HAW HAMBURG

Masterarbeit

Patrick Nagorski

Echtzeit-Objektverfolgung anhand von Videosequenzen

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Patrick Nagorski

Echtzeit-Objektverfolgung anhand von Videosequenzen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang *Master of Science Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter: Prof. Dr. Tim Tiedemann

Eingereicht am: 17. Februar 2022

Patrick Nagorski

Thema der Arbeit

Echtzeit-Objektverfolgung anhand von Videosequenzen

Stichworte

Maschinelles Lernen, Objekterkennung, Objektverfolgung, YOLO, Tiny-YOLO, DeepSORT, COCO, Open Images

Kurzzusammenfassung

Durch Objekterkennung können die Positionen und Klassen von Objekten identifiziert werden. Für diese Arbeit wurde mithilfe eines YOLO Netzwerkes ein Modell mit Bildern ausgewählter Objektklassen trainiert, welches anschießend für eine Objekterkennung dieser Objektklassen genutzt werden konnte. Die Informationen der Objekterkennung konnten im Anschluss mit dem DeepSORT Algorithmus verarbeitet werden, sodass eine Verfolgung von Objekten dieser Klassen durchgeführt werden konnte. Durch Anpassungen der verwendeten COCO und Open Images Datensätze konnte festgestellt werden, was für einen Einfluss diese auf die Objekterkennung und Objektverfolgung haben. Außerdem wurde durch eine Perfomanceanalyse festgestellt, wie leistungsfähig ein YOLO Modell und ein Tiny-YOLO Modell auf verschiedenen Geräten für eine Objekterkennung und Objektverfolgung genutzt werden kann. Durch Experimente konnte festgestellt werden, dass zufriedenstellende Objekterkennungen und Objektverfolgungen durchgeführt werden können.

Patrick Nagorski

Title of Thesis

Real-time object tracking based on video sequences

Keywords

Machine learning, object detection, object tracking, YOLO, Tiny-YOLO, DeepSORT, COCO, Open Images

Abstract

Object detection can be used to identify the positions and classes of objects. For this purpose a YOLO network was used to train a model with images of selected object classes which can be used for object detection of these object classes. The information from the object detection can be processed afterwards using the DeepSORT algorithm so that a tracking of objects of these classes can be performed. In addition a performance analysis was performed to evaluate how effective a YOLO model and a Tiny-YOLO model can be used on different devices for object detection and object tracking. Through experiments it was concluded that satisfying object detections and object trackings can be performed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AM All miss.

AP Average Precision.

Ass-IoU Association IoU.

AssA Association Accuracy.

AT All true.

CNN Convolutional Neural Network.

COCO Common Objects in Context.

CPU Central Processing Unit.

DeepSORT Simple Online and Realtime Tracking with a Deep Association Metric.

Det-IoU Detection IoU.

DetA Detection Accuracy.

FN False Negative.

FNA False Negative Associations.

FP False Positive.

FPA False Positive Associations.

FPS Frames per second.

GB Gigabyte.

GPU Graphics Processing Unit.

HOTA Higher Order Tracking Accuracy.

ID Identifikator.

IDSW ID Switches.

IoU Intersection over Union.

LiDAR Light Detection and Ranging.

Loc-IoU Localization IoU.

LocA Localization Accuracy.

mAP mean Average Precision.

MARS Motion Analysis and Re-identification Set.

ML Mostly lost.

MOT Multiple Object Tracking.

MT Mostly tracked.

MTA Measurement-to-Track Association.

NMS Non-maximum Suppression.

P Precision.

PASCAL VOC PASCAL Visual Object Classes.

PT Partially tracked.

R Recall.

RAM Random Access Memory.

SIFT Scale-invariant Feature Transform.

SORT Simple Online and Realtime Tracking.

SOT Single Object Tracking.

Tiny-YOLO Tiny You Only Look Once.

Tiny-YOLOv3 Tiny You Only Look Once Version 3.

TN True Negative.

TP True Positive.

TPA True Positive Associations.

XML Extensible Markup Language.

YOLO You Only Look Once.

YOLOv2 You Only Look Once Version 2.

YOLOv3 You Only Look Once Version 3.

YOLOv4 You Only Look Once Version 4.

1 Einleitung

1.1 Motivation

Objekterkennung wird in einer Vielzahl von Anwendungen, wie beispielsweise in der Automobilindustrie als Assistenzsystem verwendet, um Hindernisse zu erkennen. Allgemein wird die Objekterkennung zur automatischen Auswertung von Objekten in Bildern genutzt. Mittels Objektverfolgung lässt sich zusätzlich das Verhalten von Objekten innerhalb eines Videos feststellen, da jedes Objekt dabei eine einzigartige ID zugeordnet bekommt. Dies kann beispielsweise für eine Analyse genutzt werden, um zu zählen, wie viele Objekte eine virtuelle Linie überqueren und um welche Art von Objekt es sich dabei handelt. Zusätzlich lässt sich durch die Erkennung eines Verhaltens vorhersagen, wie sich das Objekt zukünftig bewegen wird, um beispielsweise Gefahrensituationen erkennen zu können.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung eines Systems, welches eine Objekterkennung und Objektverfolgung beliebiger Objektklassen ermöglicht. Der Fokus lag dabei auf der Analyse und Verbesserung der für das Training verwendeten Datensätze. Das System soll Trainingsdaten beliebiger Objektklassen beziehen und mittels geeigneter Algorithmen Objekte erkennen und verfolgen. Dieses System soll in unterschiedlichen Bereichen wie beispielsweise als Assistenzsystem in der Automobilindustrie eingesetzt werden können.

2 Grundlagen und verwandte Arbeiten

2.1 Arten der Objektidentifizierung

Die Objektidentifizierung beschreibt Bildverarbeitungsaufgaben, bei denen es um die Erkennung von Objekten in Bildern geht. [3] Es existieren verschiedene Arten der Objektidentifizierung, die sich folgendermaßen unterscheiden:

Bildklassifizierung:
Bei der Bildklassifizierung geht es um die Vorhersage der Klasse eines Objekts in einem Bild. Dabei wird jedem Bild eine Klasse zugeordnet. [3]

Objektlokalisierung:
Bei der Objektlokalisierung geht es darum, die Position eines oder mehrerer Objekte in einem Bild zu bestimmen und eine Bounding Box um sie herum zu zeichnen. [3]

Objekterkennung:
Die Objekterkennung kombiniert die Bildklassifizierung mit der Objektlokalisierung, sodass eines oder mehrere Objekte innerhalb eines Bildes lokalisiert und klassifiziert werden. [3]

Bildsegmentierung:
Bei der Bildsegmentierung wird jeder einzelne Bildpunkt innerhalb eines Bildes überprüft, ob dieser einer bestimmten Klasse eines Objektes angehört. [3]

Objektverfolgung:
Die Objektverfolgung ist eine Objekterkennung innerhalb einer Bildsequenz. Zusätzlich zur Klasse jedes Objektes wird jedem Objekt eine einzigartige ID zugewiesen, wodurch es bei der Verarbeitung der Bildsequenz eindeutig identifizierbar ist. [3]

Die verschiedenen Arten der Objektidentifizierung sind zur Veranschaulichung in Abbildung 2.1 grafisch dargestellt.

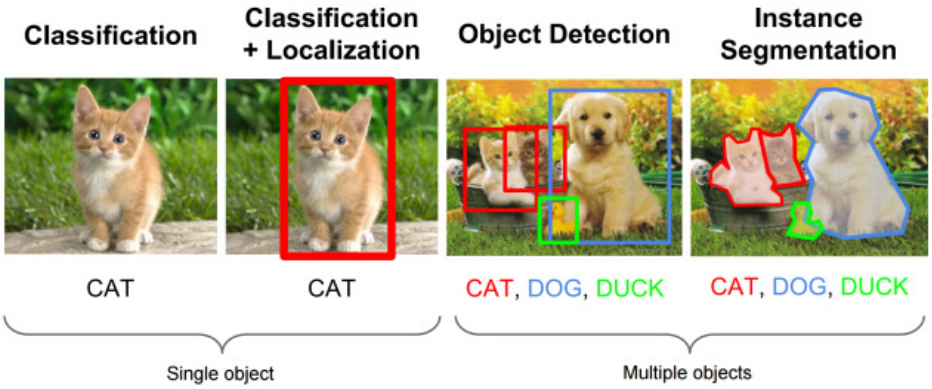


Abbildung 2.1: Arten der Objektidentifizierung. [34]

2.2 Objekterkennung

Die Objekterkennung zielt auf die Lokalisierung bestimmter Objektklassen innerhalb eines Bildes mittels Bounding Boxes ab. Die typische Ausgabe eines Objekterkennungssystems ist eine Menge von Bounding Boxes mit einer Klassenbezeichnung und einem Konfidenzwert, der aussagt, mit was für einer Wahrscheinlichkeit es sich um ein Objekt dieser Klasse handelt. [9] In Abbildung 2.2 ist eine Objekterkennung dargestellt, bei der zwei Objekte als Personen identifiziert wurden.

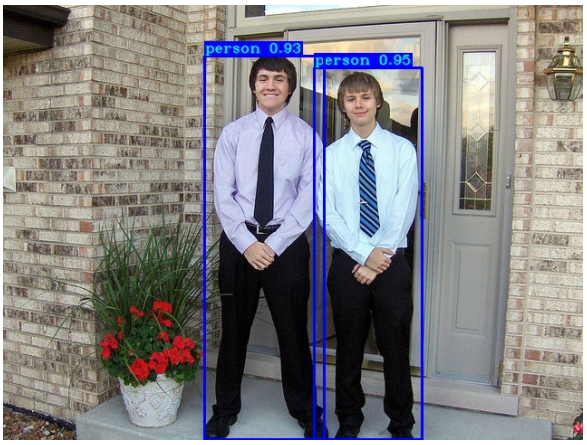


Abbildung 2.2: Objekterkennung anhand eines Bildes des COCO Datensatzes.

2.3 Objektverfolgung

Mittels Objektverfolgung werden (sich bewegende) Objekte innerhalb einer Bildsequenz eindeutig lokalisiert. Dabei wird der Zustand jedes Objektes anhand vorheriger Informationen geschätzt. Es wird zwischen Single Object Tracking (SOT) und Multiple Object Tracking (MOT) unterschieden. [4]

Beim SOT wird nur ein einziges Objekt verfolgt, auch wenn sich in der Umgebung mehrere Objekte befinden. Das zu verfolgende Objekt wird durch die Initialisierung des ersten Bildes der Bildsequenz bestimmt. Beim MOT dagegen werden alle in der Umgebung vorhandenen Objekte im Laufe der Zeit verfolgt, bei der Verwendung eines auf Erkennung basierenden Trackers können zusätzlich neue Objekte verfolgt werden, die innerhalb des Videos erscheinen. [4] In Abbildung 2.3 sind die einzelnen Schritte der Objektverfolgung dargestellt.

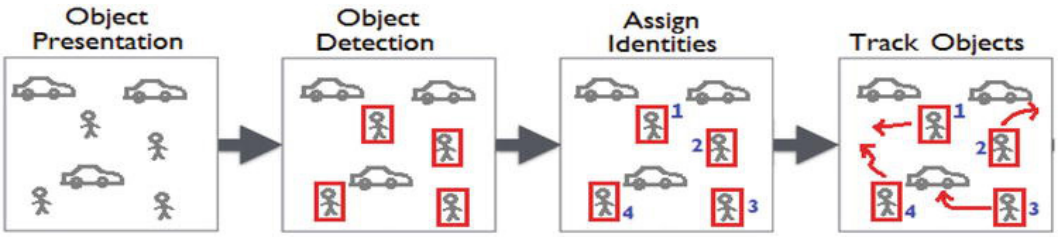


Abbildung 2.3: Schritte der Objektverfolgung. [28]

Ausgehend von einer Bildsequenz findet innerhalb eines Bildes eine Objekterkennung statt. Den Objekten wird jeweils eine ID zugeordnet und innerhalb der Bildsequenz bleibt die ID des jeweiligen Objektes auch bei der Veränderung ihrer Position erhalten.

Die größte Herausforderung bei der gleichzeitigen Verfolgung mehrerer Objekte ist, dass sie aus dem Sichtbereich verschwinden oder durch Interaktionen untereinander verdeckt werden können und dadurch eine neue ID erhalten, obwohl die Objekte bereits in den vorherigen Bildern existierten. Daher führt die Anwendung von SOT-Modellen zur Lösung von MOT zu schlechten Ergebnissen, die oft zu einer Abweichung ihrer tatsächlichen Objektklasse und zahlreichen ungewollten ID-Wechsel führen, da solche Modelle in der Regel nur schwer zwischen ähnlich aussehenden Objekten innerhalb einer Klasse unterscheiden können. [4]

In Abbildung 2.4 ist ein Szenario abgebildet, bei dem eine Person von einer anderen Person teilweise verdeckt wird. Auch in diesem Fall sollen im Idealfall beide Personen korrekt erkannt und die jeweiligen IDs beibehalten, die sie zuvor erhalten haben.

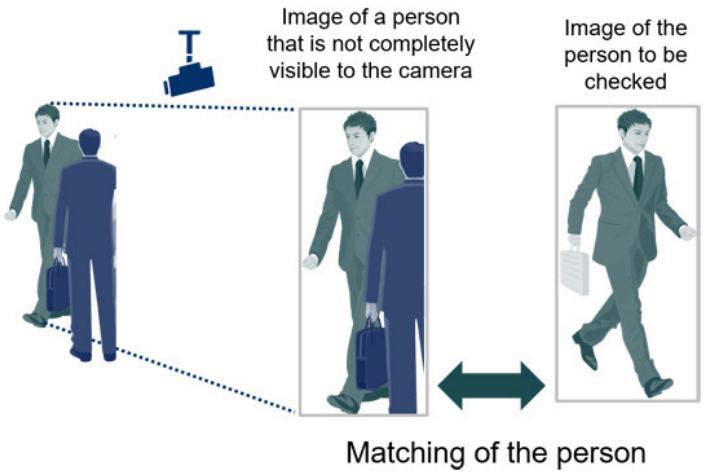


Abbildung 2.4: Beispiel eines verdeckten Objektes. [28]

In Abbildung 2.5 ist eine Objektverfolgung anhand von zwei Bildern innerhalb einer Bildsequenz dargestellt. Anhand des Bildes ist zu sehen, dass die IDs der Personen erhalten geblieben sind. Auch bei der Person 137, welche größtenteils von Person 129 verdeckt ist.

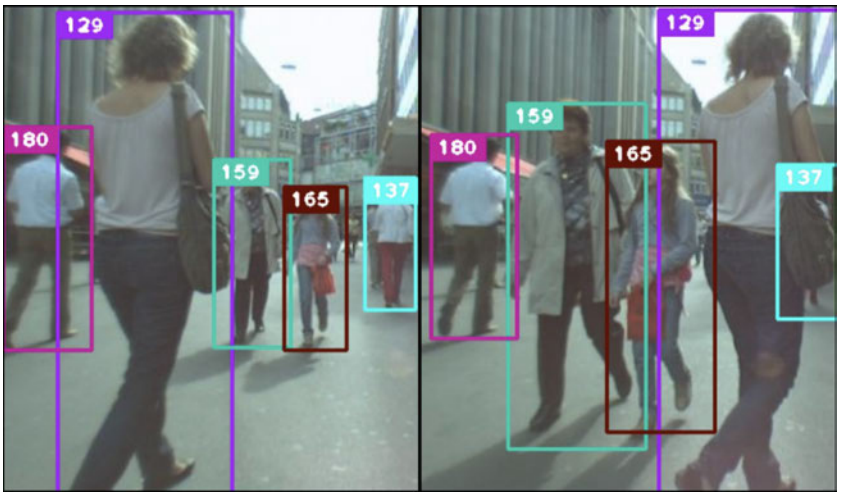


Abbildung 2.5: Beispiel einer Objektverfolgung. [28]

2.4 Qualitätsmaße (Metriken)

2.4.1 Metriken für die Objekterkennung

Objekterkennungsmetriken dienen als Maß dafür, wie gut ein Modell bei der Objekterkennung abschneidet. Dies ermöglicht einen objektiven Vergleich zwischen mehreren Erkennungssystemen. Zusätzlich können Erkennungssysteme anhand von bestehenden Benchmarks bewertet werden. Daher bieten bekannte Challenges, wie PASCAL VOC und COCO vordefinierte Metriken an, um zu bewerten, wie verschiedene Algorithmen zur Objekterkennung mit ihren Datensätzen abschneiden. [1]

Da eine Klassifizierung nur die Wahrscheinlichkeit des Auftretens einer Objektklasse im Bild bewertet, ist es für einen Klassifizierer eine einfache Aufgabe, korrekte Vorhersagen von falschen zu unterscheiden. Bei der Objekterkennung wird das Objekt jedoch mit einer Bounding Box lokalisiert, die mit dem entsprechenden Konfidenzwert verknüpft ist, um anzugeben, wie zuverlässig diese der erkannten Objektklasse entspricht. Um festzustellen, wie viele Objekte richtig und wie viele Objekte falsch erkannt wurden wird die Intersection over Union (IoU) verwendet. [1]

IoU ist eine Bewertungsmetrik, die die Ähnlichkeit zwischen der vorhergesagten Bounding Box und einer Ground-Truth-Bounding Box misst, um zu bewerten, wie gut die vorhergesagte Bounding Box ist. Der IoU-Wert liegt im Bereich von 0 bis 1. Je näher die beiden Boxen beieinander liegen, desto höher ist der IoU-Wert. Zur Berechnung des IoU-Wertes wird der Bereich der Überlappung zwischen der vorhergesagten Bounding Box und der Ground-Truth-Bounding Box und der Bereich der sowohl von der vorhergesagten Bounding Box als auch von der Ground-Truth-Bounding Box umschlossen wird, berechnet. Die IoU entspricht dabei der Division dieser beiden Werte.[27] In Abbildung 2.6 ist eine IoU grafisch dargestellt.

$$IoU_{pred}^{truth} = \frac{truth \cap pred}{truth \cup pred} =$$

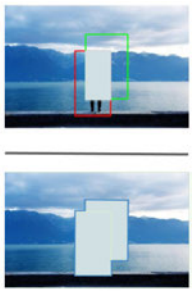


Abbildung 2.6: Grafische Darstellung einer IoU. [1]

Durch die Berechnung des IoU-Wertes für jede Erkennung wird ein Schwellwert für die Klassifizierungen festgelegt, wobei IoU-Werte über diesem Schwellenwert als richtige Vorhersagen und diejenigen unter diesem Schwellenwert als falsche Vorhersagen betrachtet werden. Die Vorhersagen werden in True Positive (TP), False Negative (FN) und False Positive (FP) unterteilt. [1] In Abbildung 2.7 sind diese Fälle grafisch dargestellt.

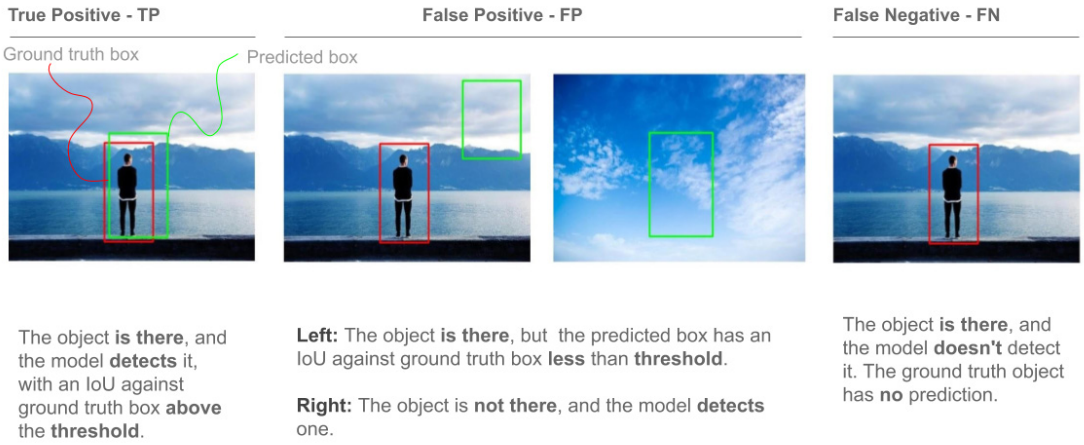


Abbildung 2.7: Grafische Darstellung von TP, FP und FN [1]

Auf der linken Seite befindet sich eine TP Vorhersage, in der Mitte befinden sich zwei FP Vorhersagen und auf der rechten Seite befindet sich eine FN Vorhersage.

