wert von über 90 %. Anschließend werden beide Objekte auf einem konstant hohen Niveau, über 90 %, durch den YOLOv4 Ansatz detektiert. Mit zunehmender Distanz stabilisiert sich die Genauigkeit bis zu der maximalen Distanz von 280 cm und erreicht einen Wert von 100 %. Bei dem Tiny-YOLOv4 Ansatz ist ebenfalls ein Anstieg in der Genauigkeit erkennbar, jedoch mit deutlichen Schwankungen bei der Genauigkeit. Mit zunehmender Distanz erreicht jedoch auch der Tiny-YOLOv4 Ansatz ein hohes Niveau bei der Genauigkeit.
Bei der Betrachtung des Trainingsdatensatzes fällt auf, dass ein überwiegender Anteil der



(a) (b)

Abbildung 5.11: Winkel(a) Winkel 0 Grad, (b) Winkel 75 Grad

dargestellten Objekte mit einer größeren Distanz zwischen der Kamera und dem Objekt dargestellt sind. Diese Beobachtung spiegelt sich auch in dem Graph wieder. Während nahe Objekte erst spät, bzw. schlecht erkannt werden, werden von beiden Ansätzen Objekte mit einer größeren Distanz und mit einer höheren Genauigkeit detektiert. Des Weiteren ist bisher nicht bekannt, dass der YOLOv4 Ansatz Schwierigkeiten bei der Detektion von nahen Objekten besitzt. Viel mehr gleicht der YOLOv4 Ansatz durch das Zusammenfügen unterschiedlicher, erzeugter Featuremaps desselben Bildes die Problematik aus, deutlich kleinere Objekte zu detektieren, was vor allem im ersten YOLO Ansatz zu eine schlechteren Detektion führt.
Eine mögliche Ursache für die Differenz zwischen den Ergebnissen der Ansätze ist die unterschiedliche Anzahl an Schichten zum Extrahieren von Eigenschaften der Objekte. Der Tiny YOLOv4 besitzt weniger Schichten zum Extrahieren der Merkmale, wodurch eine schlechtere Differenzierung durch den Ansatz zwischen den Objekten möglich ist. Dies führt zu einer schlechteren Einschätzung der Ergebnisse, wodurch Tiny-YOLOv4 Ansatz nicht in der Lage das Objekt richtig zu detektieren und zu klassifizieren. Erst mit zunehmender Distanz sind die Parameter besser auf die Voraussetzungen eingestellt, wodurch auch der Tiny-YOLOv4 Ansatz in der Lage ist die Objekte zu detektieren und zu klassifizieren. Auch dies ist auf die Positionierung der Objekte im Trainingsdatensatz zurückzuführen.
Durch das Verdecken von Objekten wird untersucht, inwiefern die Objektdetektion in der Lage ist, auch Objekte zu erkennen, die verdeckt sind, durch andere Objekte oder

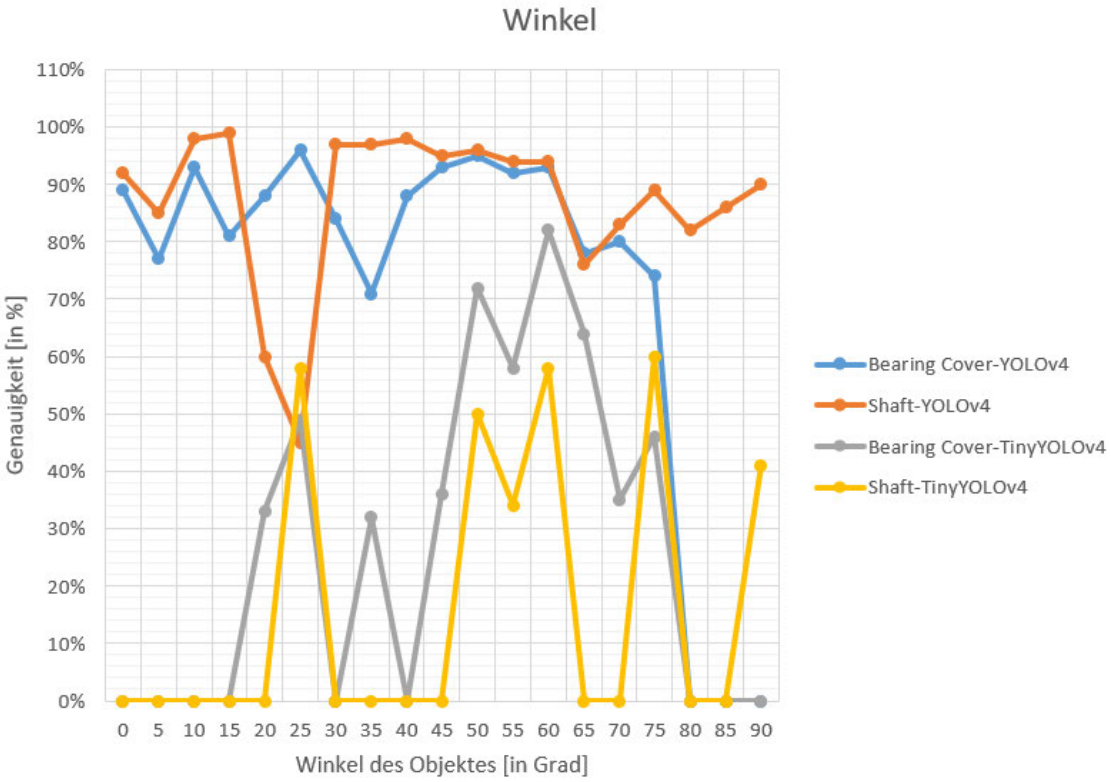
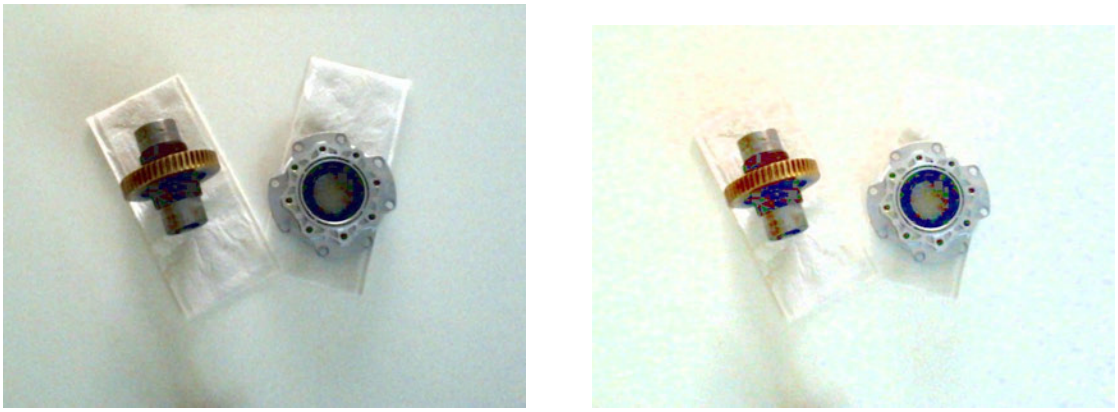


Abbildung 5.12: YOLOv4 Winkel

Objekte die nicht vollständig in dem Bild abgebildet werden. Untersucht werden die Grenzen der Objektdetektion mit Hilfe eines Testaufbaus, bei dem die Kamera über dem zu detektierenden Objekt positioniert wird. Es wird pro Testreihe nur ein Getriebebauteil genutzt. Alle sonstigen Objekte werden von dem Tisch entfernt. Das zu detektierende Objekt wird auf einem einfarbigen Tisch platziert. Zum Verdecken des Objektes wird ein weißes Blatt Papier genutzt. Der Testaufbau ist beispielhaft mit Hilfe der Kugellagerabdeckung in der Abbildung 5.9 dargestellt. Das Objekt und die Kamera werden während der Aufnahme der Testreihe nicht bewegt. Das Blatt Papier wird in Schritten von fünf Prozent, der Gesamtgröße des Objektes, über das zu detektierende Objekt gelegt. Bei jedem Schritt wird ein 2D-Bild aufgenommen. Die Bilder werden sowohl für die Evaluierung des Tiny-YOLOv4 Ansatzes, wie auch für die Evaluierung des YOLOv4 Ansatzes genutzt. Der Test wird für die Getriebewelle, wie auch für Kugellagerabdeckung durchgeführt.
Die Ergebnisse sind in einem Graphen in der 5.8 dargestellt. Auf der Ordinate ist die Genauigkeit in Prozent aufgetragen. Auf der Abzisse ist aufgetragen, um wie viel Pro-

zent das Objekt verdeckt ist, beginnend mit 0 %. Bei einem Wert von 0 % ist das Objekt nicht verdeckt. Insgesamt sind vier Graphen eingezeichnet. Unterteilt sind die Graphen in Tiny-YOLOv4 und YOLOv4, sowie im Verdeckungsgrad der Getriebewelle und der Kugellagerabdeckung.
Dem Tiny YOLOv4 Ansatz ist es nicht möglich die Kugellagerabeckung zu detektieren, auch wenn die Kugellagerabdeckung vollständig aufgedeckt ist. Im Vergleich detektiert der YOLOv4 Ansatz eine vollständig aufgedeckte Kugellagerabdeckung mit einer Genauigkeit von etwa 84 %. Auch bei einer zu 25 % verdeckten Kugellagerabdeckung, bei der der YOLOv4 Ansatz den höchsten Genauigkeitswert besitzt, ist es dem Tiny YOLOv4 Ansatz nicht möglich das Objekt zu detektieren. Das beschriebene Verhalten des Tiny YOLOv4 Ansatzes ist auch bei jedem anderen Prozentsatz der Verdeckung zu beobachten. Ab einer Verdeckung der Kugellagerabdeckung von 10 % steigt die Genauigkeit des YOLOv4 Ansatzes auf 76 % an. Der Maximalwert von 99 % wird bei einer Verdeckung von 25 % erreicht. Mit zunehmender Verdeckung fällt die Genauigkeit des YOLOv4 Ansatzes bei der Detektion der Kugellagerabdeckung. Der YOLOv4-Ansatz erkennt die Kugellagerabdeckung bis zu einer Verdeckung von 50 %. Dabei erreicht der YOLOv4 Ansatz einen Genauigkeitswert von 38 %. Ein anderes Ergebnis erzielt der Tiny-YOLOv4 bei der Detektion der Getriebewelle. Dabei erreicht der Tiny-YOLOv4 Ansatz bei keiner Verdeckung eine Genauigkeit von etwa 88 %. Dieser Wert ähnelt dem Ergebnis der YOLOv4 Ansatzes, der bei etwa einer Genauigkeit von 98 % liegt. Sowohl der Tiny-YOLOv4 Ansatz, wie auch der YOLOv4 Ansatz weisen ein ähnliches Verhalten bei der Detektion der Getriebewelle auf. Mit zunehmender Verdeckung des Objektes nimmt die Genauigkeit ab. Bei einer Verdeckung von 55 % ist bei beiden Ansätzen eine Detektion mit einer geringen Genauigkeit der Getriebewelle weiterhin möglich. Ab einer Verdeckung von 60 % ist es sowohl dem Tiny-YOLOv4 Ansatz, wie auch dem YOLOv4 Ansatz nicht mehr möglich die Getriebewelle zu erkennen.
Eine mögliche Ursache für das nicht Detektieren der Kugellagerabdeckung durch den Tiny-YOLOv4 Ansatz ist der zu geringe Abstand zwischen der Kamera und dem Objekt. Die durchgeführten Tests wurden auf Grund des zeitlichen Mangels nacheinander durchgeführt, ohne die Ergebnisse der übrigen Tests auszuwerten. Somit wurde der im Distanztest ermittelte Mindestabstand für den Tiny-YOLOv4 Ansatz für eine erfolgreiche Detektion einer Kugellagerabdeckung beim Verdeckungstest nicht berücksichtigt. Eine Wiederholung des Tests ist auf Grund der zu geringen, verbleibenden Zeit nicht umsetzbar. Dennoch ist aus den Ergebnissen erkennbar, dass unter der Berücksichtigung des minimalen Abstandes ein ähnliches Verhalten zwischen dem Tiny-YOLOv4 Ansatz und dem YOLOv4 Ansatz erwartet wird. Bestätigt wird dies durch ein ähnliches Ver-



(a) (b)

Abbildung 5.13: Kugellagerabdeckung vierter Datensatz (a) Helligkeitsstufe 7, (b) Helligkeitsstufe 12

halten bei der Detektion der Getriebewelle. Sowohl der Tiny-YOLOv4 Ansatz, wie auch der YOLOv4 Ansatz detektieren die Getriebewelle mit einer geringeren Genauigkeit, mit zunehmender Verdeckung des Objektes. Dem Tiny-YOLOv4 Ansatz und dem YOLOv4 Ansatz ist es möglich Objekte zu detektieren, die nur zu 50 % sichtbar sind. Eine weitere Grenze der Objektdetektion, die untersucht wird, ist bis zu welchem Winkel die Objektdetektion in der Lage ist Objekte zu detektieren und zu klassifizieren.
Für die Untersuchung des Winkels werden die Getriebebauteile einzeln vor der Kamera positioniert. Die Kamera wird auf einem Tisch fixiert. Die zu detektierenden Objekte werden in einem Abstand von 60 cm vor der Kamera positioniert. Der Aufbau des Versuches ist in der Abbildung 5.11 an Hand der Kugellagerabdeckung dargestellt. Für die Einstellung des Winkels wird eine Box verwendet. Die Box wird ebenfalls als Bezugsebene für die Kugellagerabdeckung genutzt. Auf diese Weise kann der Winkel nur um die vertikale Achse verändert werden. Die Winkel werden in Schritten von fünf Grad verändert, bis zu einem Winkel von 90 Grad. Mit jeder Veränderung des Winkels wird ein 2D-Bild erzeugt. Die aufgenommenen Bilder werden sowohl auf den Tiny-YOLOv4 Ansatz, wie auch auf den YOLOv4 Ansatz angewandt.
Die ermittelten Ergebnissen werden in einem Koordinatensystem eingetragen. Dabei ist auf die Ordinate die Genauigkeit in Prozent aufgetragen, von 0 % bis 100 %. Auf der Abzisse ist der Winkel von 0 bis 90 Grad aufgetragen. Die graphischen Ergebnisse sind in Abbildung 5.12 dargestellt. Insgesamt sind vier Graphen eingezeichnet. Dabei wird bei zwei Graphen der Tiny-YOLOv4 Ansatz angewandet. Diese Graphen unterscheiden sich durch zwei zu detektierende Objekte, auf die Tiny-YOLOv4 Ansatz angewandt wurde.

Sowohl die Untersuchung der Kugellagerabdeckung, wie auch der Getriebewelle wird mit dem YOLOv4 Ansatz wiederholt.
Der YOLOv4 Ansatz detektiert die Kugellagerabdeckung in der Ausgangsposition mit einer Genauigkeit von 89 %. Die Genauigkeit schwankt bei der Detektion der Kugellagerabdeckung zwischen 70 % und 98 %. Dabei erreicht der YOLOv4 Ansatz bei einem Winkel von 35 Grad die niedrigste Genauigkeit einer detektieren Kugellagerabdeckung. Bei einem Winkel von 25 Grad erreicht der YOLOv4 Ansatz die höchste Genauigkeit. Ab einem Winkel von 75 Grad ist es dem YOLOv4 Ansatz nicht mehr möglich die Kugellagerabdeckung zu detektieren. In der Ausgangsposition der Getriebewelle erreicht der YOLOv4 Ansatz eine Genauigkeit von 92 %. Bei einem Winkel von 25 Grad detektiert der YOLOv4 Ansatz die Getriebewelle mit einer Genauigkeit von 44 %. Dies ist der niedrigste aufgenommene Wert. Mit der nächsten Veränderung des Winkels, auf 30 Grad, erreicht der YOLOv4 Ansatz eine Genauigkeit von 98 %. Bei einem Winkel vn 90 Grad erreicht der YOLOv4-Ansatz eine Genauigkeit von 90 %. Der YOLOv4 ist in der Lage unter jedem der gemessenen Winkel die Getriebewelle zu detektieren. Die Ergebnisse des Tiny-YOLOv4 Ansatzes unterschieden sich von den Ergebnissen des YOLOv4 Ansatzes. Sowohl für die Getriebewelle, als auch für Kugellagerabdeckung ist eine Schwankung in der Genauigkeit vorhanden. Der Tiny-YOLOv4 Ansatz ist nicht in der Lage die Kugellagerabdeckung in der Ausgangsposition zu detektieren. Erst bei einem Winkel von 20 Grad detektiert der Tiny-YOLOv4 Ansatz die Kugellagerabdeckung mit einer Genauigkeit von 33 %. Ein vergleichbares Verhalten ist bei der Detektion der Getriebewelle durch den Tiny-YOLOv4 Ansatz vorhanden. Erst bei einem Winkel 25 Grad detektiert der Tiny-YOLOv4 Ansatz die Getriebewelle mit einer Genauigkeit von 58 %. Starke Schwankungen bei der Detektion beider Objekte folgen. Bei Winkeln von 30 bis 45, 65 und 70 und bei einem Winkel von 80 und 85 Grad ist es dem Tiny-YOLOv4 Ansatz nicht möglich die Getriebewelle zu detektieren. Bei einem Winkel von 90 Grad besitzt der Tiny-YOLOv4 Ansatz eine Genauigkeit bei der Detektion der Getriebewelle von 42 %. Die Kugellagerabdeckung wird ebenfalls nicht zuverlässig durch den Tiny-YOLOv4 Ansatz detektiert. Bei einem Winkel von 30 und 40 Grad, sowie bei einem Winkel zwischen 80 und 90 Grad ist es dem Tiny-YOLOv4 Ansatz nicht möglich die Kugellagerabdeckung zu detektieren. Die höchste gemessen Genauigkeit liegt bei einem Winkel von 60 Grad mit einer Wert von 82 %.
Der YOLOv4 Ansatz ist in der Lage die Getriebebauteile bis zu einem Winkel von 75 Grad zu detektieren. Ab einem Winkel von 75 Grad wird die Kugellagerabdeckung nicht mehr durch den YOLOv4 Ansatz detektiert. Eine mögliche Ursache für dieses Verhalten sind nicht ausreichende Beispiele im Trainingsdatensatz. Der YOLOv4 Ansatz ist nicht