Zusätzliche Bewertungen sind Accuracy, Precision (P) und Recall (R). Als Accuracy wird der Anteil der richtig vorhergesagten Objekte unter allen Vorhersagen bezeichnet. [1] Die Accuracy wird folgendermaßen berechnet:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.1}$$

Die Precision gibt die Wahrscheinlichkeit an, dass die vorhergesagten Bounding Boxes mit den tatsächlichen Bounding Boxes übereinstimmen. Die Precision Werte liegen zwischen 0 und 1. Eine hohe Precision bedeutet, dass die meisten erkannten Objekte mit den Ground-Truth Objekten übereinstimmen. [1] Die Precision wird folgendermaßen berechnet:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Der Recall ist die True-Positive-Rate, die die Wahrscheinlichkeit misst, dass Ground-Truth Objekte richtig erkannt werden. Die Recall Werte liegen ebenfalls zwischen 0 und 1. Ein hoher Recall-Wert bedeutet, dass die meisten Ground-Truth Objekte erkannt wurden. [1] Der Recall wird folgendermaßen berechnet:

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

In Abbildung 2.8 sind Precision und Recall zur Veranschaulichung grafisch dargestellt.

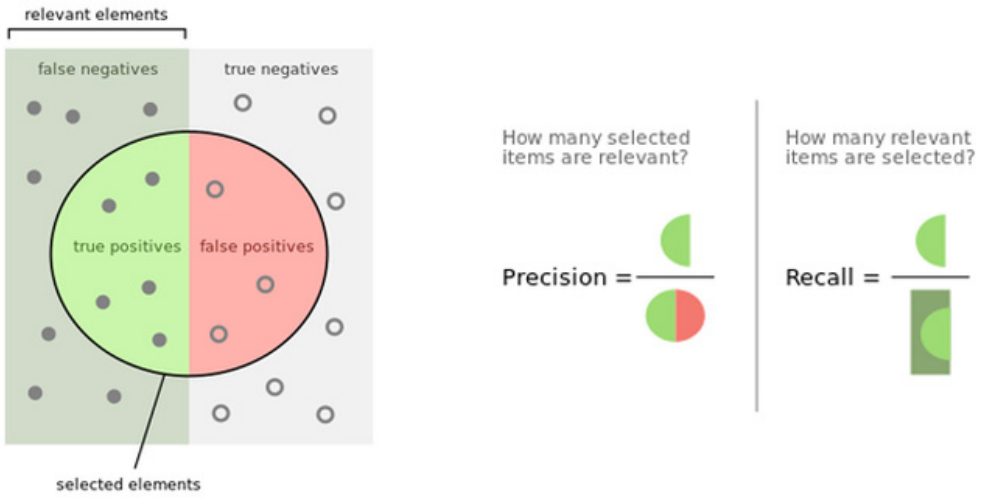


Abbildung 2.8: Precision und Recall. [26]

Eine weitere Metrik, die sowohl Precision als auch Recall zusammenfasst und eine Bewertung des Modells ermöglicht, ist die Precision-Recall Kurve. Da beide Metriken keine True Negatives verwenden, ist die Precision-Recall Kurve ein geeignetes Maß, um die Leistung des Modells bei unausgewogenen Datensätzen zu bewerten. Die Precision-Recall Kurve zeigt die Recall-Werte auf der x-Achse und die Precision-Werte auf der y-Achse, wobei jeder Punkt der Kurve die Recall- und Precision-Werte für einen bestimmten Konfidenzwert darstellt. [1] In Abbildung 2.9 ist eine Precision-Recall Kurve exemplarisch dargestellt.

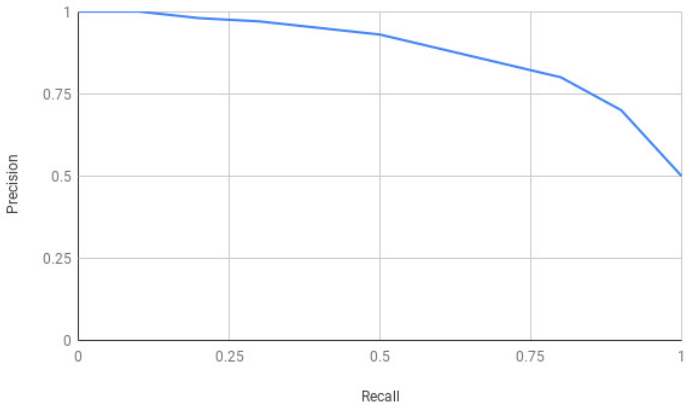


Abbildung 2.9: Precision-Recall Kurve. [1]

Bei einem steigenden Recall hat ein ideales Modell eine hohe Precision, andernfalls ist die Leistung des Modells schlecht. In dieser Situation sollte der Recall erhöht werden, um alle Ground-Truth Objekte korrekt zu erkennen, indem die Konfidenzschwelle gesenkt wird. Beim Vergleich mehrerer Modelle wird das Modell mit der höheren Kurve im Precision-Recall Diagramm als das leistungsstärkere angesehen. [1]

Die Average Precision (AP) dient als Maß für die Bewertung der Leistung von Objektdetektoren. Es handelt sich dabei um eine Metrik mit einer einzigen Zahl, die sowohl die Precision als auch den Recall beinhaltet und die Precision-Recall Kurve zusammenfasst. Die AP wird durch Mittelwertbildung der Precision-Werte über die Recall-Werte berechnet. Dies entspricht zudem der Fläche unter der Precision-Recall Kurve. [1] Die Average Precision wird folgendermaßen berechnet:

$$AP = \int_0^1 p(r)dr \tag{2.4}$$

Für den Fall, dass der Datensatz mehr als eine Klasse beinhaltet, wird die mean Average Precision (mAP) anhand des Durchschnitts der Average Precision über alle vorhandenen Klassen berechnet. Dadurch ist eine allgemeine Bewertung des Modells möglich. [1] Die mean Average Precision wird folgendermaßen berechnet:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.5}$$

Die PASCAL VOC Challenge verwendet die mAP als Metrik mit einem IoU-Schwellenwert von 0.5, während die COCO Challenge den Durchschnitt des mAPs über verschiedene IoU-Schwellenwerte (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95) mit einer Schrittweite von 0.05 ermittelt. Diese Metrik wird mit mAP@[.5,.95] bezeichnet. COCO mittelt somit nicht nur die AP über alle Klassen, sondern auch über die definierten IoU-Schwellenwerte. Zu beachten ist, dass mAP und AP oft als synonym verwendet werden, beispielsweise bei der COCO Challenge. [1]

2.4.2 Metriken für die Objektverfolgung

Für die Objektverfolgung existieren ebenfalls verschiedene Metriken, anhand dieser die Bewertung von Objektverfolgungsmodellen möglich ist. Higher Order Tracking Accuracy (HOTA) ist eine Metrik, anhand dieser Modelle bei der Verfolgung von mehreren Objekten (MOT) bewertet werden können. [20]

HOTA besteht aus der Kombination von drei IoU-Bewertungen. Es unterteilt die Aufgabe der Bewertung der Verfolgung in drei Teilbereiche (Lokalisierung, Erkennung und Assoziation) und berechnet für jeden Teilbereich eine Bewertung unter Verwendung einer IoU-Formel. Anschließend werden die drei IoU-Bewertungen für jeden Teilbereich zu einer endgültigen HOTA-Bewertung kombiniert. [20]

Lokalisierung:
Die Lokalisierung bewertet die räumliche Übereinstimmung zwischen einer vorhergesagten Erkennung und eines Ground-Truths. [20] In Abbildung 2.10 ist grafisch dargestellt, wie die Loc-IoU gebildet wird.

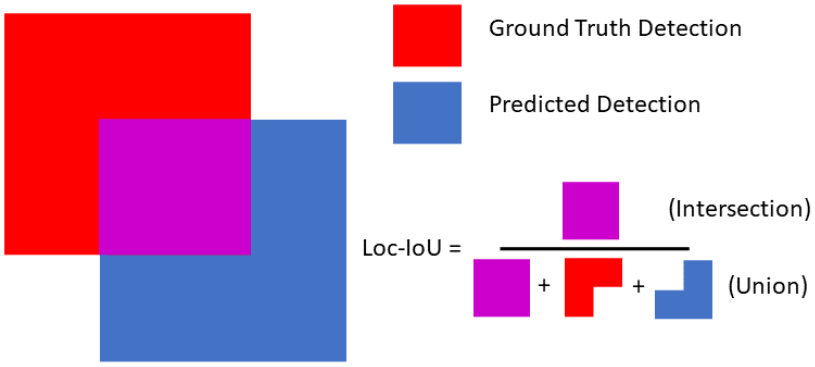


Abbildung 2.10: Grafische Darstellung der Loc-IoU. [20]

Die Localization Accuracy (LocA) wird berechnet, indem der Durchschnitt der Loc-IoU über alle Paare übereinstimmender vorausgesagter und tatsächlicher Erkennungen im gesamten Datensatz gebildet wird. [20] Die LocA wird mit folgender Formel berechnet:

$$LocA = \frac{1}{|TP|} \sum_{c \in TP} Loc\text{-}IoU(c) \tag{2.6}$$

Erkennung:
Die Erkennung misst die Übereinstimmung zwischen der Menge aller vorhergesagten Erkennungen und der Menge aller Ground-Truths. [20] Der Det-IoU wird mit folgender Formel berechnet:

$$Det\text{-}IoU = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{2.7}$$

Während Loc-IoU den Abgleich zwischen einer einzelnen Vorhersage und einem Ground-Truth misst, misst Det-IoU den Abgleich zwischen der Menge aller vorhergesagten Erkennungen und der Menge aller Ground-Truths. [20] DetA wird gemessen, indem der Det-IoU anhand der Anzahl der TPs, FNs und FPs über den gesamten Datensatz berechnet wird:

$$DetA = Det\text{-}IoU = \frac{|TP|}{|TP| + |FN| + |FP|} \tag{2.8}$$

Assoziation:
Die Assoziation misst, wie gut ein Tracker Erkennungen im Laufe der Zeit mit denselben Identitäten verknüpft, während die Menge der Identitätsverknüpfungen der Ground-Truths betrachtet wird. Dies wird gemessen, indem eine vorhergesagte Erkennung und ein zusammenpassendes Ground-Truth miteinander abgeglichen werden. Anhand der gesamten Verfolgungsstrecken der vorhergesagten Erkennungen und der Ground-Truths werden die Übereinstimmungen zwischen den beiden Strecken gemessen. [20]

Die Überschneidung zwischen zwei Verfolgungen wird als die Anzahl der True Positive Matches zwischen den beiden Verfolgungen gemessen, die als True Positive Associations (TPA) bezeichnen werden. Alle verbleibenden Erkennungen innerhalb der vorhergesagten Verfolgung entsprechen den False Positive Associations (FPA) und alle verbleibenden

Erkennungen innerhalb der Ground-Truth Verfolgung entsprechen den False Negative Associations (FNA). [20] Die Ass-IoU wird mit folgender Formel berechnet:

$$Ass\text{-}IoU = \frac{|TPA|}{|TPA| + |FNA| + |FPA|} \tag{2.9}$$

Auf diese Weise wird die Übereinstimmung zwischen zwei Verfolgungen gemessen, was für jedes Paar übereinstimmender Erkennungen (TPs) ein Maß dafür liefert, wie gut die Assoziation für diese Erkennung ist. [20] In Abbildung 2.11 ist ein visuelles Beispiel für die TPA, FNA und FPA anhand von zwei Verfolgungen dargestellt.

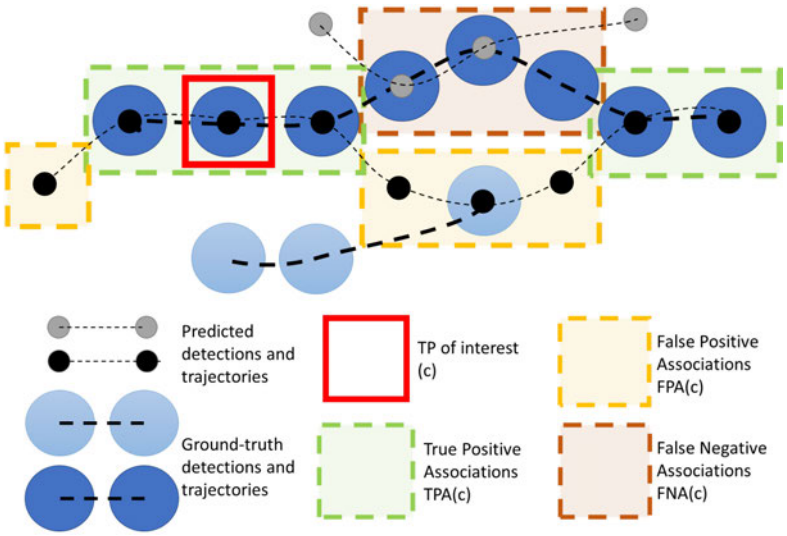


Abbildung 2.11: TPA, FNA und FPA grafisch dargestellt. [20]

Das rote Quadrat kennzeichnet das übereinstimmende TP-Paar aus einer Vorhersage und einem Ground-Truth, für die ein Assoziationswert ermittelt werden soll. Um zu ermitteln, wie gut die zeitliche Assoziation zwischen diesen Erkennungen ist, werden alle Erkennungen innerhalb der beiden Verfolgungen betrachtet. Die übereinstimmenden Erkennungen entsprechen den TPAs in Grün und die nicht übereinstimmenden entsprechen den FPAs in Gelb und FNAs in Braun.[20] Die AssA wird mit folgender Formel berechnet:

$$\begin{aligned} AssA &= \frac{1}{|TP|} \sum_{c \in TP} Ass\text{-}IoU(c) \\ &= \frac{1}{|TP|} \sum_{c \in TP} \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \end{aligned} \tag{2.10}$$

Der HOTA-Wert wird abschließend aus allen drei IoU-Werten mit folgender Formel berechnet:

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha} \tag{2.11}$$

$$HOTA = \int_{0<\alpha\leq 1} HOTA_\alpha \tag{2.12}$$

Zu beachten ist, dass sowohl DetA als auch AssA anhand eines ungarischen Matchings auf der Grundlage eines bestimmten Loc-IoU-Schwellenwerts (α) definiert wurden. Da sowohl die DetA- als auch die AssA-Bewertung von den Loc-IoU-Werten abhängen, werden diese Bewertungen über eine Reihe verschiedener α-Schwellenwerte berechnet. Für jeden Schwellenwert wird die endgültige Bewertung als geometrisches Mittel aus der Erkennungsbewertung und der Assoziationsbewertung berechnet. Durch anschließende Integration über die verschiedenen α-Schwellenwerte wird die Lokalisierungsgenauigkeit in die Endbewertung einbezogen. [20]

Die Verwendung des geometrischen Mittels für die Kombination von Erkennung und Assoziation stellt sicher, dass beide in der Endbewertung gleich gewichtet werden und dass die Bewertung auf 0 zurückgeht, wenn entweder die Erkennung oder die Assoziation auf 0 geht. [20]

Weitere Metriken sind ID Switches (IDSW), Mostly tracked (MT), Partially tracked (PT) und Mostly lost (ML). Bei IDSW handelt es sich um die Anzahl der Wechsel von IDs eines Objektes. MT entspricht der Anzahl der Objekte, die über 80% der "Lebensdauer" verfolgt werden konnten, PT entspricht der Anzahl der Objekte, die zwischen 20% und 80% der "Lebensdauer" verfolgt werden konnten und ML entspricht der Anzahl der Objekte, die weniger als 20% der "Lebensdauer" verfolgt werden konnten. [5]

Für die Objektverfolgung existieren ebenfalls Challenges. MOT Challenge[1] ist eine davon und bietet eine große Sammlung von Datensätzen, anhand dieser Objektverfolgungsmodelle bewertet werden können. Die Datensätzen bestehen aus einer Reihe von Videos mit dazugehörigen Ground-Truth Daten unterschiedlicher Szenen, in denen sich Fußgänger befinden. [10]

[1] https://motchallenge.net/

2.5 Verwandte Arbeiten

Liu et al. [19] entwickelten in ihrer Arbeit eine semantische Merkmalserkennung anhand einer Fußgängererkennung. In traditionellen Methoden sucht der Detektor nach Merkmalen durch Identifizierung von beispielsweise Kanten, Ecken und Flächen im gesamten Bild. In dieser Arbeit jedoch sucht der Detektor nach Merkmalen auf einer höheren Abstraktionsebene. In diesem Fall nach zentralen Positionen der Fußgänger, außerdem werden die Breite und Höhe der Fußgänger ausgehend der zentralen Positionen vorausgesagt. Die Detektion basiert zudem auf einer anchor-freien Methode, sodass keine vordefinierten Anchors für die Vorhersagen notwendig sind. [19] In Abbildung 2.12 ist die Pipeline des Detektors dargestellt. Die letzten Faltungen sind in zwei Kanäle aufgeteilt. Der erste Kanal liefert eine Heatmap, die die Position des Zentrums des Fußgängers darstellt und der zweite Kanal dient zur Vorhersage der Breite und Höhe für die erkannten zentralen Positionen der Fußgänger.

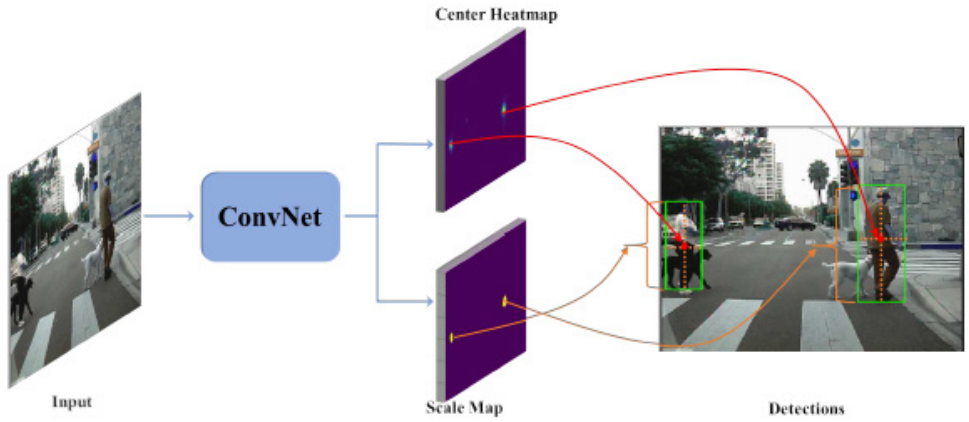


Abbildung 2.12: Pipeline des Detektors. [19]

Li et al. [17] entwickelten in ihrer Arbeit ebenfalls eine Fußgängererkennung. Der Fokus dieser Arbeit bestand darin, die Wirksamkeit der Erkennung bei schlechtem Wetter zu verbessern. Die Wirksamkeit nimmt bei der klassischen Fußgängererkennung bei einer schlechten Sichtbarkeit und unscharfen Umrissen der Fußgänger aufgrund von schlechten Wetterbedingungen stark ab. Um dieses Problem zu lösen, haben die Autoren drei auf YOLO basierende Deep-Learning-Ansätze entwickelt. Für das Training des neuronalen Netzwerkes haben Li et al. einen neuen Fußgänger-Datensatz entwickelt, welcher Fußgänger bei schlechtem Wetter beinhaltet. [17] In Abbildung 2.13 ist das Framework der Erkennungsmethoden dargestellt.

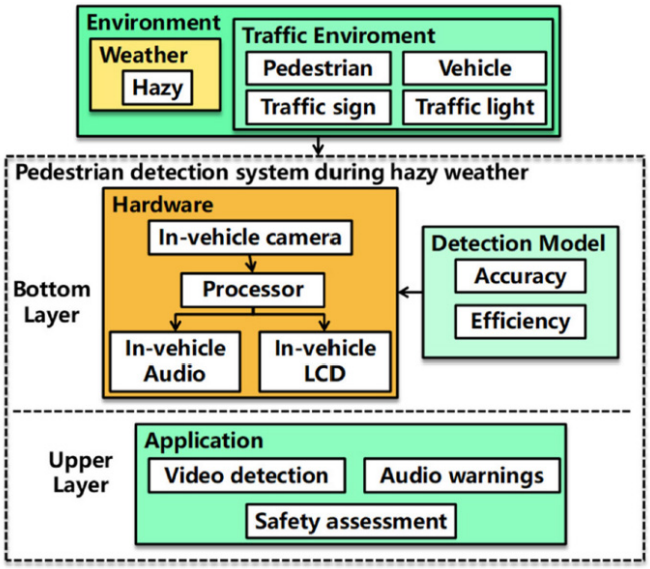


Abbildung 2.13: Framework der Erkennungsmethoden. [17]

Experimentelle Ergebnisse haben gezeigt, dass die von Li et al. vorgeschlagenen Methoden Fußgänger bei schlechtem Wetter zuverlässiger erkennen können und aktuelle Methoden hinsichtlich Genauigkeit und Performance deutlich übertreffen. [17] In Abbildung 2.14 sind die Methoden hinsichtlich der Average Precision (AP), Precision (P) und Recall (R) anhand des von den Autoren erstellten Datensatzes (HazePerson Dataset) abgebildet. Zusätzlich wurde der Anteil der Bilder, auf denen alle Fußgänger richtig erkannt wurden (AT), der Anteil der Bilder, bei denen alle Fußgänger nicht erkannt wurden (AM) und die Performance anhand der Frames per second (FPS) dargestellt.