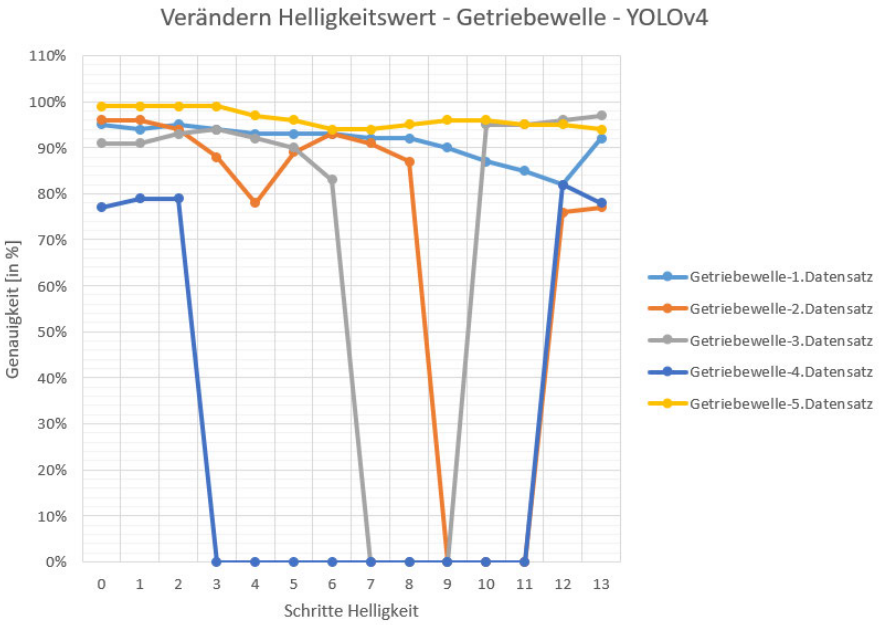


Abbildung 5.14: Tiny YOLOv4 Helligkeitsgraph Getriebewelle

in der Lage die Parameter für die Kugellagerabdeckung einzustellen, wodurch Merkmale des Objektes ab einem Winkel von 75 Grad durch den Ansatz extrahiert werden. Der YOLOv4 Ansatz ist nicht in der Lage die Kugellagerabdeckung eindeutig zuzuordnen. Bei dem Distanztest ist der Datensatz unter einer Ausrichtung der Getriebebauteile erstellt worden. Eine mögliche Ursache für den schwankenden Verlauf der Graphen beim Tiny-YOLOv4 Ansatz könnte sein, dass der Ansatz bei unterschiedlichen Winkeln, eine unterschiedliche Distanz zwischen Kamera und Objekt benötigt, um die Getriebebauteile zu detektieren. Diese Vermutung wird bestärkt durch die Detektion der Getriebebauteile in der Ausgangsposition bei einem Winkel von 0 Grad. Dabei sind beide Bauteile in derselben Position die auch beim Distanztest verwendet wird. Auf Grund der Distanz zwischen Kamera und Objekt von 60 cm ist es dem Tiny-YOLOv4 Ansatz nicht möglich die Getriebebauteile zu detektieren. Des Weiteren ist die unterschiedliche Anzahl an verwendeten Schichten auch in diesem Test erkennbar. Der Tiny-YOLOv4 Ansatz detektiert die Getriebebauteile mit einer deutlich geringeren Differenz, als der YOLOv4 Ansatz. Dem Tiny-YOLOv4 Ansatz ist nicht möglich die Merkmale für die Getriebebauteile zu extrahieren, sodass eine eindeutige Zuordnung der Klasse durch den Tiny-YOLOv4 Ansatz erfolgt.

Ein weiterer Einfluss der bei der Nutzung Mini-Factory aufrtritt sind unterschiedli-

che Lichtverhältnisse. Die unterschiedlichen Lichtverhältnisse können durch die Deckenbeleuchtung, durch unterschiedliche Wetterverhältnisse, durch die Tageszeit oder auch durch die Nutzung weiterer Lichtquellen verändert werden. Dies kann einen Einfluss auf die Objektdetektion besitzen. Ein möglicher Einfluss auf die Objektdetektion durch sich verändernde Lichtverhältnisse wird durch den folgenden Test untersucht. Dabei werden fünf Bilder der Getriebebauteile aufgenommen. Die Getriebebauteile besitzen unterschiedliche Positionen und Rotationen, die zufällig gewählt sind. Des Weiteren werden die Bilder bei unterschiedlichen Lichtverhältnissen aufgenommen. Dabei sind die unterschiedlichen Lichtverhältnisse durch das Deckenlicht im Büro von Dassault Systèmes am ZAL, wie auch durch die unterschiedlichen Tageslichtverhältnisse definiert. Zudem werden die Getriebebauteile auch unter einem Tisch positioniert, um einen weiteren Einflussfaktor bei den Lichtverhältnissen zu erreichen. Die Szenarien werden als RGB-Bilder aufgenommen. Eine anschließende Transformation der RGB-Bilder in den HSV-Farbraum ermöglicht es, den Helligkeitswert der Bilder nachträglich zu ändern. Die Anpassung des Helligkeitswertes erfolgt schrittweise, wobei der Wert pro Schritt um eins geändert wird. Dadurch ist es möglich unterschiedliche Lichtverhältnisse vom selben Bild zu erzeugen. Nach der Veränderung des Helligkeitswertes wird das Bild in den RGB-Farbraum transformiert. Die Bilder werden nicht unter Laborbedingungen aufgenommen. Des Weiteren sind keine Messmittel zur Bestimmung der aktuellen Lichtverhältntisse im Büro vorhanden. Der Helligkeitswert wird in 14 Schritten angepasst. Bis zu diesem Wert sind bei allen fünf Bildern die Pixelwerte überwiegend nicht in der Sättigung, wodurch die Pixel keine zufälligen Werte annehmen, die zu einer Verfälschung des Bildes führen. Unter dem Begriff der Sättigung ist ein Pixelwert zu verstehen, der gleich dem Wert 0 oder 255 entspricht. Mit dem Überschreiten des Sättigungswertes kann dies zu einem zufälligen Pixelwert führen. Der Testaufbau ist in der Abbildung 5.13 dargestellt. Zu erkennen sind sowohl die Getriebewelle, als auch die Kugellagerabdeckung, die auf einen weißen Hintergrund gelegt sind. Dabei ist zwischen den Bildern ein unterschiedlicher Helligkeitswert eingestellt. Sowohl der Tiny-YOLOv4 Ansatz, wie auch der YOLOv4 Ansatz werden auf die erzeugten Datensätze angewandt.
Die Ergebnisse des Helligkeitstests werden in vier Koordiantensysteme eingezeichnet. Zwei der Koordinatensysteme werden für den Tiny-YOLOv4 Ansatz verwendet, die Anderen zwei Koordinatensysteme werden für den YOLOv4 Ansatz genutzt. Des Weiteren wird zwischen der Detektion der Getriebewelle, als auch der Detektion der Kugellagerabdeckung unterschieden. Die Ergebnisse für die Detektion der Getriebewelle durch den YOLOv4 Ansatz sind in der Abbildung 5.14 dargestellt. Den Einfluss der Helligkeit auf die Detektion der Kugellagerabdeckung durch den YOLOv4 Ansatz sind in der Abbil-

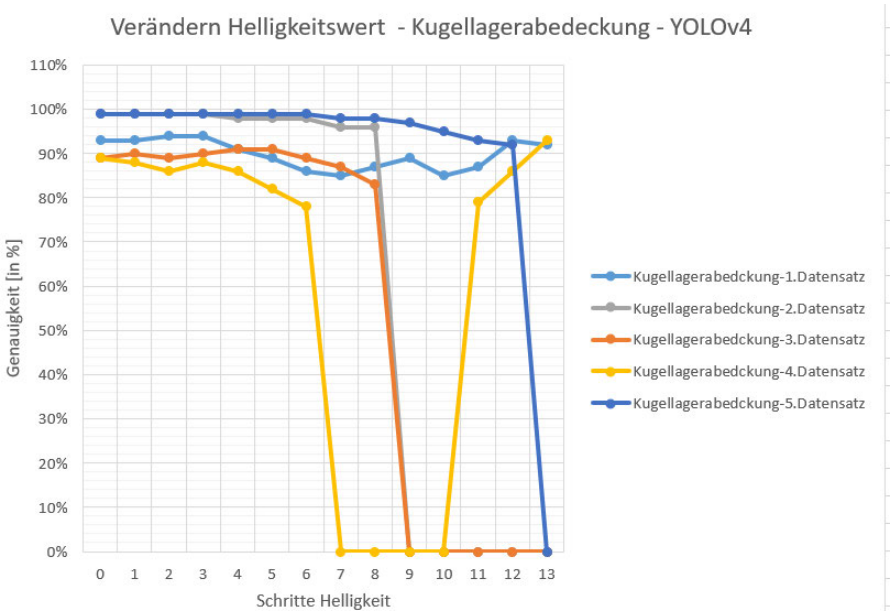


Abbildung 5.15: YOLOv4 Helligkeitsgraph Kugellagerabdeckung

dung 5.15 dargestellt. In der Abbildung 5.16 ist die Untersuchung des Einflusses der unterschiedlichen Helligkeiten auf die Getriebewelle durch den Tiny-YOLOv4 Ansatz dargestellt. Das Koordinatensystem, in das die Ergebnisse des Helligkeitstests bei der Detektion der Kugellagerabdeckung durch den Tiny-YOLOv4 Ansatz aufgetragen sind, sind in der Abbildung 5.17 dargestellt. Alle dargestellten Koordinatensysteme besitzen dieselbe Skalierung und Beschriftung. Auf die Ordinate der Koordinatensysteme ist die Genauigkeit in Prozent aufgetragen, von 0 % bis 100 %. Die Abzisse stellt die vorgestellte Schrittweite zur Einstellung des Helligkeitswertes im HSV-Farbraum dar. In jedes Koordiantensystem sind fünf Graphen eingezeichnet, wobei jeder Graph ein Bild repräsentiert, dessen Helligkeitswert verändert wird.

In der Abbildung 5.14 wird der Einfluss des Helligkeitswertes durch den YOLOv4 Ansatz bei der Detektion der Getriebewelle dargestellt. Der YOLOv4 Ansatz detektiert die Getriebewelle in allen Bildern bei dem Ausgangshelligkeitswert von 1. Dabei ähneln sich die ermittelten Genauigkeitswerte, außer die Genauigkeit des vierten Datensatzes. Der YOLOv4 Ansatz detektiert das Ausgangsbild des vierten Datensatzes mit einer Genauigkeit von 77 %. Die Ausgangsbilder der übrigen Datensätze werden mit einer Genauigkeit zwischen 91 % und 99 % detektiert. Bei einer Helligkeitsstufe von 13 werden alle Getriebewellen durch den YOLOv4 Ansatz detektiert. Zwischen einer Helligkeitsstufe von 9 und 11 wird die Getriebewelle des zweiten Datensatzes nicht detektiert. Die Getriebe-

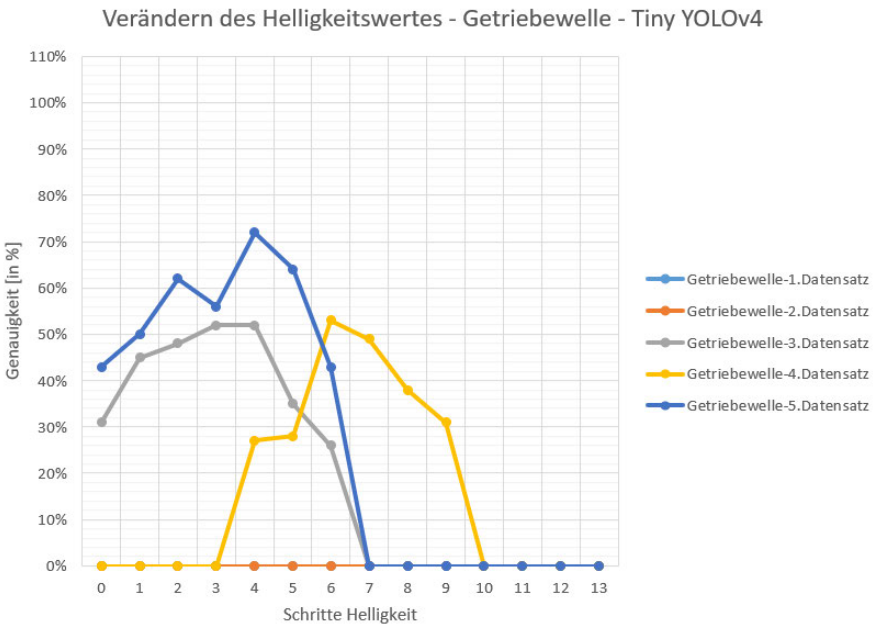


Abbildung 5.16: Tiny YOLOv4 Helligkeitsgraph Getriebewelle

welle des dritten Datensatzes werden bei den Helligkeitsstuen zwischen 7 und 9 nicht detektiert. Die Getriebewelle wird bei den Bilder des vierten Datensatzes zwischen einer Helligkeitsstufe von 3 und 11 nicht detektiert. Bei den übrigen Helligkeitsstufen werden die Getriebewelle aller Datensätze durch den YOLOv4 Ansatz detektiert. Mit zunehmender Helligkeitsstufe nimmt die Genauigkeit der Detektion der Getriebewelle beim ersten Datensatz ab. Bei einer Helligkeitsstufe von 13 steigt die Genauigkeit an. Die Getriebewelle wird mit einer Genauigkeit von 91 % detektiert. Die Getriebewelle des fünften Datensatzes wird bei jeder Helligkeitsstufe mit einer Genauigkeit zwischen 94 % und 99 % detektiert.

Die Abbildung 5.15 repräsentiert das Ergebnis der Detektion der Kugellagerabdeckung durch den YOLOv4 Ansatz bei einem sich verändernden Helligkeitswert. Die durch den YOLOv4 Ansatz ermittelten Genauigkeiten für die Bilder mit einer Helligkeitsstufe von 0 liegen zwischen 89 % und 99 %. Bei einem Helligkeitswert von 13 werden die Kugellagerabdeckungen des ersten und des vierten Datensatzes mit einer Genauigkeit von über 90 % detektiert. Die Kugellagerabdeckungen des zweiten, des dritten und des fünften Datensatzes werden durch den YOLOv4 Ansatz werden bei einem Helligkeitswert von 13 nicht detektiert. Die Kugellagerabdeckung des ersten Datensatzes wird mit einer Genauigkeit zwischen 85 % und 94 % bei jedem Helligkeitswert detektiert. Bis zu einem

Helligkeitswert von 8 besitzt der YOLOv4 Ansatz bei der Detektion der Kugellagerabdeckung des zweiten Datensatzes eine Genauigkeit zwischen 96 % und 99 %. Ab einem Helligkeitswert von 9 wird die Kugellagerabdeckung des zweiten Datensatzes nicht weiter detektiert. Ein ähnliches Verhalten wie beim zweiten Datensatz ist auch beim Verlauf des dritten Datensatzes erkennbar. Der YOLOv4 Ansatz detektiert die Kugellagerabdeckung bei einem Helligkeitswert zwischen 7 und 10 nicht. Die Kugellagerabdeckung des fünften Datensatzes wird bis zu einem Helligkeitswert von 12 mit einer Genauigkeit zwischen 95 % und 99 % detektiert. Ab einem Helligkeitswert von 13 wird die Kugellagerabdeckung im fünften Datensatz nicht mehr detektiert.
In Abbildung 5.16 ist das Ergebnis des Helligkeitstests bei Anwendung des Tiny-YOLOv4 Ansatz auf die Getriebewellen-Datensätze dargestellt. Der Tiny-YOLOv4 Ansatz detektiert bei keinem Helligkeitswert die Getriebewelle des ersten und des zweiten Datensatzes. Die Getriebewelle des dritten und des fünften Datensatzes werden durch den Tiny-YOLOv4 Ansatz bei einem
Helligkeitswert von 0 detektiert. Bei beiden Datensätzen steigen die Genauigkeitswerte bis zu einem Maximum an. Das Maximum wird bei einem Helligkeistwert von 4 erreicht. Anschließend fallen die Genauigkeitswerte, bis Tiny-YOLOv4 die Getriebewelle ab einem Helligkeitswert von 7 für den dritten und den fünften Datensatz nicht mehr detektiert. Die Getriebewelle des vierten Datensatzes wird durch den Tiny-YOLOv4 Ansatz erst ab einer Helligkeit von 3 detektiert und steigt anschließend an. Bei einem Helligkeitswert von 6 erreicht der Tiny-YOLOv4 Ansatz das Maximum in der Genauigkeit, mit einem Wert von 54 % und fällt anschließend ab. Tiny-YOLOv4 detektiert die Getriebewelle des vierten Datensatzes ab einem Helligkeitswert von 10 nicht mehr.
Die Abbildung 5.17 stellt das Koordinatensystem dar, in welches das Ergebnis des Helligkeitstests bei Anwendung des Tiny-YOLOv4 Ansatzes auf die Datensätze der Kugellagerabdeckung angewandt wird. Der erste, der zweite und der dritte Datensatz der Kugellagerabdeckung werden bei keinem Helligkeitswert durch den Tiny-YOLOv4 Ansatz detektiert. Bei dem vierten Datensatz erreicht der Tiny-YOLOv4 Ansatz bei einem Helligkeitswert von 0 eine Genauigkeit von 80 %. Mit zunehmenden Helligkeitswerten nimmt die Genauigkeit des Tiny-YOLOv4 Ansatzes ab. Bei einem Helligkeitswert von 8 wird die Kugellagerabdeckung des vierten Datensatzes nicht mehr detektiert. Mit einer Genauigkeit von 26 % detektiert der Tiny-YOLOv4 Ansatz die Kugellagerabdeckung des vierten Datensatzes bei einem Helligkeitswert von 9. Eine Detektion mit zunehmenden Helligkeitswert wird nicht gemessen. Die Kugellagerabdeckung des fünften Datensatzes wird bei einem Helligkeitswert von 0 mit einer Genauigkeit von 58 % gemessen. Bei einem Helligkeitswert von 7 und einer Genauigkeit von 78 % wird die Kugellagerabdeckung