Method	AP	P	R	AT	AM	FPS
HOG+SVM	43.5	71.3	54.7	39.9	9.1	–
Haar+Adaboost	35.6	64.7	50.2	44.8	27.9	–
YoloV3	81.6	87.2	83.3	83.9	7.7	22.2
Simple-Yolo	62.7	76.8	70.2	64.3	2.1	80.1
VggPrioriBoxes-Yolo	80.8	85.1	84.1	81.8	2.8	81.7
MNPrioriBoxes-Yolo	**86.6**	**88.0**	**89.3**	**86.7**	**0.7**	**151.9**

Abbildung 2.14: Ergebnisse der Erkennungen anhand des HazePerson Datensatzes. [17]

Dimitrievski et al. [6] entwickelten in ihrer Arbeit einen 2D-3D-Fußgänger-Tracker, der für Anwendungen in autonomen Fahrzeugen entwickelt wurde. Das System arbeitet nach dem Tracking-by-Detection-Prinzip und kann mehrere Fußgänger in komplexen städti-

schen Verkehrssituationen verfolgen. Durch die Verwendung eines verhaltensbasierten Bewegungsmodells ist das System in der Lage, unvorhersehbare Fußgängerbewegungen von verdeckten Fußgängern genau zu verfolgen. Die Autoren verwendeten eine Kombination von Kamera- und Light Detection and Ranging (LiDAR) Daten. [6] In Abbildung 2.15 ist ein Schaubild, mit den verschiedenen Pfaden des Informationsflusses dargestellt. Auf der linken Seite werden die CNN-Erkennungen (blau) mit den 3D-LiDAR-Daten aufeinander abgestimmt, auf der rechten Seite sind die Vorhersagen und Aktualisierungen des Bildebenen-Trackers (blau) und des Bodenebenen-Trackers (rot) abgebildet und in der Mitte erfolgt der Abgleich zwischen den Beobachtungen und Tracking-Schätzungen.

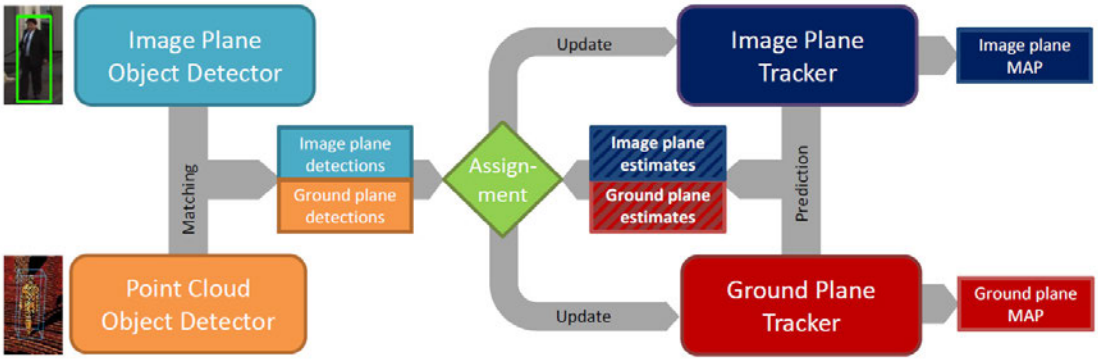


Abbildung 2.15: Pfade des Informationsflusses des Systems. [6]

3 Technologien und Algorithmen

3.1 YOLO

YOLO [25] steht für "You Only Look Once" und ist ein Objekterkennungsalgorithmus, der von Redmon et al. entwickelt wurde. Bei YOLO handelt es sich um einen einstufigen Objektdetektoren, wodurch er im Gegensatz zu anderen Objekterkennungsalgorithmen nur einen einzigen Durchlauf auf das gesamte Bild ausführt. Dadurch ist er performanter als andere Algorithmen, wobei er dennoch die Genauigkeit der Objekterkennung beibehält. [25]

Bei der Vorhersage gibt YOLO die Positionen der Objekte anhand von Bounding Boxes und den dazugehörigen Objektklassen und Konfidenzwerte aus. [22] YOLO betrachtet zudem bei der Vorhersage das gesamte Bild, wodurch der globale Kontext des Bildes einbezogen wird. [24] In Abbildung 3.1 sind verschiedene Objekterkennungsalgorithmen hinsichtlich ihrer Ausführungszeit und ihrer Average Precision (AP) anhand des COCO Datensatzes mit einer Nvidia GeForce TITAN X Grafikkarte zu sehen.

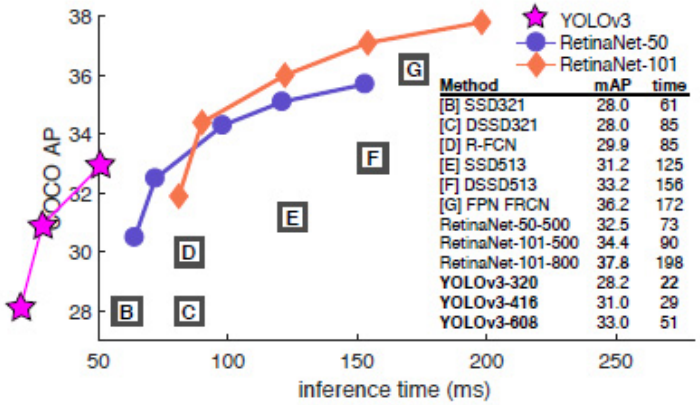


Abbildung 3.1: Vergleich der AP und Performance der Algorithmen anhand des COCO Datensatzes (Nvidia GeForce TITAN X). [25]

3.1.1 YOLOv3

Bei YOLOv3 handelt es sich um die dritte Version des YOLO Algorithmus. In YOLOv3 wurde der Algorithmus verbessert, wodurch er im Vergleich zu seinen Vorgängern robuster gemacht wurde. Außerdem gibt es wesentliche Unterschiede zwischen YOLOv3 und den älteren Versionen in Bezug auf die Bestimmung der Objektklassen, die Genauigkeit und die Performance. [24]

YOLOv3 verwendet unabhängige logistische Klassifikatoren und eine binäre Kreuzentropie Verlustfunktion für die Klassenvorhersagen während des Trainings, anstelle des in YOLOv2 verwendeten Softmax. Diese Änderungen ermöglichen die Verwendung komplexer Datensätze, wie Microsofts Open Images Datensatz für das Training, da der Datensatz überschneidende Objektklassen, wie z.B. “Mann” und “Person” enthält. [24]

In Abbildung 3.2 sind verschiedene Objekterkennungsalgorithmen hinsichtlich ihrer Average Precision (AP) anhand des COCO Datensatzes im Vergleich dargestellt. Die Zahl neben der AP steht dabei für die gewählte IoU als Schwellwert (50 entspricht einem IoU von 0.5). Die Buchstaben S, M und L signalisieren, ob sich die AP auf kleine, mittelgroße oder große Objekte bezieht.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Abbildung 3.2: Vergleich der AP von Objekterkennungsalgorithmen anhand des COCO Datensatzes. [25]

Die Algorithmen sind in zweistufige und einstufige Detektoren aufgeteilt. Die zweistufigen Detektoren suchen zunächst in der ersten Stufe nach den relevanten Objekten innerhalb eines Bildes. In der zweiten Stufe werden diese Objekte klassifiziert und die Bounding Boxes um die Objekte herum bestimmt. Die einstufigen Detektoren dagegen behandeln die Objekterkennung als ein Regressionsproblem, um die Objektklassenwahrscheinlichkeiten und Bounding Box Koordinaten zu ermitteln. [30]

Außerdem sind in der Abbildung die AP hinsichtlich der Objektgröße aufgezeigt, sodass die Genauigkeit bei kleinen, mittleren und großen Objekten zu sehen ist. Die AP für kleine Objekte in YOLOv2 liegt bei nur 5.0, wodurch die Objekterkennung von kleinen Objekten bei den anderen Algorithmen besser funktioniert. Bei YOLOv3 konnte die AP auf 24.1 erhöht werden, welches der höchsten AP in der Abbildung entspricht.

Die YOLOv3 Architektur besteht aus einem Faltungsnetzwerk mit insgesamt 106 Schichten. Die ersten 53 Schichten basieren auf dem Darknet-53 Netzwerk, welches für die Extraktion von Bildmerkmalen zuständig ist. Diese Bildmerkmale werden an die darauffolgenden 53 Schichten weitergereicht, in denen aufgrund der Merkmale die Positionen und die Objektklassen der Objekte erkannt werden. [25] Die Architektur des YOLOv3 Netzwerkes ist in Abbildung 3.3 dargestellt.

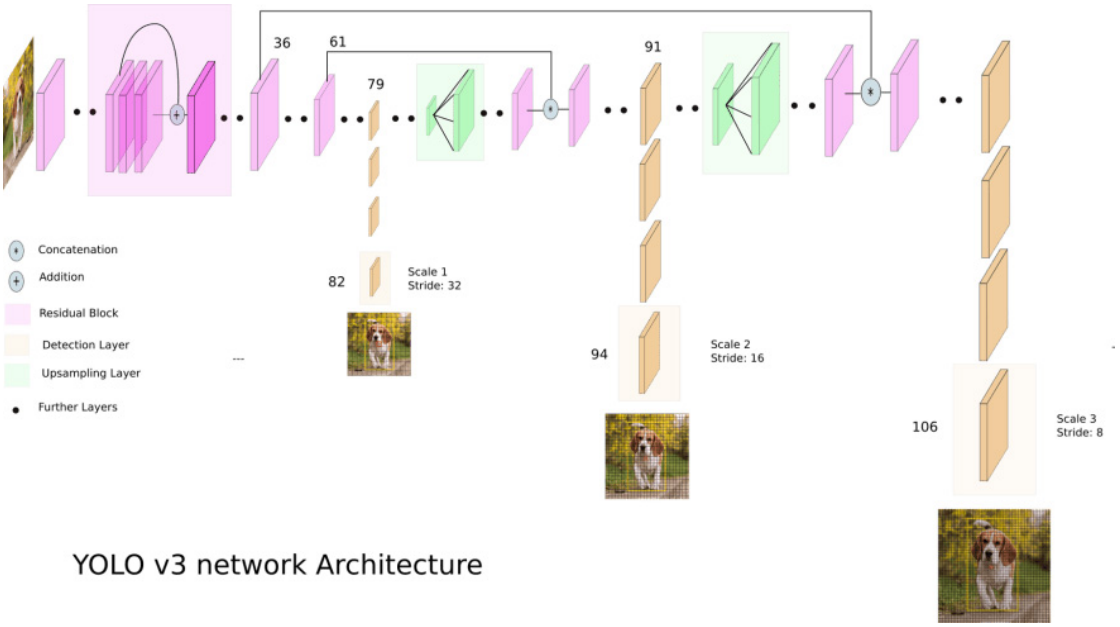


Abbildung 3.3: YOLOv3 Netzwerk-Architektur. [22]

Das Darknet-53 Netzwerk besteht hauptsächlich aus einer Folge von Faltungsschichten mit 1 x 1 und 3 x 3 Faltungsmasken. Außerdem beinhaltet das Netzwerk Residual-Schichten, um den Problemen der verschwindenden und explodierenden Gradienten entgegenzuwirken. [21] Dadurch bleibt das Training des Netzwerkes effizient und die Gewichte innerhalb der einzelnen Trainingsschritte werden nicht zu stark verändert, sodass das Netzwerk stabil bleibt. [11] [33]

Zur Verbesserung der Merkmalsextraktion wurden bei Darknet-53 die Pooling-Schichten durch weitere Faltungsschichten ersetzt, wodurch der Verlust von Low-Level-Merkmalen reduziert wurde. Dies führt dazu, dass die Erkennung kleiner Objekte besser möglich ist. [25] Zur Veranschaulichung ist das Darknet-53 Netzwerk in Abbildung 3.4 anhand eines 416 x 416 Eingabebildes mit einem Stride von 32 dargestellt.

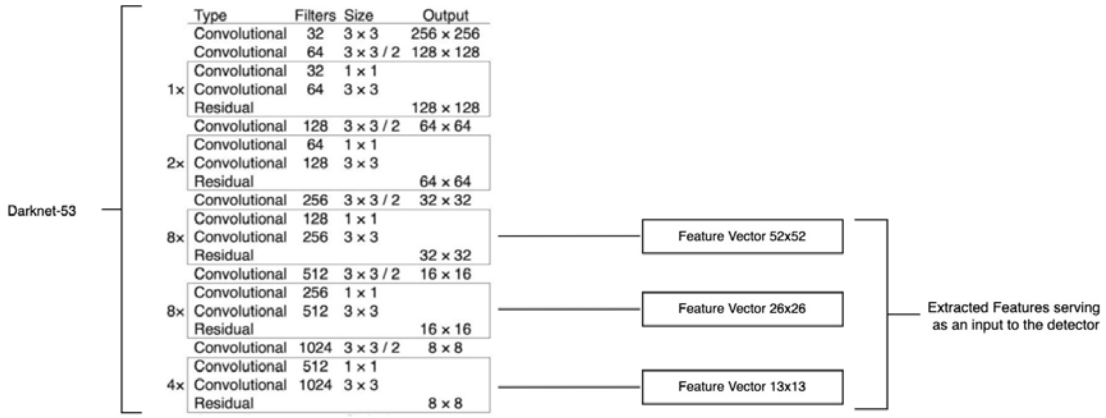


Abbildung 3.4: Darknet-53 Netzwerk anhand eines 416 x 416 Eingabebildes. [22]

Der Stride entspricht der Schrittweite der Faltungsmasken (Anzahl der Pixelverschiebungen in der Eingabematrix), anhand dieser das Eingabebild herunterskaliert wird. Ausgehend von der Form des Eingabebildes werden die Dimensionen der Feature Maps folgendermaßen berechnet: ($\frac{input\_shape}{Stride}$, $\frac{input\_shape}{Stride/2}$ und $\frac{input\_shape}{Stride/4}$). Daher sind die drei Skalierungen in diesem Fall 13 x 13, 26 x 26 und 52 x 52, wobei 13 x 13 für größere Objekte und 26 x 26 und 52 x 52 für mittlere und kleinere Objekte verwendet werden. [22] Die jeweiligen Feature Maps eines Eingabebildes sind exemplarisch in Abbildung 3.5 dargestellt.

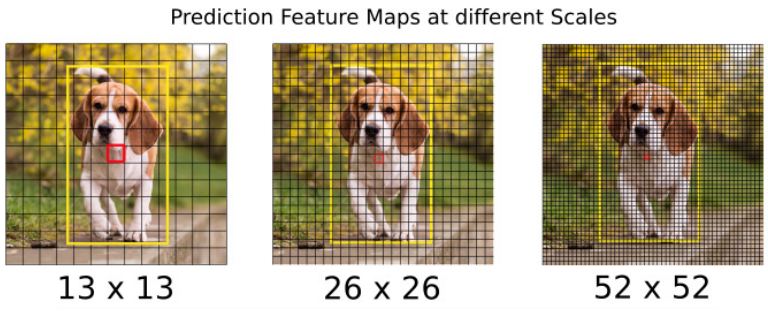


Abbildung 3.5: Feature Maps der verschiedenen Skalierungen. [15]

Die Objekterkennung findet in drei unterschiedlichen Skalierungen (Multi-scale-Detector) auf den Schichten 82, 94 und 106 statt. Der Multi-scale-Detector wird eingesetzt, um sicherzustellen, dass auch die kleinen Objekte erkannt werden. Das Netzwerk rechnet das Eingangsbild bis zur ersten Detektionsschicht (82) herunter bei der eine Erkennung unter Verwendung der Feature Map mit einem Stride von 32 erfolgt. Anschließend wird die Feature Map um den Faktor 2 hochgerechnet und mit der Feature Map der vorherigen Schichten verknüpft. Eine weitere Erkennung wird nun auf der zweiten Detektionsschicht (94) mit einem Stride von 16 durchgeführt. Das gleiche Verfahren wird wiederholt und eine letzte Erkennung erfolgt auf der dritten Detektionsschicht (106) mit einem Stride von 8. In jeder Skalierung sagt jede Zelle 3 Bounding Boxes voraus. [15] In Abbildung 3.6 ist der Aufbau einer Zelle einer Feature Map dargestellt.

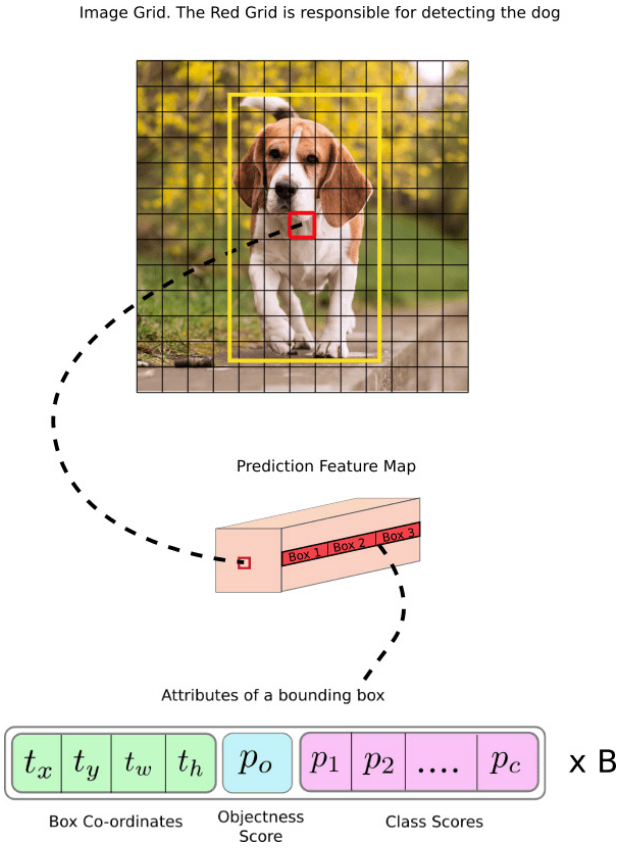


Abbildung 3.6: Aufbau einer Zelle einer Feature Map. [15]

Die Größe der Feature Map beträgt 13 x 13, sodass das Bild in 13 x 13 Zellen aufgeteilt wird. Die Zelle auf dem Eingabebild, die den Mittelpunkt der Ground-Truth-Box eines Objekts enthält wird als diejenige ausgewählt, die für die Vorhersage des Objektes

verantwortlich ist. In diesem Beispiel ist diese Zelle rot umrandet. Diese Zelle kann nun drei Bounding Boxes vorhersagen. Jede dieser Bounding Boxes besteht aus einem Vektor, welcher die Koordinaten der Bounding Box, den Objectness-Score und die Class-Scores beinhaltet. [15]

Der Objectness-Score gibt die Wahrscheinlichkeit an, dass sich ein Objekt innerhalb einer Bounding Box befindet. Er sollte für die rote und die benachbarten Gitterzellen nahezu 1 sein, während er z.B. für die Gitterzellen an den Ecken nahezu 0 ist. Die Class-Scores geben die Wahrscheinlichkeit an, dass das erkannte Objekt zu einer bestimmten Objektklasse gehört. [15]

Die Berechnung der Konfidenzwerte der potenziellen Objektklassen innerhalb einer Bounding Box einer Gitterzelle ist anhand folgender Beispielrechnungen veranschaulicht:

$$BB_{1score} = P_0 * \begin{pmatrix} p_1 \\ p_2 \\ ... \\ p_c \end{pmatrix} = \begin{pmatrix} 0.08 \\ 0.55 \\ ... \\ 0.05 \end{pmatrix} \xrightarrow{MAX} 0.55 \Rightarrow Hund \tag{3.1}$$

$$BB_{2score} = P_0 * \begin{pmatrix} p_1 \\ p_2 \\ ... \\ p_c \end{pmatrix} = \begin{pmatrix} 0.33 \\ 0.02 \\ ... \\ 0.09 \end{pmatrix} \xrightarrow{MAX} 0.33 \Rightarrow Fuchs \tag{3.2}$$

$$BB_{3score} = P_0 * \begin{pmatrix} p_1 \\ p_2 \\ ... \\ p_c \end{pmatrix} = \begin{pmatrix} 0.01 \\ 0.07 \\ ... \\ 0.48 \end{pmatrix} \xrightarrow{MAX} 0.48 \Rightarrow Wolf \tag{3.3}$$

YOLOv3 sagt die Bounding Boxes anhand von Offsets zu sogenannten Anchor Boxes voraus. Anchor Boxes sind eine Auswahl von vordefinierten Bounding Boxes mit einer bestimmten Höhe und Breite. Diese Boxen werden definiert, um den Maßstab und das Seitenverhältnis bestimmter Objektklassen zu erfassen, die Sie erkennen sollen und werden in der Regel anhand der Objektgrößen in ihren Trainingsdatensätzen ausgewählt. Der Grund für die Verwendung dieser Methode, anstelle der Vorhersage der tatsächlichen Breite und Höhe der Bounding Boxes ist, dass dadurch instabile Gradienten während des Trainings verhindert werden. [15]

Die folgenden Formeln beschreiben, wie die Netzwerkausgabe transformiert wird, um Bounding Box Vorhersagen zu erhalten:

$$b_x = \sigma(t_x) + c_x \tag{3.4}$$

$$b_y = \sigma(t_y) + c_y \tag{3.5}$$

$$b_w = (p_w)e^{t_w} \tag{3.6}$$

$$b_h = (p_h)e^{t_h} \tag{3.7}$$

b_x und b_y entsprechen den x,y Mittelpunktskoordinaten der Vorhersage, b_w und b_h entsprechen der Breite und Höhe der Vorhersage, t_x, t_y, t_w und t_h sind die Ausgaben des Netzwerkes, welche für den Offset verwendet werden, c_x und c_y entsprechen der oberen linken Koordinaten des Gitters und p_w und p_h entsprechen den Abmessungen des Anchors für die Bounding Box. Die Maße der Bounding Box werden durch Anwendung einer logarithmischen Transformation auf die Ausgabe und anschließende Multiplikation mit einem Anchor vorhergesagt. [15] In Abbildung 3.7 sind die Parameter der Formeln zur Veranschaulichung grafisch dargestellt.