durch der Tiny-YOLOv4 Ansatz detektiert. Anschließend fällt die Genauigkeit auf 0 %, der Tiny-YOLOv4 Ansatz kann die Kugellagerabdeckung des fünften Datensatzes nicht mehr detektieren.
Der Einfluss unterschiedlicher Helligkeitsstufen ist in den Ergebnissen des Tiny-YOLOv4 Ansatzes, wie auch des YOLOv4-Ansatzes erkennbar. Das Ausgangsbild jedes Datensatzes wird durch den YOLOv4 Ansatz detektiert. Auffällig ist vor allem das Ergebnis des vierten Datensatzes, der zwischen den Helligkeitsstufen zwischen 3 und 11 nicht detektiert wird. Die Abbildung 5.13 zeigt die Getriebewelle und die Kugellagerabdeckung bei einer Helligkeitsstufe von 7 und bei einer Helligkeitsstufe von 12 des vierten Datensatzes. Im Vergleich zu den Bildern der anderen Datensätze, sind die Getriebebauteile auf einem neutralen Hintergrund positioniert. Der mit Hilfe der Creative Experience erzeugte Trainingsdatensatz beinhaltet Bilder, bei denen 360 Grad Welten genutzt werden. Keine der genutzten 360 Grad Bilder repräsentiert eine neutrale Fläche. Die Vermutung liegt nah, dass der Hintergrund eine Ursache für das nicht Detektieren der Getriebebauteile ist.
Diesem Ansatz widerspricht jedoch Tremblay, der durch das Nutzen von unterschiedlichen Hintergründen und das Hinzufügen von zusätzlichen Geometrien ein abstraktes Bild geschaffen hat. Dies hilft dem Netzwerk dabei, die Merkmale unabhängig von Hintergrund zu extrahieren. Dass das Extrahieren der Merkmale auch bei der in dieser Masterarbeit vorgestellten Objektdetektion erfolgreich ist, wird zum Einen durch die anderen Datensätze besätigt, zum Anderen auch durch den vierten Datensatz. Mit einer Helligkeitsstufe von 10, bzw. 12 werden die Getriebebauteile des vierten Datensatzes durch YOLOv4 Ansatz mit einer Genauigkeit von mehr als 75 % detektiert. Da bei den anderen Datensätzen auch bei einer hohen Helligkeitsstufe die Getriebebauteile detektiert werden, kann ein zu helles Bild nicht als eine pauschale Ursache für das nicht Detektieren durch den YOLOv4 Ansatz betrachtet werden. Mit einer höheren Varianz beim Erzeugen des Trainingsdatensatzes, bezogen auf die Helligkeit, könnten beim YOLOv4 Ansatz die Ausreißer vermieden werden. Dadurch ist eine zuverlässigere Objektdetektion bei sich verändernden Lichtverhältnissen vorstellbar.
Zum Vergleich schneidet das Ergebnis des Tiny-YOLOv4 Ansatz deutlich schlechter ab. Die Getriebebauteile werden nicht bei allen Ausgangsbilder detektiert. Dies lässt sich auf die Testergebnisse der bereits vorgestellten Tests zurückführen. Die Distanz, wie auch der Winkel könnten einen Einfluss auf die Detektion durch den Tiny-YOLOv4 Ansatz besitzen. Allerdings erkennt man, bei den durch den Tiny-YOLOv4 detektierten Getriebbauteilen, dass ein Einfluss durch den Helligkeitswert entsteht. Bei einem zu hohen Helligkeitswert ist es dem Tiny-YOLOv4 Ansatz nicht mehr möglich die Getriebebauteile jedes Datensatzes zu detektieren. Auch bei den Ergebnissen des Tiny-YOLOv4 Ansat-

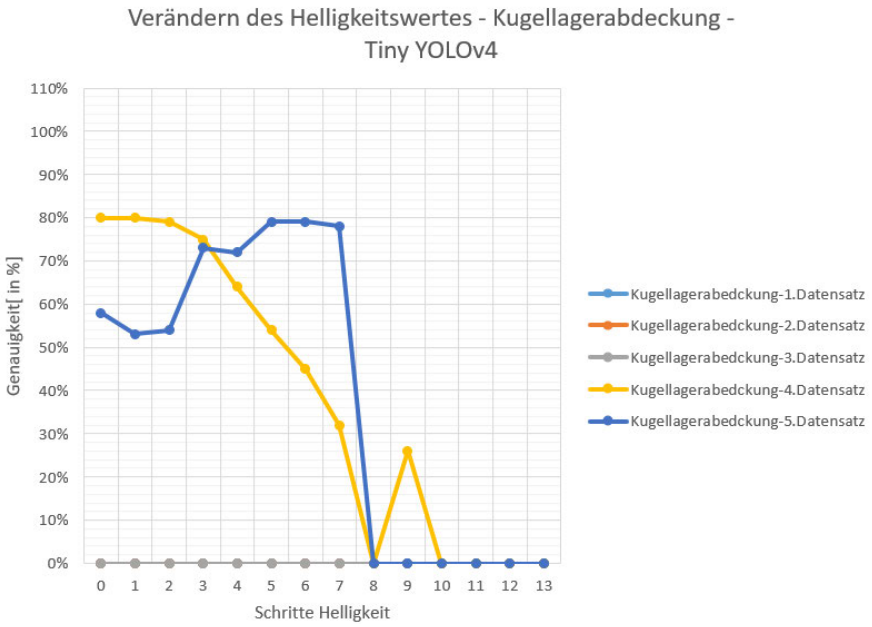


Abbildung 5.17: Tiny YOLOv4 Helligkeitsgraph Kugellagerabdeckung

zes fällt auf, dass das Ergebnis des vierten Datensatzes von den anderen Datensätzen abweicht. In der Abbildung 5.16 ist das Ergebnis der Getriebewelle dargestellt. Die Getriebewelle wird erst ab einem Helligkeitswert von 4 durch den Tiny-YOLOv4 Ansatz detektiert. Dies spricht dafür, dass die Getriebewelle zu Beginn des Tests zu dunkel für die Objektdetektion ist. Ab einem Helligkeitswert von 10 wird die Getriebewelle nicht mehr detektiert. Ab diesem Wert scheint die Getriebewelle zu hell für die Detektion dargestellt zu werden.

Die in diesem Kapitel vorgestellten und durchgeführten Tests haben gezeigt, dass eine Differenz zwischen dem Tiny-YOLOv4 Ansatz und dem YOLOv4 Ansatz besteht. Vor allem bei der Ermittlung des mAP-Wertes ist die Differenz aufgefallen, dass der YOLOv4-Ansatz besser für den Einsatz bei der Objektdetektion geeignet ist. Zwischen dem Datensatz, der auf 3D-Objekten basiert und dem realen Datensatz besteht zwar eine Differenz, jedoch beträgt diese nur etwa 15 %. Beim Tiny-YOLOv4 Ansatz beträgt diese Differenz einen Wert von über 30 %. Auch bei den anderen durchgeführten Tests ist erkennbar, dass die Sim-to-Real Gap nicht gut durch den Tiny-YOLOv4 Ansatz geschlossen wird. Die Ergebnisse zeigen, dass die Getriebebauteile nicht zuverlässig durch den Tiny-YOLOv4 Ansatz detektiert werden. Eine Anpassung bei der Erstellung des Trainingsdatensatzes, unter der Berücksichtigung der gewonnen Erkenntnisse könnte zu einem besseren Ergebnis beim Tiny-YOLOv4 Ansatz führen. Dennoch ist es unwahr-

scheinlich, dass das Ergebnis des Tiny-YOLOv4 Ansatzes besser, als das Ergebnis des YOLOv4 Ansatzes abschneidet. Die höhere Schnelligkeit bei der Ausführung des Tiny-YOLOv4 Ansatzes auf einem SBC gleicht die deutlich schlechteren Ergebnisse nicht aus. Mit Anpassungen der Trainingsdaten unter der Berücksichtungen der Ergebnisse ist das Verringern der Sim-to-Real Gap mit Hilfe des YOLOv4 Ansatzes vielversprechender. Tremblay erreicht mit dem Ansatz und unter Verwendung eines deutlichen kleineren realen Datensatzes für das Training einen mAP von 98%. Dies bestätigt die Annahme, dass das Schließen der Sim-to-Real Gap möglich ist. Für die Objektdetektion der Mini-Factory ist daher der YOLOv4-Ansatz der vielversprechendste Ansatz.

6 Fazit und Ausblick

Das Ziel dieser Masterarbeit ist es eine Objektdetektion für die mobile Fertigungsstraße, die Mini-Factory zu entwickeln. Als ersten Use-Case werden Bauteile eines Getriebes detektiert. Für eine mögliche Umsetzung werden verschiedene Ansätze betrachtet. Eine Bewertung der betrachteten Ansätze erfolgt durch die erzielten Ergebnisse bei Wettbewerben, die speziell für den Vergleich der Ansätze durchgeführt werden. Das AlexNet erreichte beim ImageNet den ersten Platz und erzielte ein deutlich besseres Ergebnis als der Zweitplatzierte. Dies führte zu einer Ausrichtung der Forschung auf die Entwicklung von neuen CNN-Architekturen. Dabei werden vor allem die Architekturen Faster-RCNN und YOLO in einer Vielzahl von Arbeiten eingesetzt. Die genannten CNN-Architekturen verfolgen zwei unterschiedliche Ansätze zum Detektieren von Objekten. Der Faster-RCNN soll durch seinen zweistufigen Ansatz vergleichsweise langsam sein, jedoch werden die Objekte mit einer hohen Genauigkeit detektiert. Der YOLO-Ansatz ist ein einstufiger Ansatz, der die Objekte mit einer vergleichsweise niedrigen Genauigkeit detektieren soll, dafür aber eine hohe Geschwindigkeit besitzt. Es werden beide Architekturen für die Objektdetektion in Betracht gezogen. Des Weiteren wird ein für leistungsschwache Systeme angepasster YOLO-Ansatz betrachtet.
Ein CNN ist eine Methode die dem Deep-Learning zugeordnet wird. Für die Nutzung des Deep-Learnings ist ein Datensatz erforderlich, der sich aus einem Trainingsdatensatz und einem Testdatensatz zusammensetzt. Die Größe des Datensatzes ist abhängig von der Menge der zu detektierenden Objekte. In dieser Masterarbeit werden zwei Bauteile des Getriebes genutzt, die für eine Überprüfung der grundsätzlichen Machbarkeit der gewählten Ansätze dienen. Des Weiteren wird ein bereits trainiertes Netzwerk genutzt, dessen Parameter an die zu detektierenden Objekte durch den selbsterstellten Datensatz angepasst werden. Dadurch verringert sich die Größe des selbsterstellten Datensatzes. Als Messwert wird der mean Average Precision Wert genutzt, um die Ergebnisse miteinander zu vergleichen. Der zu erzeugende Datensatz basiert auf 3D-Modellen. Zum Erzeugen des Datensatzes werden die verfügbare Software Blender und die 3DExcite-Creative Experience App miteinander verglichen. Die 3DExcite-CreativeExperience App wird auf Grund

der Integration in die 3DExperience-Plattform und dem damit verbundenen größeren Potential einer Weiterverarbeitung der Daten gewählt. Bei der Nutzung eines Datensatzes basierend auf 3D-Modellen ist die Sim-to-Real Gap ein Faktor, der für die Objektdetektion der Mini-Factory berücksichtigt wird. Für die Integeration der Objektdetektion auf der Mini-Factory wird ein SBC genutzt. Die Auswahl des NVIDIA Jetson Nano erfolgt nach dem Vergleich mit anderen SBCs. Die Evaluierung der trainierten CNN-Ansätze ergibt, dass der Faster-RCNN auf Grund der Geschwindigkeit und der Genauigkeit nicht für die Objektdetektion geeignet ist, weswegen der Faster-RCNN Ansatz nicht für die Implementierung der Objektdetektion genutzt wird.
Der Tiny-YOLOv4 erreicht bei der Evaluierung mit dem Datensatz, der auf 3D-Modellen basiert, einen mAP Wert von 82,3 %. Der YOLOv4 Ansatz erreicht einen Wert von 89,7 %. Daraus folgt ein Untersuchung der verbliebenen Ansätze Tiny-YOLOv4 und YOLOv4. Insgesamt fünf unterschiedliche Testszenarien mit realen Daten ermöglichen eine bessere Einschätzung des trainierten Tiny-YOLOv4 Ansatzes und des YOLOv4 Ansatzes. Dabei zeigen sich Defizite beider Ansätze, verglichen mit den vorgestellten Tests an Daten, die auf 3D-Modellen basieren. Der Tiny-YOLOv4 Ansatz erreicht bei der Evaluierung mit dem realen Datensatz einen mAP-Wert von 31,3 %. Der YOLOv4 Ansatz erreicht einen Wert von 72,6 %.
Die Defizite lassen sich u. a. auf den erzeugten Datensatz zurückführen. Auffällig ist, dass nahe Objekte durch YOLO nicht detektiert werden. Die in dem Datensatz dargestellten Objekte sind überwiegend mit einer größeren Distanz zu der Kamera im Raum positioniert. Dabei sind die Position und die Rotation der Getriebebauteile zufällig gewählt. Für die Weiterentwicklung des Trainingsdatensatzes ist ein Vorgehen sinnvoll, bei dem die Objekte nicht zufällig im Raum positioniert werden. Eine strukturierte Vorgehensweise könnte es der Objektdetektion ermöglichen auch nahegelegene Objekte zu detektieren. Dies ist vor allem für eine spätere Implementierung der Greiffunktion notwendig. Auch die Berücksichtigung der strukturierten Vorgehensweise bei der Rotation und der gewählten Helligkeit könnte zu einem besseren Ergebnis führen, wodurch die Sim-to-Real Gap verringert wird. Durch Tremblay ist es bewiesen, dass eine Objektdetektion, die auf einem Trainingsdatensatz basierend auf 3D-Modellen entwickelt wird, ein mAP-Wert von 98 % erzielt und damit die Sim-to-Real Gap nahezu geschlossen wird. Eine weitere Verbesserung die für die Weiterentwicklung der Objektdetektion durchgeführt werden sollte, ist das automatisierte Erzeugen der Daten. Zum Zeitpunkt der Erstellung dieser Masterarbeit ist das Erzeugen einer Bounding Box mit der Creative Experience App möglich. Die erzeugte Bounding Box wird als kleinstmögliche Umrandung des 3D-Modells erzeugt. Eine Umrechung der Bounding Box in ein 2D-Bild ist erforderlich. Des Weiteren ermög-

licht die Creative Experience App zur Zeit keine Möglichkeit die benötigten Dateien der Datensätze aus der 3DExperience Plattform zu exportieren. Auch das Trainieren innerhalb der 3DExperience Plattform ist zur Zeit nicht möglich. Mit einer Möglichkeit die 3D-Modelle zu labeln, die benötigten Dateien automatisch zu exportieren oder das Trainieren innerhalb der 3DExperience Plattform zu ermöglichen, wäre eine Zeitersparnis möglich. Die Zeitersparnis führt zu einer Verringerung der Kosten und wäre damit eine Lösung der in der Motivation angesprochenen Fragestellung. Größere Datensätze könnten auch von Einzelpersonen oder kleinere Gruppen erzeugt werden.
Das Hinzufügen zusätzlicher Getriebebauteile zu der Objektdetektion wäre nach dem Hinzufügen der vorgestellten Erweiterung verhältnismäßig schnell möglich. Der in dieser Masterarbeit erzeugte Datensatz, basierend auf zwei Getriebebauteilen, benötigt drei Tage zum Erzeugen und weitere drei Tage zum Labeln. Mit der Automatisierung ist keine Verringerung der benötigten Zeit zum Präparieren des Datensatzes zu erwarten, jedoch entfällt das händische Labeln der Daten. Des Weiteren ist die Wahrscheinlichkeit, das Fehler durch übersehene Objekte beim Labeln entstehen, verringert. Dies würde zu einem hochwertigeren Datensatz führen, der zu einem besseren Traingsergebnis, wie auch zu einem besseren Testergebnis führen würde. Dennoch ist auch das Ergebnis und die Zeitersparnis des Datensatzes dieser Masterarbeit hervorzuheben. Der zur Evaluierung erzeugte reale Datensatz, der eine geringe Varianz aufweist und lediglich einen Umfang von 150 Bildern besitzt, benötigt fast zwei Tage zur Erstellung. Dies ist ein deutlich höherer, benötigter Zeitaufwand, als beim Datensatz der mit der Creative Experience App erzeugt wird.
Die Implementierung der Objektdetektion in die Mini-Factory erfüllt teilweise die aufgestellten Anforderungen. Bisher ist keine Möglichkeit in die Objektdetektion integriert, mit der die Reihenfolge der Getriebebauteile angezeigt wird. Die Implementierung der Objektdetektion auf einem SBC ist erfolgreich. Dadurch besteht ein geringer E influss auf die Mini-Factory durch zusätzlich angebrachte Hardware. Der Punkt der Anforderungsanalyse gilt als erfüllt. Einen Einfluss durch die industrielle Umgebung auf die Objektdetektion ist sehr gering. Des Weiteren ist eine Detektion der Getriebebauteile durch den YOLOv4 Ansatz möglich. Ein Ansatz zur Verbesserung der Objektdetektion wurde vorgestellt. Mit der Weiterentwicklung ist ein besseres Ergebnis der Objektdetektion zu erwarten. Die Position des detektierten Objektes wird in einer zusätzlichen Datei gespeichert. Auf die Datei kann durch ein anderes Programm zugegriffen werden. Der Punkt der Anforderungsanalyse gilt als erfüllt. Der Helligkeitstest hat aufgezeigt, dass ein Einfluss durch verschiedene Lichtverhältnisse besteht. Dennoch ist es der Objektdetektion möglich die Getriebebauteile auch bei einer hohen Helligkeitsstufe zu detektieren.