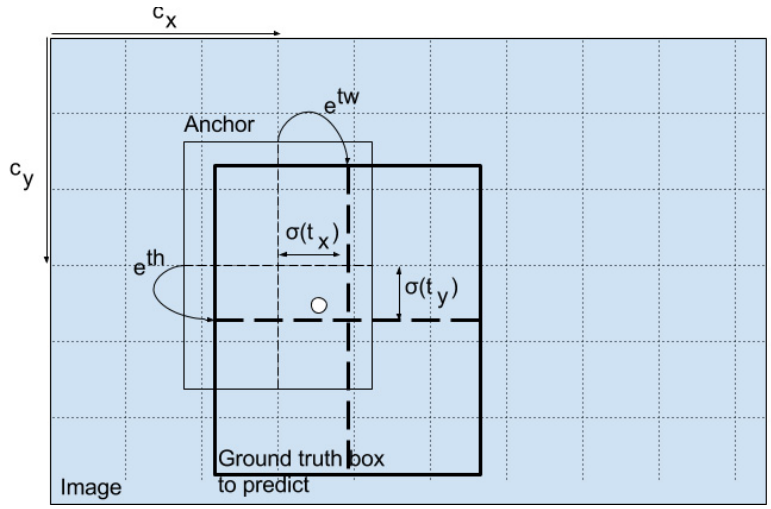
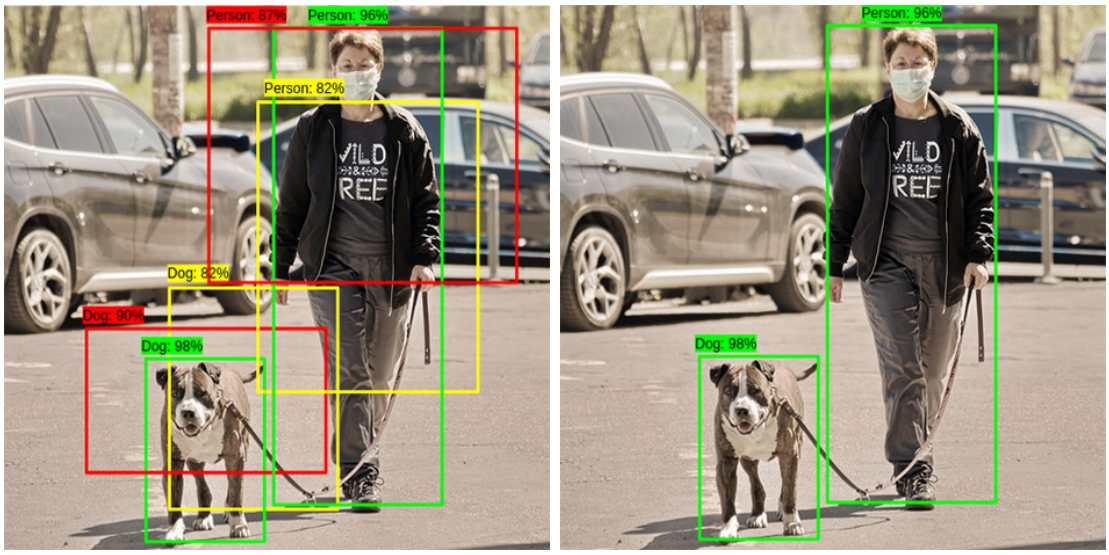


Abbildung 3.7: Grafische Darstellung der Parameter anhand von einer Anchor Box und einer Ground-Truth-Box. [15]

YOLOv3 wählt für die Erkennung eines Objektes die Bounding Box aus, deren Anchor die höchste IoU mit der Ground-Truth-Bounding Box aufweist. Somit ist diese ausgewählte Bounding Box für das Objekt verantwortlich. [15]

Für ein Bild der Größe 416 x 416 sagt YOLOv3 $(52 * 52 + 26 * 26 + 13 * 13) * 3 = 10647$ Bounding Boxes voraus. Für den Fall, wie in Abbildung 3.6, dass jedoch nur ein Objekt in diesem Bildbereich dargestellt ist, müssen die überschüssigen Bounding Boxes ausgefiltert werden. Dafür werden zunächst die Bounding Boxes unterhalb eines festgelegten Schwellwertes für den Konfidenzwert ignoriert. Anschließend werden mittels Non-maximum Suppression (NMS) die Mehrfacherkennungen aussortiert. Dabei werden der Konfidenzwert und die IoU der Bounding Boxes hinzugezogen. In Abbildung 3.8a sind Bounding Boxes mit den dazugehörigen Konfidenzwerten eingezeichnet. Dabei ist zu sehen, dass alle Bounding Boxes das jeweilige Objekt enthalten, aber nur die grünen Bounding Boxes den besten Bounding Boxes für die Erkennung des Objekts entsprechen. [29]

Bei der NMS wird die Bounding Box mit dem höchsten Konfidenzwert ausgewählt, danach werden alle anderen Boxes mit einer hohen Überlappung entfernt. Dabei wird die grüne Bounding Box für den Hund ausgewählt, da sie den höchsten Konfidenzwert von 98% hat. Anschließend werden die gelben und roten Boxes für den Hund entfernt, weil sie eine hohe Überlappung mit der grünen Box haben. Dieser Prozess wird so lange wiederholt, bis die Anzahl der Bounding Boxes nicht mehr reduziert werden kann. Ein exemplarisches Ergebnis der NMS ist in Abbildung 3.8b dargestellt. [29]



(a) Vor Durchführung der NMS. [29]

(b) Nach Durchführung der NMS. [29]

Abbildung 3.8: Beispielbild vor und nach Durchführung der NMS im Vergleich. [29]

3.1.2 Tiny-YOLOv3

Tiny-YOLOv3 ist ein leichtgewichtiger Algorithmus, der auf der Grundlage des YOLOv3 Algorithmus basiert. Tiny-YOLOv3 besteht aus einem kleineren Darknet-53 Netzwerk, welches auf 7 Faltungsschichten und 6 Max-Pooling-Schichten reduziert wurde und verwendet im Gegensatz zum dreistufigen YOLOv3 Netzwerk ein zweistufiges 13 x 13 und 26 x 26 Netzwerk zur Vorhersage der Objekte. [32] Die Architektur des Tiny-YOLOv3 Netzwerkes ist in Abbildung 3.9 dargestellt.

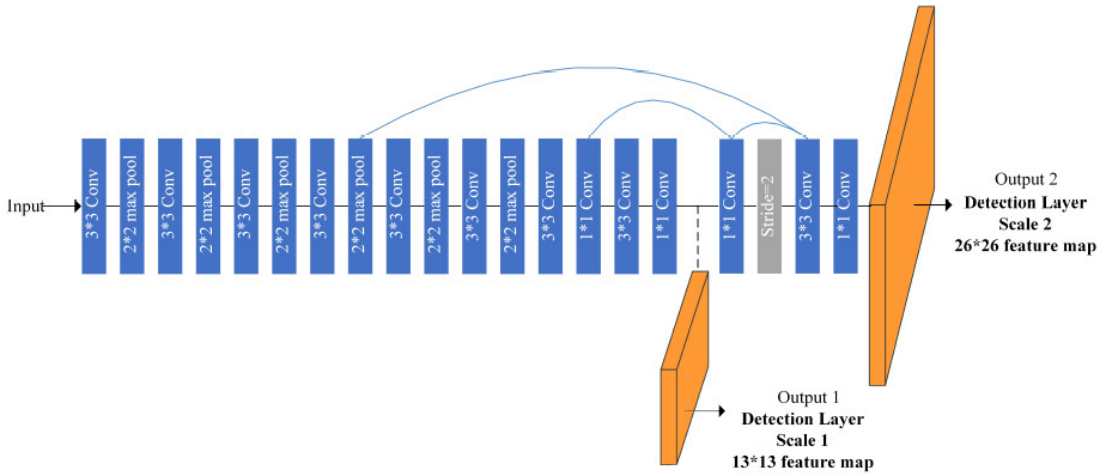


Abbildung 3.9: Tiny-YOLOv3 Netzwerk-Architektur. [7]

3.2 DeepSORT

DeepSORT steht für "Simple Online and Realtime Tracking with a Deep Association Metric" und ist ein Ansatz zur Verfolgung mehrerer Objekte mit dem Schwerpunkt auf einfachen und effektiven Algorithmen. [31]

DeepSORT ist eine Verbesserung des SORT Algorithmus, bei dem das Aussehen des erkannten Objektes hinzugezogen wird (Deep Appearance Descriptor). Dies führt dazu, dass die Anzahl der ID-Wechsel eines Objektes während einer Objektverfolgung reduziert wird. Außerdem ist DeepSORT robuster gegenüber Verdeckung von Objekten und Veränderung von Blickwinkel. Mit DeepSORT konnten die ID-Wechsel eines Objektes um 45% reduziert werden. [31] DeepSORT umfasst vier Kernkomponenten:

Erkennung:
Im ersten Schritt werden mittels eines Objekterkennungsalgorithmus die Objekte eines einzelnen Frames erkannt, beispielsweise mittels des YOLO Algorithmus. Je zuverlässiger die Objekterkennung funktioniert, desto besser funktioniert die Objektverfolgung. [14]

Schätzung:
Im nächsten Schritt wird die vorherige Erkennung vom aktuellen Frame zum nachfolgenden Frame weitergereicht. Falls eine Erkennung einem Ziel zugeordnet werden kann, wird diese dazu verwendet, um den Zustand des Ziels zu aktualisieren. Falls keine Erkennung einem Ziel zugeordnet werden kann, wird dessen Zustand mittels eines Kalman-Filters vorhergesagt. [14]

Der Kalman-Filter ist ein Algorithmus, der eine Reihe von im Laufe der Zeit beobachteten Daten verwendet, die Rauschen und andere Ungenauigkeiten enthalten, um unbekannte Variablen mit größerer Genauigkeit zu schätzen. [13] Der Algorithmus läuft rekursiv ab, indem er nur die aktuellen Eingangsmessungen und den zuvor berechneten Zustand verwendet. Dadurch kann er in Echtzeit arbeiten. Aus diesem Grund werden Kalman-Filter häufig für die Objektverfolgung eingesetzt. [2]

Der Algorithmus arbeitet in zwei Schritten: **Vorhersage** und **Korrektur**. In der Vorhersage Phase erstellt der Kalman-Filter Schätzungen des aktuellen Systemzustandes zusammen mit dessen Unsicherheiten. In der Korrektur Phase werden die Schätzungen der nachfolgenden Messung anhand der Vorhersagen aktualisiert. Der tatsächlich gemessene Wert des Systemzustandes wird unter Berücksichtigung des wahren Systemzustandes und des Messfehlers angegeben. [12]

Das Verfolgungsszenario wird mittels eines achtdimensionalen Vektors $(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ beschrieben. Die Parameter (u, v) entsprechen der Bounding Box Mittelpunktposition, γ dem Seitenverhältnis, h der Höhe, außerdem beinhaltet der Vektor die jeweiligen Geschwindigkeiten in Bildkoordinaten. Es wird ein Kalman-Filter mit konstanter Bewegungsgeschwindigkeit und einem linearen Beobachtungsmodell verwendet, wobei die Bounding Box Koordinaten (u, v, γ, h) als direkte Beobachtungen des Objektzustands betrachten werden. [31]

Vereinigung:
Bei der Zuordnung von Erkennungen zu vorhandenen Zielen wird die Bounding Box jedes Ziels durch die Vorhersage seiner neuen Position im letzten Frame geschätzt. In

diesem Schritt werden mittels der IoU zwischen jeder Erkennung und allen vorhergesagten Bounding Boxes der vorhandenen Ziele eine Zuordnungskostenmatrix berechnet. Anhand dieser werden die Gewichtungen von Fehlklassifizierungen angepasst. Mittels der ungarischen Methode (Kuhn-Munkres-Algorithmus) kann eine eindeutige Zuordnung hergestellt werden, indem die Maximum-Weight-Matchings in bipartiten Graphen gefunden werden. [31]

Erstellung und Löschung von Identitäten:
Für jede Verfolgung wird die Anzahl der Frames seit der letzten erfolgreichen Zuordnung gezählt. Dieser Zähler wird während der Kalman-Filter-Vorhersage erhöht und auf 0 zurückgesetzt, wenn das Tracking mit einer Vorhersage verknüpft worden ist. Trackings, die eine vordefinierte Zeitdauer überschreiten, gelten als Objekte, welche die Szene verlassen haben und werden aus der Tracking-Menge gelöscht. Für jede Erkennung, die keinem bestehenden Tracking zugeordnet werden kann, wird ein neues Tracking erstellt. Diese neuen Trackings werden in den ersten drei Frames als vorläufige Trackings eingestuft. Während dieser Zeit wird eine erfolgreiche Zuordnung des Objektes bei jedem Zeitschritt erwartet. Trackings, die innerhalb der ersten drei Frames nicht erfolgreich zugeordnet werden können, werden gelöscht. Dadurch werden die False Positives Fälle reduziert. [31] Die gesamte DeepSORT Architektur ist in Abbildung 3.10 dargestellt.

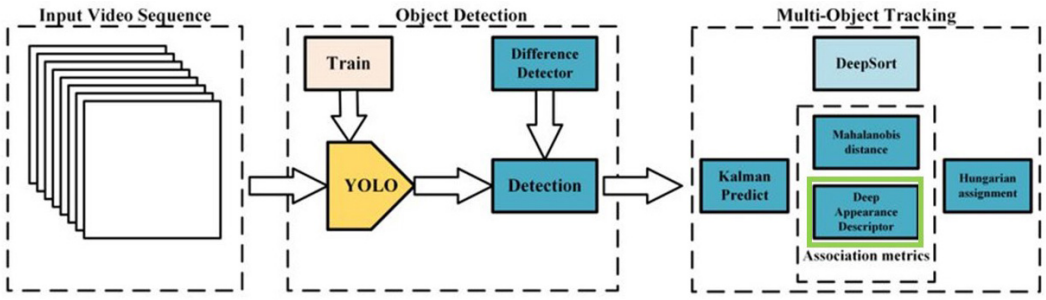
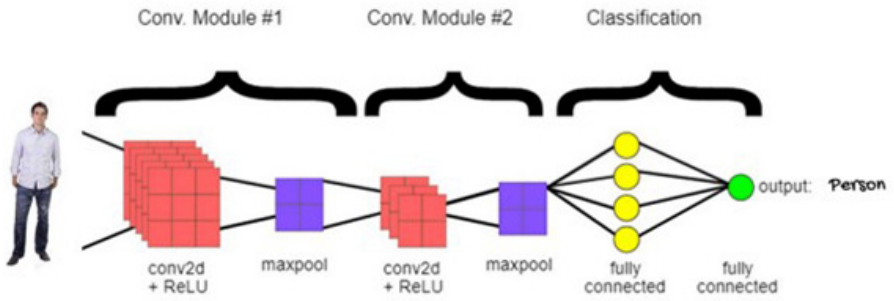
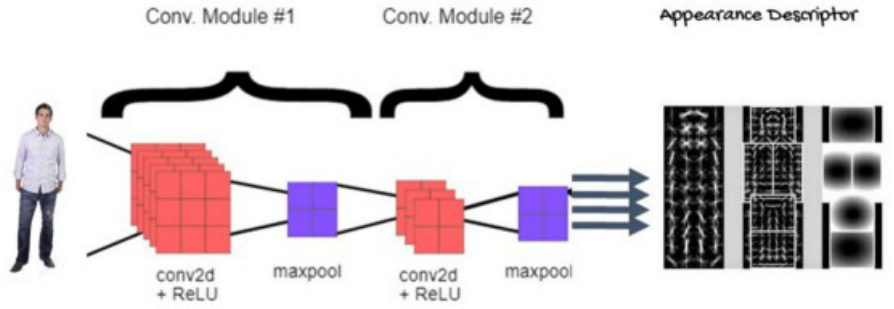


Abbildung 3.10: DeepSORT Architektur. [14]

Für den Deep Appearance Descriptor wurde auf der Grundlage eines Personendatensatzes ein Netzwerk trainiert. Diesem Netzwerk wurde die letzte Klassifizierungsschicht entfernt, sodass das Netzwerk mit einem Merkmalsvektor endet. [14] Zur Veranschaulichung ist das Vorgehen in Abbildung 3.11 dargestellt.



(a) Netzwerk ohne entfernter Klassifizierungsschicht.



(b) Netzwerk mit entfernter Klassifizierungsschicht.

Abbildung 3.11: Netzwerk mit und ohne entfernter Klassifizierungsschicht im Vergleich. [14]

Anhand des Merkmalsvektors wird mittels Nearest-Neighbor Abfragen eine Measurement-to-Track Association (MTA) hergestellt. Mittels der MTA wird die Beziehung zwischen einer Schätzung und einem bestehenden Trackings bestimmt. In diesem Prozess können Trackings, die bestimmte Attributskriterien erfüllen, über Filter ausgewählt werden. [8][14]

3.3 Relevante Datensätze

Open Images Datensatz:
Der Open Images Datensatz ist ein von Google erstellter Datensatz. Er beinhaltet 9 Millionen beschriftete Bilder. Die Beschriftungen sind unter anderem Labels, Bounding Boxes und Objektsegmentierungsmasken. Insgesamt enthält der Datensatz 16 Millionen Bounding Boxes für 600 Objektklassen auf 1.9 Millionen Bildern und ist somit der größte existierende Datensatz mit Beschriftungen, Objektpositionen und den dazugehörigen Klassifizierungen. Die Bounding Boxes wurden größtenteils manuell von professionellen

Annotatoren gezeichnet, um Genauigkeit und Konsistenz zu gewährleisten. Die Bilder sind sehr vielfältig und enthalten oft komplexe Szenen mit mehreren Objekten. Im Durchschnitt beinhalten die Bilder des Open Images Datensatzes 8.3 Objekte pro Bild. [16]

COCO Datensatz:
COCO steht für "Common Objects in Context" und ist ein von Microsoft entwickelter Datensatz, der eine große Anzahl von Bildern enthält, auf denen gewöhnliche Objekte in komplexen Alltagsszenen abgebildet sind. Damit unterscheidet sich COCO von anderen Datensätzen, die sich nur auf die Bildklassifizierung, die Lokalisierung von Bounding Boxes oder die Objektsegmentierung beziehen. [23] Der Datensatz besteht aus 2.5 Millionen beschrifteten Objekten in 328.000 Bildern. Er beinhaltet Beschriftungen von 80 Objektklassen. In dem COCO Datensatz befinden sich im Durchschnitt 3.5 Objektklassen und 7.7 Objekte pro Bild. [18]

In der Tabelle 3.1 sind die Informationen der beiden Datensätze im Vergleich gegenübergestellt.