Ein Ansatz zu einem verbesserten Verhalten bei sich verändernden Lichtverhältnissen ist vorgestellt wurden.
Die Ergebnisse der Masterarbeit haben gezeigt, dass es möglich ist eine Objekdetektion zu entwicklen, die auf künstlichen Daten basiert. Die Sim-to-Real Gap kann nicht vollständig geschlossen werden, jedoch ist der vorgestellte Ansatz der Objedetektion in der Lage reale Objekte zu detektieren. Die Getriebebauteile können durch die Objektdetektion voneinander unterschieden werden. Die vorgestellten Ergebnisse bieten eine Grundlage für die Weiterentwicklung der Objektdetektion. Ansätze zur Verbesserung werden beschrieben.

Literaturverzeichnis

[1] : *3D PERSPECTIVES | from Dassault Systèmes*. – URL https://blogs.3ds.com/perspectives/. – Zugriffsdatum: 2020-09-16

[2] : *Convolutional Neural Networks - Basics · Machine Learning Notebook*. – URL https://mlnotebook.github.io/post/CNN1/. – Zugriffsdatum: 2020-09-09

[3] *Daten sind das neue Gold*. – URL https://www.haz.de/Nachrichten/Politik/Daten-sind-das-neue-Gold. – Zugriffsdatum: 2020-06-25. – Library Catalog: www.haz.de

[4] : *DIN 44 300: Informationsverarbeitung. Nr. 9.2.11. Beuth Verlag, Berlin (1985)*

[5] : *ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012)*. – URL http://www.image-net.org/challenges/LSVRC/2012/results.html. – Zugriffsdatum: 2020-08-27

[6] Arva, Gabor ; Fryza, Tomas: Embedded video processing on Raspberry Pi. In: *2017 27th International Conference Radioelektronika (RADIOELEKTRONIKA)*, S. 1–4

[7] Baimukashev, Daulet ; Zhilisbayev, Alikhan ; Kuzdeuov, Askat ; Oleinikov, Artemiy ; Fadeyev, Denis ; Makhataeva, Zhanat ; Varol, Huseyin A.: Deep Learning Based Object Recognition Using Physically-Realistic Synthetic Depth Scenes. 1, Nr. 3, S. 883–903. – URL https://www.mdpi.com/2504-4990/1/3/51. – Zugriffsdatum: 2020-05-08. – ISSN 2504-4990

[8] Bochkovskiy, Alexey ; Wang, Chien-Yao ; Liao, Hong-Yuan M.: YOLOv4: Optimal Speed and Accuracy of Object Detection. . – URL http://arxiv.org/abs/2004.10934. – Zugriffsdatum: 2020-05-13

[9] Dalal, N. ; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* Bd. 1, IEEE, S. 886–893. – URL http://ieeexplore.

ieee.org/document/1467360/. – Zugriffsdatum: 2020-03-31. – ISBN 978-0-7695-2372-9

[10] Danielczuk, Michael ; Xu, Jingyi ; Mahler, Jeffrey ; Matl, Matthew ; Chentanez, Nuttapong ; Goldberg, Ken: REACH: Reducing False Negatives in Robot Grasp Planning with a Robust Efficient Area Contact Hypothesis Model.

[11] Deng, Jia ; Dong, Wei ; Socher, Richard ; Li, Li-Jia ; Li, Kai ; Fei-Fei, Li: ImageNet: A Large-Scale Hierarchical Image Database.

[12] Dertat, Arden: *Applied Deep Learning - Part 4: Convolutional Neural Networks*. November 2017. – URL https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. – Zugriffsdatum: 2020-09-14

[13] Frid-Adar, Maayan ; Klang, Eyal ; Amitai, Michal ; Goldberger, Jacob ; Greenspan, Hayit: Synthetic data augmentation using GAN for improved liver lesion classification. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, S. 289–293. – ISSN: 1945-8452

[14] Girshick, Ross: Fast R-CNN. In: *arXiv:1504.08083 [cs]* (2015), September. – URL http://arxiv.org/abs/1504.08083. – Zugriffsdatum: 2020-09-07. – arXiv: 1504.08083

[15] Girshick, Ross ; Donahue, Jeff ; Darrell, Trevor ; Malik, Jitendra: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *arXiv:1311.2524 [cs]* (2014), Oktober. – URL http://arxiv.org/abs/1311.2524. – Zugriffsdatum: 2020-09-07. – arXiv: 1311.2524

[16] Goswami, Subrata: *A deeper look at how Faster-RCNN works*. Juli 2018. – URL https://medium.com/@whatdhack/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd. – Zugriffsdatum: 2020-09-18

[17] Gupta, Ankush ; Vedaldi, Andrea ; Zisserman, Andrew: Synthetic Data for Text Localisation in Natural Images. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, S. 2315–2324. – URL http://ieeexplore.ieee.org/document/7780623/. – Zugriffsdatum: 2020-05-05. – ISBN 978-1-4673-8851-1

[18] Harrell, Frank: *Classification vs. Prediction*. Januar 2017. – URL https://fharrell.com/post/classification/. – Zugriffsdatum: 2020-09-02

[19] Harrell, Frank: *Damage Caused by Classification Accuracy and Other Discontinuous Improper Accuracy Scoring Rules*. März 2017. – URL https://fharrell.com/post/class-damage/. – Zugriffsdatum: 2020-09-02

[20] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *arXiv:1512.03385 [cs]* (2015), Dezember. – URL http://arxiv.org/abs/1512.03385. – Zugriffsdatum: 2020-08-28. – arXiv: 1512.03385

[21] Židek, Kamil ; Lazorík, Peter ; Piteľ, Ján ; Hošovský, Alexander: An Automated Training of Deep Learning Networks by 3D Virtual Models for Object Recognition. 11, Nr. 4, S. 496. – URL https://www.mdpi.com/2073-8994/11/4/496. – Zugriffsdatum: 2020-05-06. – ISSN 2073-8994

[22] Izidio, Diogo M. F. ; Ferreira, Antonyus P. A. ; Medeiros, Heitor R. ; Barros, Edna N. d.: An embedded automatic license plate recognition system using deep learning. 24, Nr. 1, S. 23–43. – URL http://link.springer.com/10.1007/s10617-019-09230-5. – Zugriffsdatum: 2020-04-07. – ISSN 0929-5585, 1572-8080

[23] Jiang, Yun ; Moseson, Stephen ; Saxena, Ashutosh: Efficient grasping from RGBD images: Learning using a new rectangle representation. In: *2011 IEEE International Conference on Robotics and Automation*, S. 3304–3311. – ISSN: 1050-4729

[24] Josifovski, Josip ; Kerzel, Matthias ; Pregizer, Christoph ; Posniak, Lukas ; Wermter, Stefan: Object Detection and Pose Estimation Based on Convolutional Neural Networks Trained with Synthetic Data. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, S. 6269–6276. – URL https://ieeexplore.ieee.org/document/8594379/. – Zugriffsdatum: 2020-05-07. – ISBN 978-1-5386-8094-0

[25] Karaoguz, Hakan ; Jensfelt, Patric: Object Detection Approach for Robot Grasp Detection. In: *2019 International Conference on Robotics and Automation (ICRA)*, S. 4953–4959. – ISSN: 1050-4729

[26] K.Dharpure, Jaydeo ; B. Potdar, M. ; Pandya, Manoj: Counting Objects using Convolution based Pattern Matching Technique. In: *International Journal of Applied Information Systems* 5 (2013), Juni, Nr. 8, S. 14–19. – URL http://research.ijais.org/volume5/number8/ijais13-450964.pdf. – Zugriffsdatum: 2020-08-27. – ISSN 22490868

[27] Komorkiewicz, Mateusz ; Kluczewski, Maciej ; Gorgon, Marek: Floating point HOG implementation for real-time multiple object detection. In: *22nd International Conference on Field Programmable Logic and Applications (FPL)*, S. 711–714. – ISSN: 1946-1488

[28] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F. (Hrsg.) ; Burges, C. J. C. (Hrsg.) ; Bottou, L. (Hrsg.) ; Weinberger, K. Q. (Hrsg.): *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., S. 1097–1105. – URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. – Zugriffsdatum: 2020-02-26

[29] LeCun, Yann ; Bottou, Leon ; Bengio, Yoshua ; Ha, Patrick: Gradient-Based Learning Applied to Document Recognition. (1998), S. 46

[30] Lenz, Ian ; Lee, Honglak ; Saxena, Ashutosh: Deep Learning for Detecting Robotic Grasps.

[31] Li, Fei-Fei: *Fei-Fei Li | Speaker | TED*. – URL https://www.ted.com/speakers/fei_fei_li. – Zugriffsdatum: 2020-09-29

[32] Mizuno, Kosuke ; Terachi, Yosuke ; Takagi, Kenta ; Izumi, Shintaro ; Kawaguchi, Hiroshi ; Yoshimoto, Masahiko: Architectural Study of HOG Feature Extraction Processor for Real-Time Object Detection. In: *2012 IEEE Workshop on Signal Processing Systems*, S. 197–202. – ISSN: 2162-3570

[33] Rajpura, Param S. ; Bojinov, Hristo ; Hegde, Ravi S.: Object Detection Using Deep CNNs Trained on Synthetic Images. . – URL http://arxiv.org/abs/1706.06782. – Zugriffsdatum: 2020-05-08

[34] Redmon, Joseph ; Angelova, Anelia: Real-time grasp detection using convolutional neural networks. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, S. 1316–1322. – ISSN: 1050-4729

[35] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. . – URL http://arxiv.org/abs/1506.02640. – Zugriffsdatum: 2020-04-03. – version: 5

[36] Redmon, Joseph ; Farhadi, Ali: YOLO9000: Better, Faster, Stronger. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE,

S. 6517–6525. – URL http://ieeexplore.ieee.org/document/8100173/. – Zugriffsdatum: 2020-02-26. – ISBN 978-1-5386-0457-1

[37] Redmon, Joseph ; Farhadi, Ali: YOLOv3: An Incremental Improvement.