Tabelle 3.1: Open Images vs. COCO

	Open Images	COCO
Ersteller	Google	Microsoft
Anzahl Bilder insgesamt	9.000.000	330.000
Anzahl beschrifteter Bilder	1.900.000	>200.000
Anzahl Beschriftungen	16.000.000	1.500.000
Anzahl Objektklassen	600	80

4 Implementierung

4.1 Systembeschreibung

Als Grundlage für das entwickelte System wurde das von pythonlessons implementierte System[1] verwendet und um weitere Funktionalitäten erweitert. Bei dem verwendeten System handelt es sich um eine in TensorFlow 2 einwickelte YOLO Implementierung. Über eine Vielzahl von Parameter lässt sich das System individuell steuern.

Die Architektur des entwickelten Systems ist in Abbildung 4.1 vereinfacht dargestellt. Das System besteht aus sieben Kernkomponenten: Datenbeschaffung, Datenvorbereitung, Training, Evaluation der Objekterkennung und Objektverfolgung, Objekterkennung und Objektverfolgung. Außerdem wurden Tools eingesetzt, mit denen die Datensätze manuell gefiltert, manuell beschriftet und mittels Data Augmentation vervielfältigt werden können. Diese Tools wurden hinzugefügt, um die Qualität der Datensätze zu verbessern.

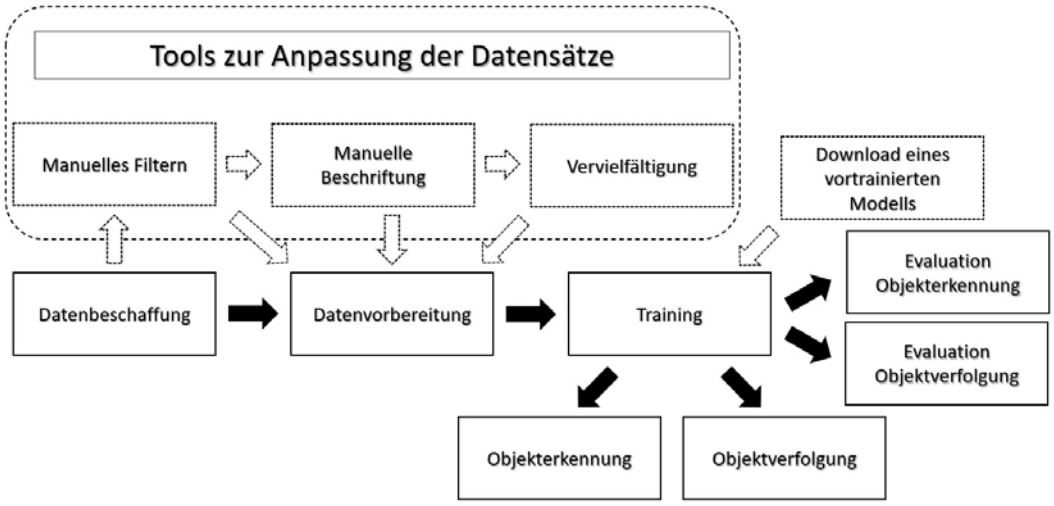


Abbildung 4.1: Architektur des entwickelten Systems.

[1] https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3

Datenbeschaffung:
Das System beinhaltet eine integrierte Datenbeschaffung, wodurch der Download beschrifteter Bilddaten entweder aus dem COCO oder dem Open Images Datensatz möglich ist. Über festgelegte Parameter lässt sich der zu verwendende Datensatz, die Anzahl der beschrifteten Trainings- und Testbilder und die relevanten Objektklassen festlegen. Es ist außerdem möglich, mehrere Objektklassen anzugeben, falls eine Objekterkennung bzw. Objektverfolgung anhand von mehreren Objektklassen gleichzeitig erforderlich ist. Über einen Parameter lassen sich zudem die vorhandenen Objektklassen des jeweiligen Datensatzes ausgeben.

Für den ausgewählten Datensatz wird ein neuer Ordner mit dem Namen des Datensatzes angelegt. In diesem Ordner sind die Daten in Trainings- und Testdaten aufgeteilt. In den Trainings- und Testordnern befinden sich weitere Ordner, die den Namen der jeweiligen Objektklasse haben. In diesen Ordnern befinden sich die Bilder und die Beschriftungen.

Datenvorbereitung:
Für die Datenvorbereitung werden die zuvor heruntergeladenen Beschriftungen zunächst ins PASCAL VOC Format und anschließend ins YOLO Format konvertiert. Die XML Dateien im PASCAL VOC Format beinhalten Informationen über die jeweiligen Bilder, wie zum Beispiel den Dateinamen, den Pfad zum Bild, die Auflösung des Bildes und die Koordinaten der einzelnen Bounding Boxes innerhalb des Bildes.

Für die Konvertierung ins YOLO Format werden zwei Dateien erstellt, in der ersten Datei befinden sich alle im Datensatz vorkommenden Objektklassen und in der zweiten Datei befinden sich in jeder Zeile der Dateipfad des jeweiligen Bildes, die Koordinaten der Bounding Boxes, die sich in dem Bild befinden und die jeweiligen Objektklassenindizes aus der ersten Datei. Die jeweiligen Formate sind in Abbildung 4.2 zur Veranschaulichung exemplarisch dargestellt.

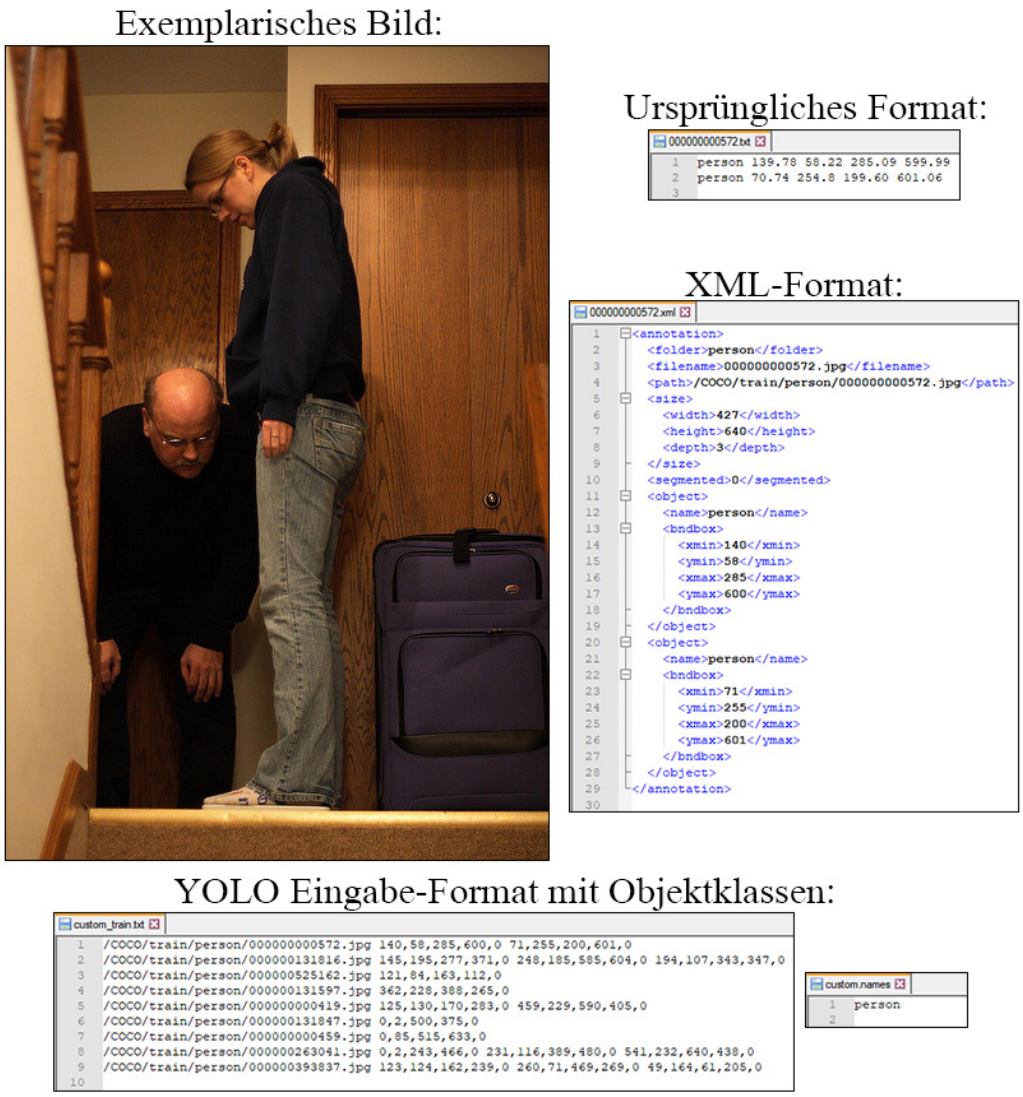


Abbildung 4.2: Darstellung der Formate anhand eines Beispielbildes.

Training:

Es besteht die Möglichkeit für das Training ein vortrainiertes Modell zu verwenden, sodass es sich dabei um einen Transfer Learning Prozess handelt. Hierfür werden die YOLO bzw. Tiny-YOLO Gewichte von der offiziellen YOLO Website heruntergeladen, diese werden von Redmon et al. bereitgestellt.

Für das Training lassen sich die YOLO Version, die Eingabegröße, die Batch Size, die zu verwendenden Anchors, die Anzahl der Epochen und die Lernrate festlegen. Zusätzlich ist es möglich für das Training die Tiny-YOLO Variante der ausgewählten YOLO Version

zu verwenden. Über einen Parameter lässt es sich steuern, ob eine Data Augmentation während des Trainings stattfinden soll. Diese beinhaltet eine zufällige horizontale Drehung, ein zufälliges Zuschneiden und eine zufällige Drehung des Bildes. Außerdem wurde ein Early Stopping Mechanismus entwickelt, sodass das Training vorzeitig gestoppt wird, sobald innerhalb der festgelegten Anzahl der Epochen kein Trainingsfortschritt vorhanden ist. Zum Abschluss des Trainings wird außerdem eine Grafik über den Verlauf des Train- und Test-Losses über die Trainingsepochen erstellt.

Evaluation (Objekterkennung):
Für die Evaluation der Objekterkennung wurde das von Cartucho entwickelte Tool[2] verwendet. Dieses Tool evaluiert das trainierte Modell anhand der Vorhersagen des Modells und der Ground-Truth Daten des Test-Datensatzes und liefert eine Aussage darüber, wie hoch der mAP ist. Außerdem zeigt das Tool, wie viele True Positives und False Positives vorhergesagt und wie viele Objekte sich insgesamt in dem Test-Datensatz befinden und es wird eine Grafik mit einer Precision-Recall Kurve erstellt.

Optional ist es möglich, dass sowohl die Ground-Truth-Bounding Boxes (blau), als auch die vorhergesagten Bounding Boxes (grün / rot) in die Bilder des Test-Datensatzes eingezeichnet werden. Die Farbe der vorhergesagten Bounding Boxes signalisiert, wie gut die Vorhersage ist. Bei einem IoU von 0.5 zur Ground-Truth-Bounding Box werden die vorhergesagten Bounding Boxes grün dargestellt, anderenfalls werden sie rot dargestellt. Die Bounding Boxes werden auf zwei unterschiedliche Weisen in die Bilder eingezeichnet. Einerseits werden alle vorhergesagten Bounding Boxes und die Bounding Boxes aus dem Ground Truth eingezeichnet. Andererseits werden pro Bild nur eine vorhergesagte Bounding Box mit der dazugehörigen Ground-Truth-Bounding Box eingezeichnet. Diese Bilder beinhalten zusätzlich Informationen über den IoU und die Genauigkeit der richtigen Klassifizierung. Diese Bilder befinden sich in einem separaten Ordner und sind der Genauigkeit nach absteigend sortiert.

Evaluation (Objektverfolgung):
Für die Evaluation der Objektverfolgung wurde das von Luiten entwickelte Tool[3] verwendet. Das Tool ermöglicht die Evaluation der Objektverfolgung anhand verschiedener Metriken (z.B. HOTA). Zusätzlich unterstützt das Tool die Ausführung der Metriken anhand von bestehenden Tracking-Benchmarks (z.B. MOT Challenge).

[2] https://github.com/Cartucho/mAP
[3] https://github.com/JonathonLuiten/TrackEval

Objekterkennung:
Anhand des trainierten Modells ist eine Objekterkennung möglich. Hierfür wird das Modell geladen, sodass unter Eingabe eines Bildes die Bounding Boxes, Objektklassen und die Konfidenzwerte ermittelt und eingezeichnet werden.

Objektverfolgung:
Das System beinhaltet die von Wojke entwickelte DeepSORT Implementierung[4]. Der Deep Appearance Descriptor basiert auf einem Modell, welches mit dem MARS Datensatz trainiert wurde, sodass er primär zur Wiedererkennung von Personen genutzt werden kann.

Das System ermöglicht eine Objektverfolgung anhand eines Videos oder der Bilder, die von einer Webcam aufgenommen werden. Es werden die Bounding Boxes um die vorhergesagten Objekte, die jeweiligen Objektklassen und die jeweilige ID des Objektes ausgegeben. Zusätzlich werden die FPS während der Objektverfolgung dargestellt. Optional lässt sich der Konfidenzwert der YOLO-Vorhersage darstellen.

4.2 Tools zur Anpassung der Datensätze

4.2.1 Manuelles Filtern der Datensätze

Dieses Tool dient zum Aussortieren von schlecht beschrifteten Bildern. Hierzu werden in das jeweilige Bild die Bounding Boxes mit den dazugehörigen Objektklassen eingezeichnet, sodass über eine Taste entschieden werden kann, ob es sich dabei um ein gut beschriftetes oder schlecht beschriftetes Bild handelt. Zusätzlich ist es möglich ungeeignete Bilder komplett auszusortieren, die nicht für das Training verwendet werden sollen.

Anhand dieser Entscheidung wird das Bild mit den dazugehörigen Beschriftungen in den jeweiligen Ordner verschoben, sodass im nächsten Schritt der manuellen Beschriftung nur die schlecht beschrifteten Bilder berücksichtigt werden müssen.

Falls die letzte Entscheidung zu voreilig getroffen wurde, ist möglich, diese zu widerrufen. Dabei wird die Verschiebung des letzten Bildes mit der zugehörigen Beschriftung wieder rückgängig gemacht.

[4] https://github.com/nwojke/deep_sort

4.2.2 Manuelle Beschriftung der Datensätze

Das von Tzutalin entwickelte Tool labelImg[5] ermöglicht das händische Einzeichnen von Bounding Boxes in ein Bild und der Zuordnung einer dazugehörigen Objektklasse.

Zu Beginn müssen das Verzeichnis, in dem sich die Bilder befinden und das Verzeichnis, in das die Beschriftungen gespeichert werden, ausgewählt werden. Das Tool speichert die Beschriftungen im PASCAL VOC, YOLO oder CreateML Format. Da das ursprüngliche Format des COCO und Open Image Datensatzes nicht erkannt wird, wird es zunächst ins PASCAL VOC Format transformiert. Ein Screenshot des Tools ist in Abbildung 4.3 dargestellt.

Abbildung 4.3: Screenshot des labelImg Tools.

Mit Hilfe des Tools lassen sich Bounding Boxes einzeichnen, löschen oder die Größen der bereits bestehenden Bounding Boxes anpassen. Bereits vorhandene Bounding Boxes aus den Datensätzen werden in dem Tool dargestellt, sodass der Aufwand bei kleinen Anpassungen reduziert wird.

[5] https://github.com/tzutalin/labelImg

4.2.3 Vervielfältigung der Daten

Die Bibliothek Albumentations[6] ermöglicht eine Vielzahl von Transformationsmöglichkeiten von Bildern. Der Zweck der Transformationen besteht darin, neue Bilddaten aus den vorhandenen Daten zu erstellen, um trainierte Modelle zu verbessern.

Albumentations unterscheidet zwischen Transformationen auf Pixelebene und Transformationen auf räumlicher Ebene. Bei den Transformationen auf räumlicher Ebene werden gleichzeitig sowohl das Eingabebild als auch die Ziele, wie Masken, Bounding Boxes oder Keypoints verändert (z.B. CenterCrop, HorizontalFlip oder Rotate). Bei den Transformationen wird ausschließlich das Eingabebild verändert (z.B. GaussNoise, GaussianBlur oder Sharpen).

Insgesamt stehen 44 Transformationen auf Pixelebene und 35 Transformationen auf räumlicher Ebene zur Verfügung. Die Transformationen lassen sich über Parameter individuell anpassen, zusätzlich wird für jede Transformationen ein Wahrscheinlichkeitswert festgelegt, sodass die jeweilige Transformation nur mit dieser Wahrscheinlichkeit ausgeführt wird. Das dient dazu, die Erstellung identischer neuer Bilder zu verhindern.

Zu beachten ist, dass nicht jede Transformation auf räumlicher Ebene für alle Arten der Beschriftung geeignet ist. Anhand einer Tabelle des offiziellen Github Repositories[7] ist dargestellt, welche Transformationen für Bilder, Masken, Bounding Boxes und Keypoints geeignet sind. In Abbildung 4.4 sind das Eingabebild und fünf Bilder dargestellt, die durch Transformationen erstellt wurden. Es wurden hierfür sowohl Transformationen auf Pixelebene als auch Transformationen auf räumlicher Ebene angewendet.

[6] https://albumentations.ai/
[7] https://github.com/albumentations-team/albumentations

(a) Eingabebild. (b) Transformationen 1. (c) Transformationen 2.

(d) Transformationen 3. (e) Transformationen 4. (f) Transformationen 5.

Abbildung 4.4: Eingabebild mit exemplarischen Transformationen.

5 Durchführung

5.1 Anpassung der Datensätze

Durch den Einsatz der zuvor beschriebenen Tools wurden sowohl der COCO, als auch der Open Images Datensatz angepasst. Zunächst wurden aus den jeweiligen Datensätzen 1000 Trainingsbilder und 500 Testbilder (kleiner Datensatz) und 10000 Trainingsbilder und 500 Testbilder (großer Datensatz) der Klasse "Person" heruntergeladen.

Für die Anpassung der Datensätze wurden jeweils die kleinen Datensätze verwendet. Diese Bilder wurden im ersten Schritt in drei Kategorien zugeordnet: Akzeptiert, verworfen und vollständig verworfen. Bilder, die reale Personen beinhalten und diese korrekt mit Bounding Boxes umrandet sind wurden akzeptiert, falls nicht alle Personen korrekt mit Bounding Boxes umrandet waren, wurden diese verworfen und falls sich keine realen Personen oder zu viele Personen innerhalb eines Bildes befinden, wurden diese vollständig verworfen.

Die verworfenen Bilder wurden im nächsten Schritt händisch beschriftet und die vollständig verworfenen Bilder durch geeignete Bilder aus den jeweiligen großen Datensätzen ersetzt. Nach diesen Schritten wurden alle geeigneten Bilder zusammengeführt, sodass zum Abschluss dieser Schritte angepasste COCO und Open Images Datensätze entstanden sind, die ausschließlich aus gut beschrifteten Bildern bestehen.

Die Trainingsbilder wurden im nächsten Schritt vervielfältigt, indem aus jedem Bild mittels Data Augmentation neun weitere Bilder erstellt wurden. In Tabelle 5.1 sind die verwendeten Transformationen dargestellt. Für jede Transformation wurde eine Wahrscheinlichkeit von 0.2 festgelegt, dass diese auf ein Bild angewendet wird.

Tabelle 5.1: Verwendete Transformationen.

Transformation	Beschreibung
Blur	Weichzeichnen mit einem zufälligen Kernel
Emboss	Prägt das Eingabebild und überlagert das Ergebnis mit dem Originalbild
Equalize	Angleichen des Bildhistogramms
GaussNoise	Anwendung eines Gaußschen Rauschens
GaussianBlur	Weichzeichnen mit einem Gauß-Filter mit einem zufälligen Kernel
HueSaturationValue	Zufällige Änderung des Farbtons und der Sättigung
ImageCompression	Erhöhung der JPEG- und WebP-Komprimierung des Bildes
MultiplicativeNoise	Multiplizieren des Bildes mit einer Zufallszahl
Posterize	Reduzierung der Anzahl der Bits für jeden Farbkanal
RGBShift	Zufällige Verschiebung der Werte für jeden Kanal des RGB-Eingabebildes
RandomBrightnessContrast	Zufällige Änderung der Helligkeit und des Kontrast
RandomFog	Simuliert Nebel im Bild
RandomRain	Fügt Regen-Effekte hinzu
RandomToneCurve	Zufällige Änderung des Verhältnis zwischen hellen und dunklen Bereichen des Bildes durch Manipulation der Tonwertkurve
Sharpen	Schärft das Eingabebild und überlagert das Ergebnis mit dem Originalbild
ToGray	Konvertiert das eingegebene RGB-Bild in ein graustufiges Bild
HorizontalFlip	Spiegelt die Eingabe horizontal um die y-Achse
RandomScale	Zufällige Veränderung der Größe der Eingabe

Nach Abschluss aller Schritte standen insgesamt zehn Varianten der Datensätze zur Verfügung. Diese sind in Tabelle 5.2 aufgeführt. Diese wurden anschließend für das Training der neuronalen Netzwerke verwendet.

Tabelle 5.2: Alle Varianten der beiden Datensätze.

Datensatz Variante		Beschreibung
COCO	C1 (Train:1000u, Test:500u)	Kleiner COCO Datensatz ohne Anpassungen
	C2 (Train:1000u, Test:500m)	Kleiner COCO Datensatz mit angepasstem Test-Datensatz
	C3 (Train:10000u, Test:500m)	Großer COCO Datensatz mit angepasstem Test-Datensatz
	C4 (Train:1000m, Test:500m)	Kleiner COCO Datensatz mit angepasstem Trainings- und Test-Datensatz
	C5 (Train:10000m, Test:500m)	Großer COCO Datensatz, der aus dem kleinen Datensatz entstanden ist mit angepasstem Test-Datensatz
Open Images	O1 (Train:1000u, Test:500u)	Kleiner Open Images Datensatz ohne Anpassungen
	O2 (Train:1000u, Test:500m)	Kleiner Open Images Datensatz mit angepasstem Test-Datensatz
	O3 (Train:10000u, Test:500m)	Großer Open Images Datensatz mit angepasstem Test-Datensatz
	O4 (Train:1000m, Test:500m)	Kleiner Open Images Datensatz mit angepasstem Trainings- und Test-Datensatz
	O5 (Train:10000m, Test:500m)	Großer Open Images Datensatz, der aus dem kleinen Datensatz entstanden ist mit angepasstem Test-Datensatz

5.2 Training

Die neuronalen Netzwerke wurden auf dem JupyterHub[1] der HAW trainiert. Das Trainingssystem besteht aus einem 4 Kern Prozessor, einer NVIDIA Tesla V100 16 GB HBMS Grafikkarte und 16 GB Arbeitsspeicher. Trainiert wurde zudem mit der TensorFlow 2.6 Version.

Es wurden jeweils pro Datensatz Variante zwei neuronale Netzwerke (YOLOv3 und Tiny-YOLOv3) trainiert. Die Trainingsdauer wurde auf 30 Epochen festgelegt. Für die Trainingsprozesse wurden die Gewichte der vortrainierten YOLOv3 und Tiny-YOLOv3 Modelle aus der offiziellen YOLO Website[2] hinzugezogen, sodass es sich um einen Transfer Learning Prozess handelt.

[1] https://jupyterhub.informatik.haw-hamburg.de
[2] https://pjreddie.com/darknet/yolo/

5.3 Evaluation der Objekterkennung

Die Objekterkennung wurde anhand des zuvor vorgestellten Tools[3] evaluiert. Zunächst wurden die Modelle aller Datensatz Varianten je nach Datensatzursprung separat anhand des YOLOv3 und Tiny-YOLOv3 Variante evaluiert. Anschließend wurden die Modelle anhand eines zusammengeführten Test-Datensatzes evaluiert. Dieses besteht aus der Zusammenführung des modifizierten COCO Test-Datensatzes und des modifizierten Open Images Test-Datensatzes.

5.4 Evaluation der Objektverfolgung

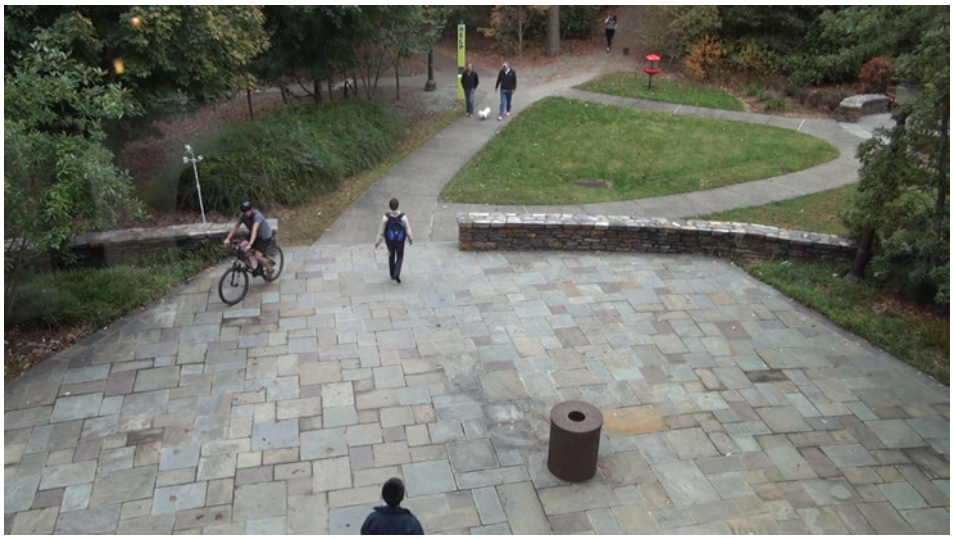
Die Objektverfolgung wurde anhand des zuvor vorgestellten Tools[4] durchgeführt. Die Evaluation wurde anhand von zwei Videos durchgeführt. Die Videos stammen aus dem Multiview Object Tracking Dataset[5]. Für die Evaluation wurden die Videos mit dem MoviePy Tool[6] aufgrund der Anzahl der zu evaluierenden Modelle auf 10 Sekunden gekürzt.

Der view-HC4 Videoausschnitt hat eine Auflösung von 1920 x 1080 mit 30 FPS und beinhaltet Personen, die sich teilweise zu Fuß, als auch mit einem Fahrrad auf einem Gehweg fortbewegen. Die Kamera befindet sich auf einer bestimmten Höhe, sodass die Personen von oben aufgezeichnet werden. Insgesamt befinden sich 6 unterschiedliche Personen innerhalb des Videoausschnittes. Der view-IP1 Videoausschnitt hat ebenfalls eine Auflösung von 1920 x 1080 mit 30 FPS und beinhaltet Personen, die sich in einem Park aufhalten. Die Personen durchqueren den Park zu Fuß oder mit einem Fahrrad oder spielen mit einem Ball. Die Kamera befindet sich auf der Höhe der Personen und ist direkt auf die Personen gerichtet. Insgesamt befinden sich ebenfalls 6 unterschiedliche Personen innerhalb des Videoausschnittes, wobei diese im Vergleich zu den Personen aus dem view-HC4 Videoausschnitt größer dargestellt sind.

Die Ground Truth Daten der Videos mussten für die Evaluation aus dem bestehenden Format in das Format der MOT Challange transformiert werden. Außerdem mussten die

[3] https://github.com/Cartucho/mAP
[4] https://github.com/JonathonLuiten/TrackEval
[5] https://bitbucket.org/merayxu/multiview-object-tracking-dataset/src/master/
[6] https://zulko.github.io/moviepy/

Daten aufgrund der Kürzung der Videos zusätzlich angepasst werden. In Abbildung 5.1 ist jeweils ein Einzelbild des jeweiligen Videos dargestellt.



(a) Einzelbild des view-HC4 Videos.

(b) Einzelbild des view-IP1 Videos.

Abbildung 5.1: Einzelbilder des view-HC4 und view-IP1 Videos.

5.5 Performanceanalyse

Für die Performanceanalyse eines YOLOv3 Modells und eines Tiny-YOLOv3 Modells wurden Objekterkennungen und Objektverfolgungen auf drei Geräten durchgeführt und die Ergebnisse anschließend miteinander verglichen.

Diese wurden sowohl auf der CPU, als auch auf der GPU des Desktop PCs und des Notebooks durchgeführt. Auf dem Raspberry Pi wurde hierfür nur die CPU verwendet, da der Pi keine zusätzliche GPU besitzt. Für die Analysen wurde für beide YOLO-Varianten jeweils das C3 (COCO Train:10000u, Test:500m) Modell verwendet.

Für die Performanceanalyse der Objekterkennung wurden Objekterkennungen auf die Bilder des kombinierten COCO und Open Images Testdatensatzes (1000 Bilder) durchgeführt, die jeweilige Ausführungszeit erfasst und anschließend der Durchschnitt der Ausführungszeiten gebildet.

Für die Performanceanalyse der Objektverfolgungen wurden Objektverfolgungen auf die zwei Videos angewendet, die für die Evaluation bereits verwendet wurden (view-HC4 und view-IP1), die FPS während der Objektverfolgung erfasst und der Durchschnitt der FPS während der Ausführung der Objektverfolgung beider Videos gebildet.

In Tabelle 5.3 ist die Hardware der Systeme aufgeführt, welche für die Performanceanalyse verwendet wurden.

Tabelle 5.3: Hardware für die Performanceanalyse.

System	CPU	GPU	RAM
Desktop PC	Intel Core i7-6700K 4 x 4 GHz	Nvidia GeForce 980Ti 6 GB	16 GB
Notebook	Intel Core i7-7700HQ 4 x 3.8 GHz	Nvidia GeForce 940MX 2 GB	16 GB
Raspberry Pi 3 Model B	ARMv8 4 x 1.2 GHz	-	1 GB

6 Ergebnisse und Diskussion

6.1 Ergebnisse der Anpassung der Datensätze

6.1.1 Auffälligkeiten der Datensätze

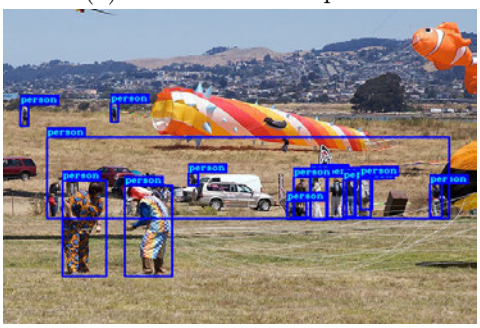
Folgende Merkmale sind bei der Filterung der Datensätze aufgefallen, die dazu geführt haben, dass diese Bilder verworfen bzw. vollständig verworfen wurden:

- Nicht alle Personen wurden mit Bounding Boxes umrandet
- Gruppen von Personen wurden als eine Person klassifiziert
- Personen wurden nicht vollständig mit Bounding Boxes erfasst
- Objekte wurden fälschlicherweise als Personen klassifiziert
- Es befanden sich zu viele Personen in einem Bild (z.B. Tribüne eines Stadions)
- Es befanden sich keine realen Personen im Bild (z.B. Personen in Videospielen)
- Aufgrund der Bildqualität und der Größe von Objekten konnten Personen nicht eindeutig identifiziert werden
- Es war nicht eindeutig, ob es sich um eine Person handelt (z.B. Schuh am Rand eines Bildes)

Weitere Auffälligkeiten waren, dass teilweise innerhalb eines Bildes Personen im Vordergrund nicht klassifiziert und Personen im Hintergrund klassifiziert wurden. Außerdem waren die Klassifizierungen nicht einheitlich, da nicht reale Personen teilweise mit einer Bounding Box umrandet waren und teilweise nicht mit Bounding Box umrandet waren. Dies trifft ebenfalls auf Spiegelbilder von Personen zu. In Abbildung 6.1 sind akzeptierte, verworfen und vollständig verworfen Bilder exemplarisch dargestellt.

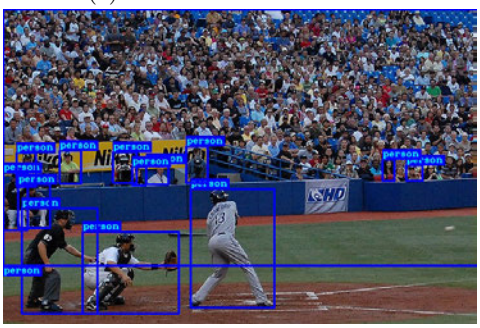
(a) COCO Bild akzeptiert.

(b) Open Images Bild akzeptiert.

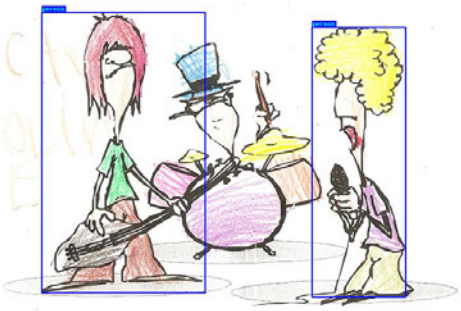


(c) COCO Bild verworfen.

(d) Open Images Bild verworfen.



(e) COCO Bild vollständig verworfen.



(f) Open Images Bild vollständig verworfen.

Abbildung 6.1: Exemplarische COCO- und Open Images Bilder, die akzeptiert, verworfen und vollständig verworfen wurden.

Abbildung 6.1a und 6.1b wurden akzeptiert, da alle Personen, die sich auf den Bildern befinden korrekt mit Bounding Boxes umrandet sind. Abbildung 6.1c und 6.1d wurden verworfen, da in den Bildern nicht alle Personen mit einer Bounding Box umrandet sind. Zusätzlich wurde eine Gruppe von Personen mit einer großen Bounding Box umrandet. Abbildung 6.1e wurde vollständig verworfen, da sich zu viele Personen in dem Bild befinden. Außerdem wurde ebenfalls eine Gruppe von Personen mit einer großen Bounding Box umrandet. Abbildung 6.1f wurde vollständig verworfen, da sich keine realen Personen in dem Bild befinden.

6.1.2 Ergebnisse der Filterung der Datensätze

Die Anzahl der akzeptierten, verworfenen und vollständig verworfenen Bilder ist in Tabelle 6.1 dargestellt.

Tabelle 6.1: Anzahl der akzeptierten, verworfen und vollständig verworfenen Bilder.

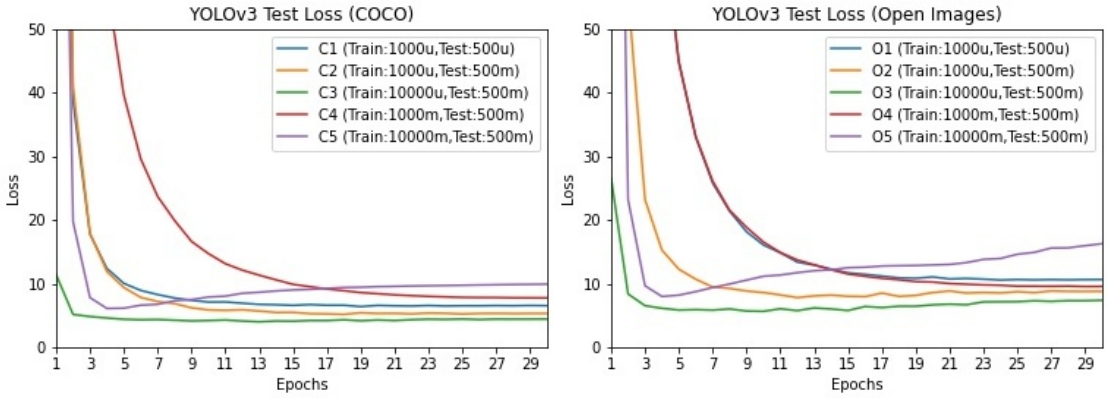
Datensatz		Akzeptierte Bilder	Verworfene Bilder	Vollständig verworfene Bilder
COCO	Train	796 (80%)	154 (15%)	50 (5%)
	Test	376 (75%)	71 (14%)	53 (11%)
	Gesamt	1172 (78%)	225 (15%)	103 (7%)
Open Images	Train	481 (48%)	271 (27%)	248 (25%)
	Test	203 (40%)	199 (40%)	98 (20%)
	Gesamt	684 (46%)	470 (31%)	346 (23%)

Beim COCO Datensatz wurden insgesamt 1172 Bilder akzeptiert, 225 Bilder verworfen und 103 Bilder vollständig verworfen. Beim Open Images Datensatz wurden dagegen 684 Bilder akzeptiert, 470 Bilder verworfen und 346 Bilder vollständig verworfen. Insgesamt mussten Beim COCO Datensatz 328 Bilder und beim Open Images Datensatz 816 Bilder bearbeitet bzw. ersetzt werden, somit wurden beim Open Images Datensatz 488 Bilder weniger akzeptiert als beim COCO Datensatz.

6.2 Trainingsverläufe

Die Trainingsverläufe wurden anhand des Test-Losses während des Trainings grafisch dargestellt. Dafür befinden sich alle Datensatz Varianten in einer Abbildung und die Loss-Verläufe sind außerdem in YOLOv3 und Tiny-YOLOv3 unterteilt.

In Abbildung 6.2 sind die Test-Loss-Werte anhand des YOLOv3-Trainings mit den COCO und Open Images Datensatz Varianten dargestellt.



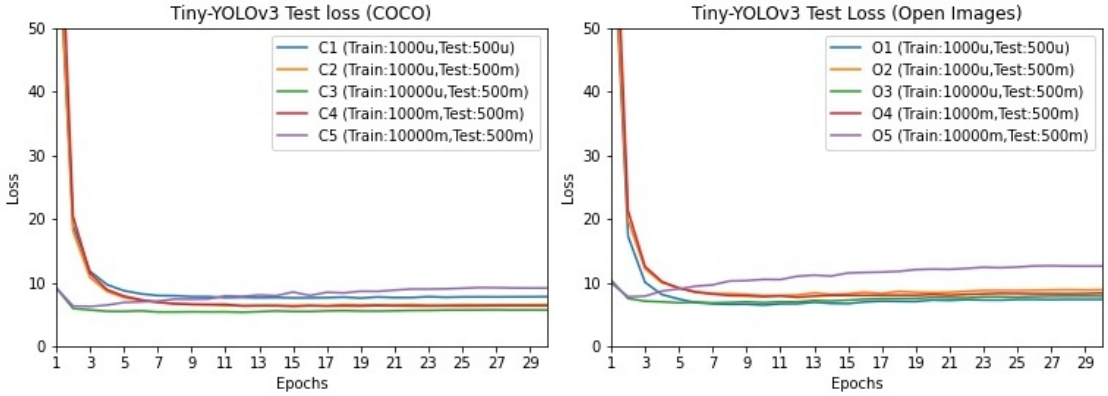
(a) YOLOv3 Test Loss (COCO). (b) YOLOv3 Test Loss (Open Images).

Abbildung 6.2: YOLOv3 Test Losses der COCO und Open Images Datensatz Varianten.

In Abbildung 6.2a beginnen C1, C2, C4 und C5 initial bei einem hohen Loss Wert, C3 dagegen bei einem Loss von 11.4. Die C1, C2, C4 und C5 Loss Kurven sinken zu Beginn stark ab, der C3 Loss sinkt weniger stark ab. Die C1, C2, C3 und C4 Loss-Werte befinden sich nach 30 Epochen zwischen 4.5 und 7.5 und der C5 Loss-Wert bei 9.9. Auffällig ist, dass der Loss von C5 im Vergleich zu den anderen Loss Verläufen stärker ansteigt.

In Abbildung 6.2b beginnen O1, O2, O4 und O5 initial bei einem hohen Loss Wert, O3 dagegen bei einem Loss von 26.3. Die O1, O2, O4 und O5 Loss Kurven sinken zu Beginn stark ab, der O3 Loss sinkt weniger stark ab. Die O1, O2, O3 und O4 Loss-Werte befinden sich nach 30 Epochen zwischen 7.3 und 10.6 und der O5 Loss-Wert bei 16.2. Sichtbar ist, dass der Loss von O5 im Vergleich zu den anderen Loss Verläufen stärker ansteigt.

In Abbildung 6.3 sind die Test-Loss-Werte anhand des Tiny-YOLOv3-Trainings mit den COCO und Open Images Datensatz Varianten dargestellt.



(a) Tiny-YOLOv3 Test Loss (COCO). (b) Tiny-YOLOv3 Test Loss (Open Images).

Abbildung 6.3: Tiny-YOLOv3 Test Losses der COCO und Open Images Datensatz Varianten.

In Abbildung 6.3a beginnen C1, C2 und C4 initial bei einem hohen Loss Wert, C3 und C5 dagegen bei einem Loss von 9.1 und 9.2. Die C1, C2 und C4 Loss Kurven sinken zu Beginn stark ab, der C3 und C5 Loss sinkt weniger stark ab. Die C1, C2, C3 und C4 Loss-Werte befinden sich nach 30 Epochen zwischen 5.7 und 7.8 und der C5 Loss-Wert bei 9.1. Auffällig ist, dass der Loss von C5 im Vergleich zu den anderen Loss Verläufen stärker ansteigt.