[38] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Cortes, C. (Hrsg.) ; Lawrence, N. D. (Hrsg.) ; Lee, D. D. (Hrsg.) ; Sugiyama, M. (Hrsg.) ; Garnett, R. (Hrsg.): *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., S. 91–99. – URL http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf. – Zugriffsdatum: 2020-02-26

[39] Saha, Sumit: *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Dezember 2018. – URL https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. – Zugriffsdatum: 2020-09-09

[40] Szegedy, Christian ; Wei Liu ; Yangqing Jia ; Sermanet, Pierre ; Reed, Scott ; Anguelov, Dragomir ; Erhan, Dumitru ; Vanhoucke, Vincent ; Rabinovich, Andrew: Going deeper with convolutions. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA : IEEE, Juni 2015, S. 1–9. – URL http://ieeexplore.ieee.org/document/7298594/. – Zugriffsdatum: 2020-08-28. – ISBN 978-1-4673-6964-0

[41] Takagi, Kenta ; Mizuno, Kosuke ; Izumi, Shintaro ; Kawaguchi, Hiroshi ; Yoshimoto, Masahiko: A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, S. 2533–2537. – ISSN: 2379-190X

[42] Tobin, Josh ; Fong, Rachel ; Ray, Alex ; Schneider, Jonas ; Zaremba, Wojciech ; Abbeel, Pieter: Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. . – URL http://arxiv.org/abs/1703.06907. – Zugriffsdatum: 2020-05-07

[43] Tremblay, Jonathan ; Prakash, Aayush ; Acuna, David ; Brophy, Mark ; Jampani, Varun ; Anil, Cem ; To, Thang ; Cameracci, Eric ; Boochoon, Shaad ; Birchfield, Stan: Training Deep Networks with Synthetic Data: Bridging the

Reality Gap by Domain Randomization. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, S. 1082–10828. – URL https://ieeexplore.ieee.org/document/8575297/. – Zugriffsdatum: 2020-04-30. – ISBN 978-1-5386-6100-0

[44] Wang, Xiaoyu ; Han, Tony X. ; Yan, Shuicheng: An HOG-LBP human detector with partial occlusion handling. In: *2009 IEEE 12th International Conference on Computer Vision*, S. 32–39. – ISSN: 2380-7504

[45] Wang, Ziyan ; Lu, Jiwen ; Lin, Ruogu ; Feng, Jianjiang ; zhou, Jie: Correlated and Individual Multi-Modal Deep Learning for RGB-D Object Recognition. . – URL http://arxiv.org/abs/1604.01655. – Zugriffsdatum: 2020-03-02

[46] Zhou, Xinwen ; Lan, Xuguang ; Zhang, Hanbo ; Tian, Zhiqiang ; Zhang, Yang ; Zheng, Narming: Fully Convolutional Grasp Detection Network with Oriented Anchor Box. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, S. 7223–7230. – ISSN: 2153-0858

A Anhang

Produkt	S922X	NJ TX2	RP4	Coral Board	NJ Nano	Priorität
Arbeitsspeicher	+	++	+	–	+	3
CPU	++	++	o	+	+	4
GPU/TPU	o	++	–	++	+	5
Support	-	+	++	-	+	7
Speicher	o	++	o	o	o	2
Kompatiblität	-	++	++	-	++	6
Betriebssystem	o	o	+	o	o	1
Preis	+	–	++	+	+	8

Tabelle A.1: Bewertungstabelle SBC

Produkt	ODROID-N2 AMLOGIC S922X	NVIDIA Jetson TX2-Entwicklerkit	Raspberry Pi 4	Google Coral Dev Board	NVIDIA Jetson Nano
Arbeitsspeicher	4GB, DDR4	8GB DDR4	4GB	160KB	4GB 64-Bit
CPU	Quadcore ARM Cortex-A73, Dual Core Cortex A53 Cluster mit GPU, 1.8 GHz	Dual Core Denver 2 64 Bit CPU + Quad-Core ARM A57 Complex	Quadcore A72 (Arm v8) 64-Bit	ARM Cortex A53, 1.5 GHz	Quad-Core-ARM 57 mit 1,43 GHz
GPU	Mali G52	NVIDIA Pascal Architecture GPU, 256 NVIDIA Cuda Cores	VideoCore VI (VC6)	Vivante GC7000Lite	Grafikprozessor: NVIDIA Maxwell 128 Core NVIDIA CUDA
TPU	-	-	-	Edge-TPU coprocessor	-
Support	ODROID Wiki, verschd. Foren	NVIDIA Forum, NVIDIA Wiki	sehr viele Foren, Wiki	Coral Wiki, Forum	NVIDIA Forum, NVIDIA Wiki
Speicher	Micro-SD-Slot	32 GB Flash Storage	SD-Slot	Micro-SD Slot	Micro-SD-Slot
Schnittstellen	4x USB 3.0, Gigabit Ethernet, HDMI2.0 Video Ausgang, 40GPIO	USB 3.0 Type A, USB 2.0 Micro AB, HDMI, Gigabit Ethernet, Full Size SD, GPIOs, I2C, I2S, SPI, CAN, TTL UART, Display Expansion Header, Camera Expansion Header, Bluetoooth	Bluetooth 5.0, Wifi 2,4GHz, Gigabit Ethernet, 2x USB 2.0, 2x USB 3.0, 40 GPIO Pins, 2 Micro HDMI, 2-Lane MIPI DSI Display Port, 4 poliger Stereo Ausgang	40-pin I/O header, USB Micro-B for serial console, USB 3.0 Typ-A host, Gigabit Ethernet,	Gigabit-Ethernet, HDMI, DP, 4x USB 3.0, USB 2.0 Micro-B, GPIO, I2C, I2S, SPI, UART
Kompatiblität	OpenCV, Tensorflow, ROS	Tensorflow, PyTorch, Caffe, Keras, OpenCV, ROS	Tensorflow, OpenCV, Keras, PyTorch	Tensorflow, einzelne Plugins für OpenCV	Tensorflow, PyTorch, Caffe, Keras, OpenCV, ROS
Betriebssystem	Images of Ubuntu 18.04	preflashed with a Linux development	Linux verschiedene	Derivat von Debian Linux	Linux Ubuntu 18.04
Preis	102,33 €	498,61 €	62,27 €	189,21 €	109,00 €

Abbildung A.1: DetaillierteBewertungsgrundlage

Glossar

AlexNet Faltungsnetzwerk(CNN), das beim ILSVRC im Jahr 2012 mit einem deutlichen Vorpsrung zum Zweiplatzierten gewonnen hat..

Backpropagation Anpassung der Parameter des CNNs, nachdem das Bild das CNN durchlaufen hat..

Balanced Gute Anpassung der Parameter des Netzwerks. Dem Netzwerk ist es auch möglich in unbekannten Bildern Objekte zu detektieren..

Bounding Boxes Minimal mögliche Umschließung einer geometrischen Form. Im Rahmen dieser Arbeit entspricht die geometrische Form die zu detektierenden Objekte..

Creative Experience App Ein Programm, welches in die 3DExperience-Plattform integriert ist und für das Erstellen von Bildern und Videos für Marketingzwecke genutzt wird. Integriert ist eine Javascript-Konsole, die es ermöglicht Umweltparameter wie das Licht anzupassen..

Feature Map Die durch das Netzwerk herausgefilterten Merkmale der Objekte werden in Feature Maps dargestellt..

Ground-Truth Bounding Box Eine Referenz Bounding Box, die im Rahmen der Masterarbeit händisch für jedes Objekt definiert wird. Die Ground-Truth Bounding Box wird für die Bewertung der vorhergesagten Bounding Box genutzt..

ILSVRC Wettbewerb bei dem unterschiedliche Ansätze zur Objektdetektion verglichen werden. Der ImageNet Datensatz dient als Trainingsgrundlage.

Mini Factory Vormontagelinie, für kollaborative Arbeit zwischen Mensch und Roboter.

Overfitting Überanpassung der Parameter des Netzwerks. Das Netzwerk ist nicht oder nur schlecht in der Lage Objekte in unbekannten Bildern zu detektieren, da die Parameter auf den Trainingsdatensatz angepasst sind..

Sim-to-Real Gap Die Sim-to-Real Gap beschreibt die Diskrepanz der Ergebnisse, die bei der Anwendung eines Netzwerkes, welches in einer simulierten Umgebung trainiert wird, auf reale Szenarieren entsteht. Einflüsse die in der realen Welt auftreten werden in die simulierten Welt nicht berücksichtigt..

Underfitting Unteranpassung der Parameter des Netzwerks. Dem Netzwerk ist nicht oder nur schlecht möglich Objekte in unbekannten Bildern zu detektieren, da die Parameter nicht ausreichend angepasst wurden, um die Eigenschaften von Objekten zu filtern..

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original