In Abbildung 6.3b beginnen O1, O2 und O4 initial bei einem hohen Loss Wert, O3 und O5 dagegen bei einem Loss von 9.9 und 10.2. Die O1, O2 und O4 Loss Kurven sinken zu Beginn stark ab, der O3 und O5 Loss sinkt weniger stark ab. Die O1, O2, O3 und O4 Loss Loss-Werte befinden sich nach 30 Epochen zwischen 7.3 und 8.9 und der O5 Loss-Wert bei 12.6. Erkennbar ist, dass der Loss von O5 im Vergleich zu den anderen Loss Verläufen stärker ansteigt.

Insgesamt fällt auf, dass allgemein der Loss der O5 Varianten schneller ansteigt als der Loss der C5 Varianten.

6.3 Ergebnisse der Objekterkennung

In den folgenden Tabellen sind die Evaluationen der Objekterkennung anhand der Datensatz Varianten abgebildet. Die Tabellen beinhalten den mAP-Wert, die Anzahl der True Positive (TP)s und False Positive (FP)s, die Anzahl der Erkennungen und die Gesamtanzahl der Objekte, die als "Person" klassifiziert sind.

In Tabelle 6.2 sind die Ergebnisse der YOLOv3 Modelle anhand der COCO Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für C1 der unmodifizierte und für C2, C3, C4 und C5 der modifizierte COCO Testdatensatz verwendet (500 Testbilder).

Tabelle 6.2: YOLOv3 mit den COCO Datensatz Varianten im Vergleich.

Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
COCO	C1 (Train:1000u, Test:500u)	54.80%	1298 (45%)	1596 (55%)	2894	2078
	C2 (Train:1000u, Test:500m)	60.28%	1171 (56%)	934 (44%)	2105	1794
	C3 (Train:10000u, Test:500m)	**66.73%**	1286 (59%)	893 (41%)	2179	1794
	C4 (Train:1000m, Test:500m)	58.11%	1148 (55%)	951 (45%)	2099	1794
	C5 (Train:10000m, Test:500m)	47.44%	939 (66%)	487 (34%)	1426	1794

C3 erreichte mit 66.73% den höchsten und C5 mit 47.44% den niedrigsten mAP-Wert. C1 hat mit 1298 die meisten und C5 mit 939 die wenigsten True Positives vorhergesagt. C5 hat mit 487 die wenigsten und C1 mit 1596 die meisten False Positives vorhergesagt. Insgesamt hat C1 mit 2894 die höchste und C5 mit 1426 die niedrigste Anzahl an Vorhersagen getätigt. C1, C2, C3 und C4 haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind. C5 hat insgesamt weniger Objekte vorhergesagt als vorhanden gewesen sind.

In Tabelle 6.3 sind die Ergebnisse der YOLOv3 Modelle anhand der Open Images Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für O1 der unmodifizierte und für O2, O3, O4 und O5 der modifizierte Open Images Testdatensatz verwendet (500 Testbilder).

Tabelle 6.3: YOLOv3 mit den Open Images Datensatz Varianten im Vergleich.

Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
Open Images	O1 (Train:1000u, Test:500u)	16.35%	495 (25%)	1488 (75%)	1983	1268
	O2 (Train:1000u, Test:500m)	42.45%	1038 (51%)	987 (49%)	2025	2061
	O3 (Train:10000u, Test:500m)	**55.39%**	1285 (56%)	1010 (44%)	2295	2061
	O4 (Train:1000m, Test:500m)	51.61%	1204 (53%)	1057 (47%)	2261	2061
	O5 (Train:10000m, Test:500m)	40.61%	968 (63%)	560 (37%)	1528	2061

O3 erreichte mit 55.39% den höchsten und O1 mit 16.35% den niedrigsten mAP-Wert. O3 hat mit 1285 die meisten und O1 mit 495 die wenigsten True Positives vorhergesagt. O5 hat mit 560 die wenigsten und O1 mit 1488 die meisten False Positives vorhergesagt. Insgesamt hat O3 mit 2295 die höchste und O5 mit 1528 die niedrigste Anzahl an Vorhersagen getätigt. O1, O3 und O4 haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind. O2 und O5 haben insgesamt weniger Objekte vorhergesagt als vorhanden gewesen sind.

In Tabelle 6.4 sind die Ergebnisse der Tiny-YOLOv3 Modelle anhand der COCO Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für C1 der unmodifizierte und für C2, C3, C4 und C5 der modifizierte COCO Testdatensatz verwendet (500 Testbilder).

Tabelle 6.4: Tiny-YOLOv3 mit den COCO Datensatz Varianten im Vergleich.

Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
COCO	C1 (Train:1000u, Test:500u)	33.20%	875 (28%)	2214 (72%)	3089	2078
	C2 (Train:1000u, Test:500m)	36.44%	801 (36%)	1424 (64%)	2225	1794
	C3 (Train:10000u, Test:500m)	**45.87%**	950 (40%)	1400 (60%)	2350	1794
	C4 (Train:1000m, Test:500m)	37.09%	833 (30%)	1981 (70%)	2814	1794
	C5 (Train:10000m, Test:500m)	36.16%	822 (31%)	1861 (69%)	2683	1794

C3 erreichte mit 45.87% den höchsten und C1 mit 33.20% den niedrigsten mAP-Wert. C3 hat mit 950 die meisten und C2 mit 801 die wenigsten True Positives vorhergesagt. C3 hat mit 1400 die wenigsten und C1 mit 2214 die meisten False Positives vorhergesagt. Insgesamt hat C1 mit 3089 die höchste und C2 mit 2225 die niedrigste Anzahl an Vorhersagen getätigt. Alle Modelle haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind.

In Tabelle 6.5 sind die Ergebnisse der Tiny-YOLOv3 Modelle anhand der Open Images Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für O1 der unmodifizierte und für O2, O3, O4 und O5 der modifizierte Open Images Testdatensatz verwendet (500 Testbilder).

Tabelle 6.5: Tiny-YOLOv3 mit den Open Images Datensatz Varianten im Vergleich.

Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
Open Images	O1 (Train:1000u, Test:500u)	9.42%	354 (14%)	2251 (86%)	2605	1268
	O2 (Train:1000u, Test:500m)	25.56%	752 (29%)	1852 (71%)	2604	2061
	O3 (Train:10000u, Test:500m)	**34.46%**	925 (31%)	2014 (69%)	2939	2061
	O4 (Train:1000m, Test:500m)	27.96%	830 (26%)	2355 (74%)	3185	2061
	O5 (Train:10000m, Test:500m)	26.73%	777 (31%)	1695 (69%)	2472	2061

O3 erreichte mit 34.46% den höchsten und O1 mit 9.42% den niedrigsten mAP-Wert. O3 hat mit 925 die meisten und O1 mit 354 die wenigsten True Positives vorhergesagt. O5 hat mit 1695 die wenigsten und O4 mit 2355 die meisten False Positives vorhergesagt. Insgesamt hat O4 mit 3185 die höchste und O5 mit 2472 die niedrigste Anzahl an Vorhersagen getätigt. Alle Modelle haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind.

In Tabelle 6.6 sind die Ergebnisse der YOLOv3 Modelle anhand der COCO und Open Images Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für alle Datensatz Varianten der kombinierte modifizierte COCO und Open Images Testdatensatz verwendet (1000 Testbilder).

Tabelle 6.6: YOLOv3 mit den Varianten beider Datensätze im Vergleich.

Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
COCO	C1 (Train:1000u, Test:500u)	54.92%	2285 (67%)	1146 (33%)	3431	3855
	C2 (Train:1000u, Test:500m)	53.69%	2251 (65%)	1226 (35%)	3477	3855
	C3 (Train:10000u, Test:500m)	**61.86%**	2526 (71%)	1033 (29%)	3559	3855
	C4 (Train:1000m, Test:500m)	55.09%	2365 (55%)	1953 (45%)	4318	3855
	C5 (Train:10000m, Test:500m)	42.53%	1865 (64%)	1059 (36%)	2924	3855
Open Images	O1 (Train:1000u, Test:500u)	45.26%	1927 (64%)	1069 (36%)	2996	3855
	O2 (Train:1000u, Test:500m)	45.13%	1853 (75%)	620 (25%)	2473	3855
	O3 (Train:10000u, Test:500m)	54.00%	2220 (75%)	731 (25%)	2951	3855
	O4 (Train:1000u, Test:500m)	53.76%	2297 (52%)	2080 (48%)	4377	3855
	O5 (Train:10000u, Test:500m)	44.34%	1921 (64%)	1060 (36%)	2981	3855

C3 erreichte mit 61.86% den höchsten und C5 mit 42.53% den niedrigsten mAP-Wert. C3 hat mit 2526 die meisten und O2 mit 1853 die wenigsten True Positives vorhergesagt. O2 hat mit 620 die wenigsten und O4 mit 2080 die meisten False Positives vorhergesagt. Insgesamt hat O4 mit 4377 die höchste und C2 mit 2473 die niedrigste Anzahl an Vorhersagen getätigt. C4 und O4 haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind. Die restlichen Modelle haben weniger Objekte vorhergesagt als vorhanden gewesen sind.

In Tabelle 6.7 sind die Ergebnisse der Tiny-YOLOv3 Modelle anhand der COCO und Open Images Datensatz Varianten im Vergleich dargestellt. Zur Evaluation wurde für alle Datensatz Varianten der kombinierte modifizierte COCO und Open Images Testdatensatz verwendet (1000 Testbilder).

Tabelle 6.7: Tiny-YOLOv3 mit den Varianten beider Datensätze im Vergleich.

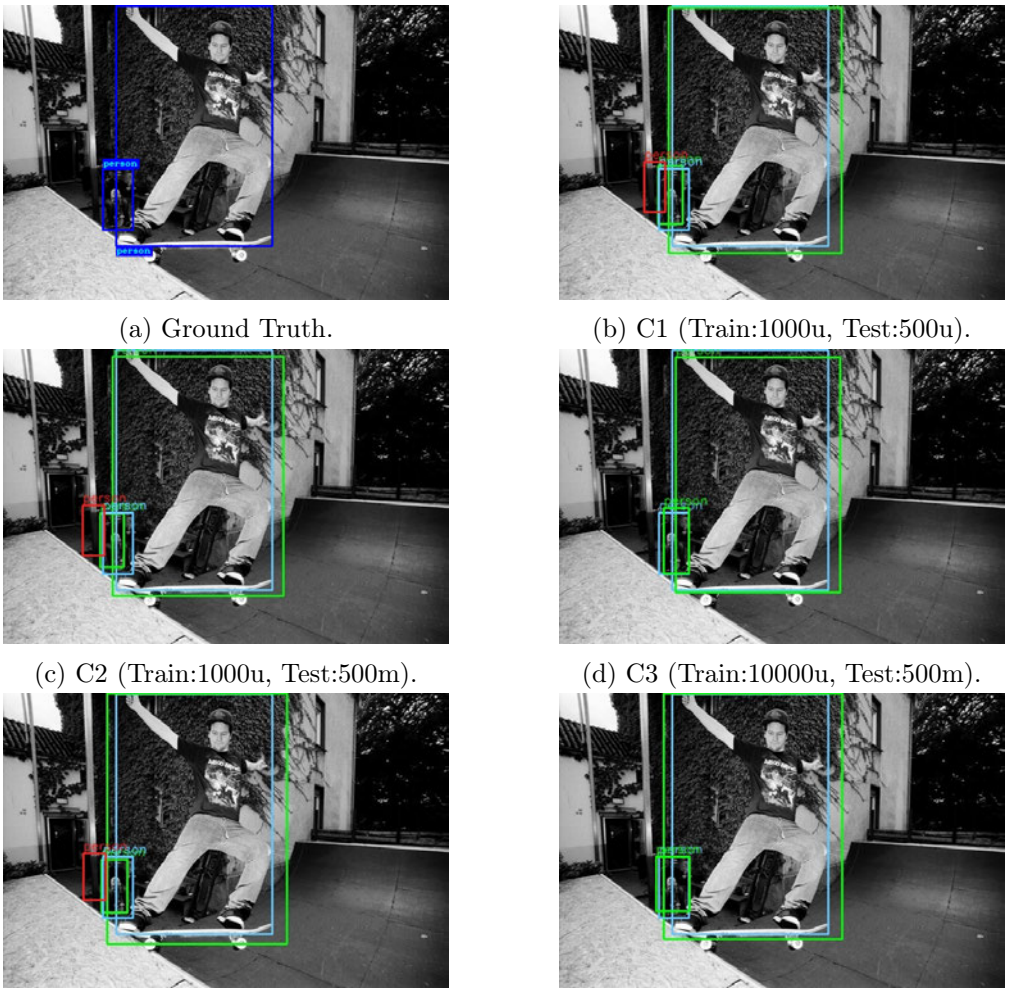
Modell der Datensatz Variante		mAP	True Positives	False Positives	Anzahl Erkennungen	Anzahl Objekte
COCO	C1 (Train:1000u, Test:500u)	32.78%	1513 (42%)	2055 (58%)	3568	3855
	C2 (Train:1000u, Test:500m)	32.72%	1525 (42%)	2128 (58%)	3653	3855
	C3 (Train:10000u, Test:500m)	**40.94%**	1852 (42%)	2547 (58%)	4399	3855
	C4 (Train:1000m, Test:500m)	33.33%	1579 (39%)	2519 (61%)	4098	3855
	C5 (Train:10000m, Test:500m)	28.27%	1277 (49%)	1335 (51%)	2612	3855
Open Images	O1 (Train:1000u, Test:500u)	29.93%	1442 (44%)	1869 (56%)	3311	3855
	O2 (Train:1000u, Test:500m)	29.77%	1413 (46%)	1630 (54%)	3043	3855
	O3 (Train:10000u, Test:500m)	37.26%	1718 (44%)	2159 (56%)	3877	3855
	O4 (Train:1000m, Test:500m)	32.53%	1578 (38%)	2627 (62%)	4205	3855
	O5 (Train:1000m, Test:500m)	29.33%	1354 (48%)	1439 (52%)	2793	3855

C3 erreichte mit 40.94% den höchsten und C5 mit 28.27% den niedrigsten mAP-Wert. C3 hat mit 1852 die meisten und C5 mit 1277 die wenigsten True Positives vorhergesagt. C5 hat mit 1335 die wenigsten und O4 mit 2627 die meisten False Positives vorhergesagt. Insgesamt hat C3 mit 4399 die höchste und C5 mit 2612 die niedrigste Anzahl an Vorhersagen getätigt. C3, C4, O3 und O4 haben insgesamt mehr Objekte vorhergesagt als vorhanden gewesen sind. Die restlichen Modelle haben weniger Objekte vorhergesagt als vorhanden gewesen sind.

Insgesamt haben die YOLOv3 Modelle einen höheren mAP-Wert im Vergleich zu den Tiny-YOLOv3 Modellen erreicht. Außerdem haben alle YOLOv3 Modelle mehr TPs als FPs vorhergesagt, bei den Tiny-YOLOv3 Modellen dagegen ist die Anzahl der TPs niedriger als die Anzahl der FPs. Auffällig ist, dass die C5 und O5 Modelle allgemein die niedrigste Anzahl von Erkennungen getätigt haben.

In den folgenden Abbildungen sind die Vorhersagen der verschiedenen Modelle anhand eines Bildes im Vergleich dargestellt. Die Abbildungen sind unterteilt in YOLOv3 und Tiny-YOLOv3 Modelle der jeweiligen COCO und Open Images Varianten.

In Abbildung 6.4 sind die YOLOv3 Vorhersagen der Modelle der COCO Varianten dargestellt.



(a) Ground Truth. (b) C1 (Train:1000u, Test:500u).

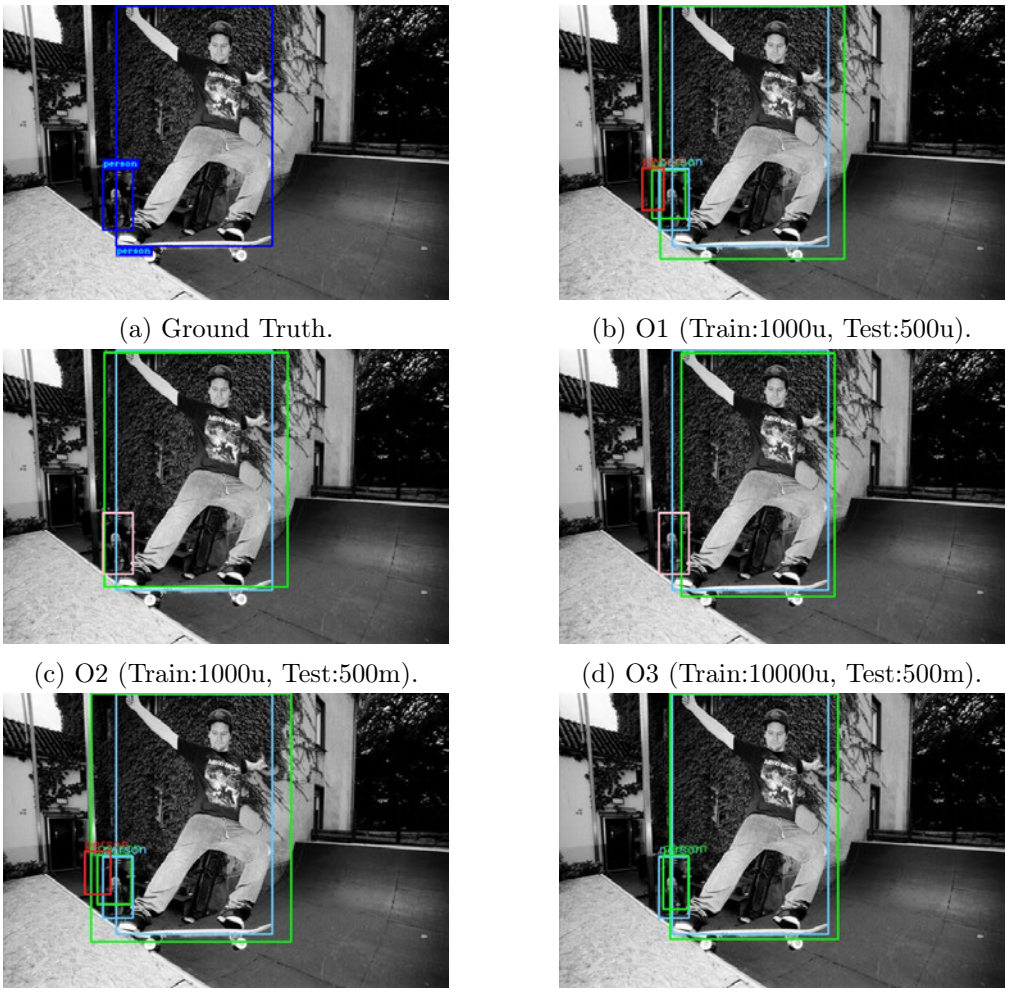
(c) C2 (Train:1000u, Test:500m). (d) C3 (Train:10000u, Test:500m).

(e) C4 (Train:1000m, Test:500m). (f) C5 (Train:10000m, Test:500m).

Abbildung 6.4: YOLOv3 Vorhersagen der Modelle der COCO Varianten.

Alle Modelle haben beide Personen korrekt erkannt, C3 und C5 haben ausschließlich die beiden Personen vorhergesagt und C1, C2 und C4 haben zusätzlich ein Objekt fälschlicherweise als Person erkannt.

In Abbildung 6.5 sind die YOLOv3 Vorhersagen der Modelle der Open Images Varianten dargestellt.



(a) Ground Truth. (b) O1 (Train:1000u, Test:500u).

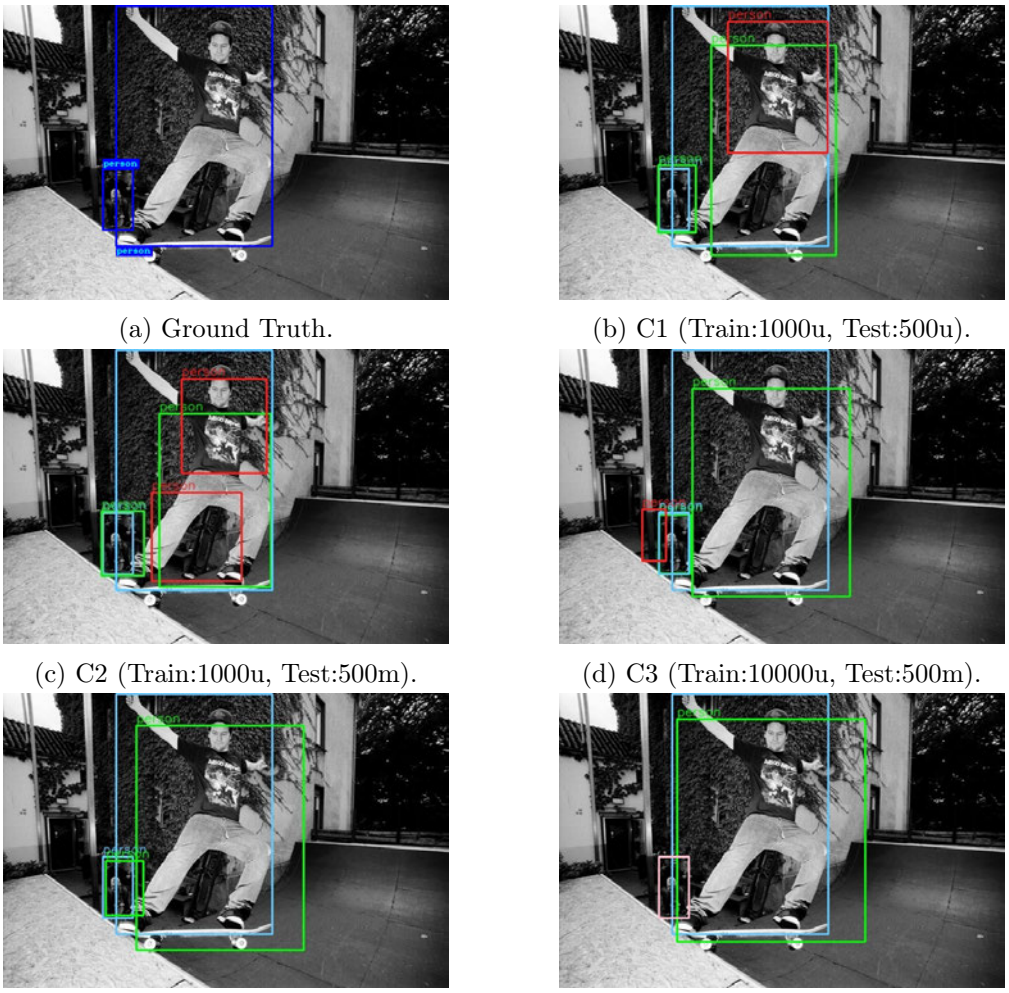
(c) O2 (Train:1000u, Test:500m). (d) O3 (Train:10000u, Test:500m).

(e) O4 (Train:1000m, Test:500m). (f) O5 (Train:10000m, Test:500m).

Abbildung 6.5: YOLOv3 Vorhersagen der Modelle der Open Images Varianten.

O1, O4 und O5 haben beide Personen korrekt erkannt, O2 und O3 haben nur die Person im Vordergrund erkannt, O5 hat ausschließlich die beiden Personen vorhergesagt und O1 und O4 haben zusätzlich ein Objekt fälschlicherweise als Person erkannt.

In Abbildung 6.6 sind die Tiny-YOLOv3 Vorhersagen der Modelle der COCO Varianten dargestellt.

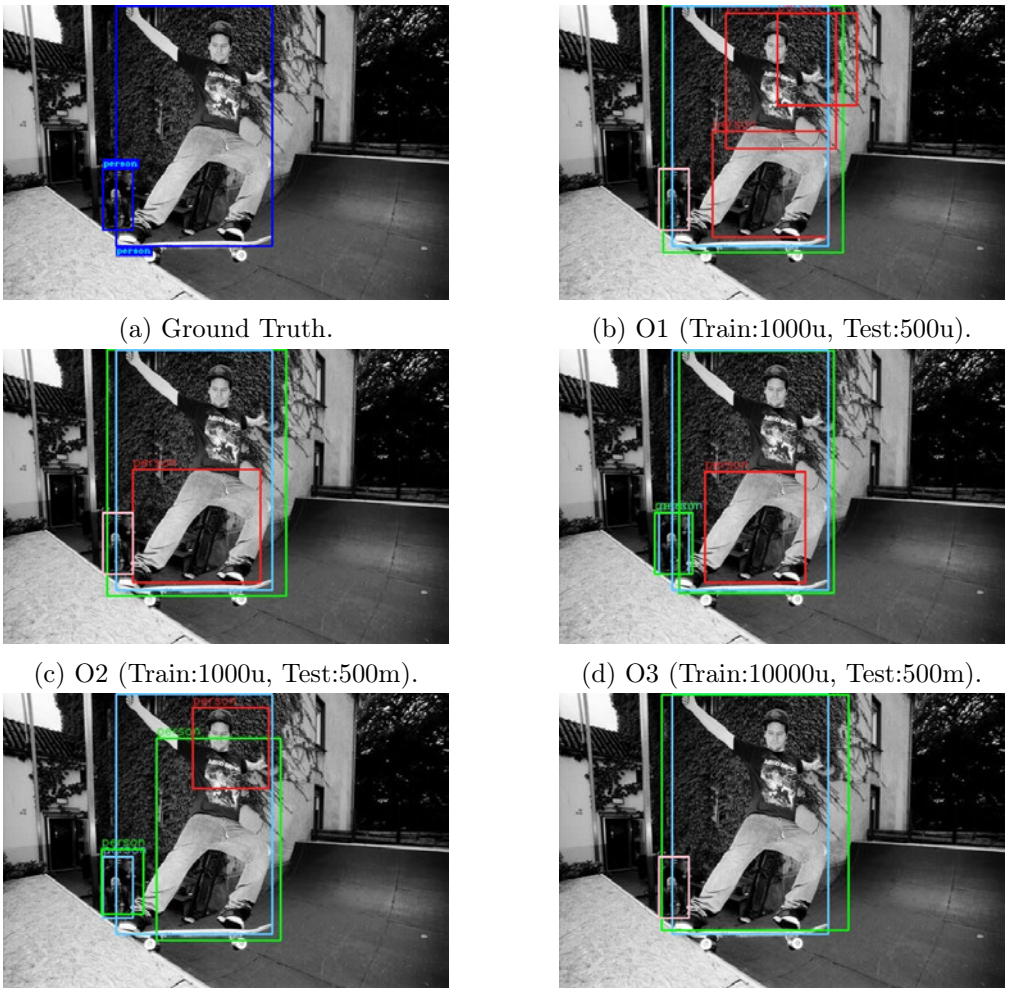


(a) Ground Truth.
(b) C1 (Train:1000u, Test:500u).
(c) C2 (Train:1000u, Test:500m).
(d) C3 (Train:10000u, Test:500m).
(e) C4 (Train:1000m, Test:500m).
(f) C5 (Train:10000m, Test:500m).

Abbildung 6.6: Tiny-YOLOv3 Vorhersagen der Modelle der COCO Varianten.

C1, C2, C3 und C4 haben beide Personen korrekt erkannt, C5 hat nur die Person im Vordergrund erkannt, C4 hat ausschließlich die beiden Personen vorhergesagt und C3 hat zusätzlich ein Objekt fälschlicherweise als Person erkannt und C1 und C2 haben die Person im Vordergrund als mehrere Personen identifiziert.

In Abbildung 6.7 sind die Tiny-YOLOv3 Vorhersagen der Modelle der Open Images Varianten dargestellt.



(a) Ground Truth. (b) O1 (Train:1000u, Test:500u).

(c) O2 (Train:1000u, Test:500m). (d) O3 (Train:10000u, Test:500m).

(e) O4 (Train:1000m, Test:500m). (f) O5 (Train:10000m, Test:500m).

Abbildung 6.7: Tiny-YOLOv3 Vorhersagen der Modelle der Open Images Varianten.

O3 und O4 haben beide Personen korrekt erkannt, O1, O2 und O5 haben nur die Person im Vordergrund erkannt, keine der Modelle hat ausschließlich die beiden Personen vorhergesagt und O1, O2, O3 und O4 haben die Person im Vordergrund als mehrere Personen identifiziert.

6.4 Ergebnisse der Objektverfolgung

In den folgenden Tabellen sind die Evaluationen der Objektverfolgung anhand von zwei Videosequenzen abgebildet. Die Tabellen beinhalten den HOTA-Wert, den DetA-Wert, den AssA-Wert und den LocA-Wert. Außerdem sind die Anzahl der IDSW, die Anzahl der MT, PT und ML und die Gesamtanzahl der zugeordneten IDs dargestellt.

In Tabelle 6.8 sind die Ergebnisse der YOLOv3 Modelle anhand des view-HC4 Videos dargestellt. Dabei wurden die COCO und Open Images Datensatz Varianten im Vergleich aufgestellt.

Tabelle 6.8: YOLOv3 mit Varianten beider Datensätze anhand des view-HC4 Videos im Vergleich.

Modell der Datensatz Variante		HOTA	DetA	AssA	LocA	IDSW	MT	PT	ML	Zugeordnete IDs
COCO	C1 (Train:1000u, Test:500u)	64.05%	66.72%	61.79%	87.02%	6	**6**	0	0	9/6
	C2 (Train:1000u, Test:500m)	49.84%	50.20%	50.12%	87.17%	11	**6**	0	0	15/6
	C3 (Train:10000u, Test:500m)	**72.92%**	**80.49%**	**66.35%**	**91.05%**	**3**	**6**	0	0	**7/6**
	C4 (Train:1000m, Test:500m)	52.82%	65.86%	43.05%	87.17%	13	**6**	0	0	14/6
	C5 (Train:10000m, Test:500m)	51.74%	61.56%	44.32%	87.42%	18	4	1	1	13/6
Open Images	O1 (Train:1000u, Test:500u)	52.31%	63.20%	44.53%	86.28%	18	**6**	0	0	11/6
	O2 (Train:1000u, Test:500m)	59.30%	64.59%	55.92%	86.10%	7	4	2	0	11/6
	O3 (Train:10000u, Test:500m)	68.22%	74.31%	63.32%	89.48%	6	**6**	0	0	9/6
	O4 (Train:1000m, Test:500m)	52.70%	63.53%	44.52%	87.03%	10	5	1	0	15/6
	O5 (Train:10000m, Test:500m)	50.56%	54.67%	47.79%	85.94%	7	3	3	0	15/6

C3 erreichte mit 72.92% den höchsten und C2 mit 49.84% den niedrigsten HOTA-Wert. C3 erreichte mit 80.49% den höchsten und C2 mit 50.20% den niedrigsten DetA-Wert. C3 erreichte mit 66.35% den höchsten und C4 mit 43.05% den niedrigsten AssA-Wert. C3 erreichte mit 91.05% den höchsten und O5 mit 85.94% den niedrigsten LocA-Wert. C3 hat mit 3 die wenigsten und C5 und O1 haben mit 18 die meisten ID-Wechsel. C1, C2, C3, C4, O1 und O3 konnten alle 6 Personen mit über 80% der "Lebensdauer" verfolgen,

die anderen Modelle konnten teilweise Personen nur mit 20% bis 80% der "Lebensdauer" oder mit unter 20% der "Lebensdauer" verfolgen. Insgesamt hat C3 7 IDs zugeordnet und liegt somit am nächsten zu den 6 Personen, C2, O4 und O5 haben jeweils 15 IDs zugeordnet und sind damit am weitesten von den 6 Personen entfernt.

In Tabelle 6.9 sind die Ergebnisse der YOLOv3 Modelle anhand des view-IP1 Videos dargestellt. Dabei wurden die COCO und Open Images Datensatz Varianten im Vergleich aufgestellt.

Tabelle 6.9: YOLOv3 mit Varianten beider Datensätze anhand des view-IP1 Videos im Vergleich.

Modell der Datensatz Variante		HOTA	DetA	AssA	LocA	IDSW	MT	PT	ML	Zugeordnete IDs
COCO	C1 (Train:1000u, Test:500u)	65.97%	75.20%	57.95%	88.68%	5	4	2	0	10/6
	C2 (Train:1000u, Test:500m)	70.21%	74.78%	66.08%	90.20%	5	5	1	0	14/6
	C3 (Train:10000u, Test:500m)	73.39%	80.14%	67.25%	91.83%	**3**	5	1	0	10/6
	C4 (Train:1000m, Test:500m)	69.29%	76.59%	63.04%	90.18%	8	5	1	0	10/6
	C5 (Train:10000m, Test:500m)	63.65%	71.34%	56.97%	86.61%	9	5	1	0	11/6
Open Images	O1 (Train:1000u, Test:500u)	72.63%	71.95%	73.42%	88.63%	4	**6**	0	0	16/6
	O2 (Train:1000u, Test:500m)	66.43%	76.78%	57.55%	90.77%	4	**6**	0	0	12/6
	O3 (Train:10000u, Test:500m)	**74.17%**	**81.34%**	67.68%	**92.35%**	**3**	5	1	0	**8/6**
	O4 (Train:1000m, Test:500m)	73.61%	72.01%	**75.31%**	88.37%	5	4	2	0	**8/6**
	O5 (Train:10000m, Test:500m)	57.28%	74.17%	44.30%	87.95%	10	5	1	0	10/6

O3 erreichte mit 74.17% den höchsten und O5 mit 57.28% den niedrigsten HOTA-Wert. O3 erreichte mit 81.34% den höchsten und C5 mit 71.34% den niedrigsten DetA-Wert. O4 erreichte mit 75.31% den höchsten und O5 mit 44.30% den niedrigsten AssA-Wert. O3 erreichte mit 92.35% den höchsten und C5 mit 86.61% den niedrigsten LocA-Wert. C3 und O3 haben mit 3 die wenigsten und O5 hat mit 10 die meisten ID-Wechsel. O1 und O2 konnten alle 6 Personen mit über 80% der "Lebensdauer" verfolgen, die anderen Modelle konnten teilweise Personen nur mit 20% bis 80% der "Lebensdauer" verfolgen. Insgesamt haben O3 und O4 8 IDs zugeordnet und liegen somit am nächsten zu den 6

Personen, O1 hat 16 IDs zugeordnet und sind damit am weitesten von den 6 Personen entfernt.

In Tabelle 6.10 sind die Ergebnisse der Tiny-YOLOv3 Modelle anhand des view-HC4 Videos dargestellt. Dabei wurden die COCO und Open Images Datensatz Varianten im Vergleich aufgestellt.

Tabelle 6.10: Tiny-YOLOv3 mit Varianten beider Datensätze anhand des view-HC4 Videos im Vergleich.

Modell der Datensatz Variante		HOTA	DetA	AssA	LocA	IDSW	MT	PT	ML	Zugeordnete IDs
COCO	C1 (Train:1000u, Test:500u)	30.67%	29.40%	33.22%	79.76%	29	5	1	0	29/6
	C2 (Train:1000u, Test:500m)	32.81%	28.50%	39.84%	77.38%	26	5	1	0	34/6
	C3 (Train:10000u, Test:500m)	44.16%	42.25%	47.92%	83.46%	17	**6**	0	0	23/6
	C4 (Train:1000m, Test:500m)	29.08%	28.43%	31.18%	79.53%	28	**6**	0	0	28/6
	C5 (Train:10000m, Test:500m)	39.09%	38.71%	41.98%	79.86%	**6**	2	4	0	**11/6**
Open Images	O1 (Train:1000u, Test:500u)	42.82%	40.91%	**49.54%**	78.11%	18	4	1	1	18/6
	O2 (Train:1000u, Test:500m)	43.68%	43.11%	47.09%	80.88%	12	2	3	1	**11/6**
	O3 (Train:10000u, Test:500m)	40.77%	40.19%	43.00%	**83.97%**	11	4	2	0	18/6
	O4 (Train:1000m, Test:500m)	39.77%	40.56%	41.16%	79.76%	22	4	1	1	16/6
	O5 (Train:10000m, Test:500m)	**46.22%**	**45.49%**	49.41%	80.09%	7	4	2	0	12/6

O5 erreichte mit 46.22% den höchsten und C4 mit 29.08% den niedrigsten HOTA-Wert. O5 erreichte mit 45.49% den höchsten und C4 mit 28.43% den niedrigsten DetA-Wert. O1 erreichte mit 49.54% den höchsten und C4 mit 31.18% den niedrigsten AssA-Wert. O3 erreichte mit 83.97% den höchsten und C2 mit 77.38% den niedrigsten LocA-Wert. C5 hat mit 6 die wenigsten und C1 hat mit 29 die meisten ID-Wechsel. C3 und C4 konnten alle 6 Personen mit über 80% der "Lebensdauer" verfolgen, die anderen Modelle konnten teilweise Personen nur mit 20% bis 80% der "Lebensdauer" oder mit unter 20% der "Lebensdauer" verfolgen. Insgesamt haben C5 und O2 11 IDs zugeordnet und liegen somit am nächsten zu den 6 Personen, C2 hat 34 IDs zugeordnet und sind damit am weitesten von den 6 Personen entfernt.

In Tabelle 6.11 sind die Ergebnisse der Tiny-YOLOv3 Modelle anhand des view-IP1 Videos dargestellt. Dabei wurden die COCO und Open Images Datensatz Varianten im Vergleich aufgestellt.

Tabelle 6.11: Tiny-YOLOv3 mit Varianten beider Datensätze anhand des view-IP1 Videos im Vergleich.

Modell der Datensatz Variante		HOTA	DetA	AssA	LocA	IDSW	MT	PT	ML	Zugeordnete IDs
COCO	C1 (Train:1000u, Test:500u)	46.88%	54.09%	40.78%	84.37%	13	**5**	0	1	18/6
	C2 (Train:1000u, Test:500m)	48.89%	55.39%	43.36%	84.85%	16	4	1	1	18/6
	C3 (Train:10000u, Test:500m)	**56.85%**	60.16%	**53.99%**	87.45%	10	**5**	0	1	17/6
	C4 (Train:1000m, Test:500m)	45.64%	54.47%	38.61%	84.06%	14	**5**	0	1	18/6
	C5 (Train:10000m, Test:500m)	52.63%	58.31%	48.09%	83.75%	13	**5**	0	1	19/6
Open Images	O1 (Train:1000u, Test:500u)	48.34%	55.88%	42.02%	84.20%	15	**5**	1	0	19/6
	O2 (Train:1000u, Test:500m)	48.16%	54.65%	42.76%	85.06%	17	4	2	0	17/6
	O3 (Train:10000u, Test:500m)	55.30%	**61.67%**	49.77%	**88.17%**	**7**	**5**	0	1	19/6
	O4 (Train:1000m, Test:500m)	43.94%	53.71%	36.25%	84.51%	11	4	1	1	16/6
	O5 (Train:10000m, Test:500m)	56.19%	60.34%	52.87%	85.30%	13	3	2	1	**10/6**

C3 erreichte mit 56.85% den höchsten und O4 mit 43.94% den niedrigsten HOTA-Wert. O3 erreichte mit 61.67% den höchsten und O4 mit 53.71% den niedrigsten DetA-Wert. C3 erreichte mit 53.99% den höchsten und O4 mit 36.25% den niedrigsten AssA-Wert. O3 erreichte mit 88.17% den höchsten und C5 mit 83.75% den niedrigsten LocA-Wert. O3 hat mit 7 die wenigsten und O2 hat mit 17 die meisten ID-Wechsel. Keines der Modelle konnte alle 6 Personen mit über 80% der "Lebensdauer" verfolgen, C1, C3, C4, C5, O1 und O3 konnten 5 Personen mit über 80% der "Lebensdauer" verfolgen, alle Modelle konnten teilweise Personen nur mit 20% bis 80% der "Lebensdauer" verfolgen. Insgesamt hat O5 10 IDs zugeordnet und liegen somit am nächsten zu den 6 Personen. C5, O1 und O3 haben 19 IDs zugeordnet und sind damit am weitesten von den 6 Personen entfernt.

Insgesamt konnten die YOLO Modelle im Vergleich zu den Tiny-YOLO Modellen höhere HOTA, DetA, AssA und LocA Werte erzielen, hatten im Vergleich weniger ID-Wechsel und haben insgesamt weniger IDs zugeordnet.

6.5 Ergebnisse der Performanceanalyse

In Tabelle 6.12 ist ein Performancevergleich zwischen drei Geräten aufgestellt. Dargestellt ist die durchschnittliche Zeit der Objekterkennungen und die durchschnittlichen FPS während der Objektverfolgungen mittels der CPU und der GPU der Geräte und anhand eines YOLOv3 und eines Tiny-YOLOv3 Modells.

Tabelle 6.12: Performancevergleich

System	YOLO Variante	Art der Hardware	⌀-Zeit für Objekterkennung	⌀-FPS während Objektverfolgung
Desktop PC	YOLOv3	CPU	304ms	3.71 FPS
		GPU	84ms	17.14 FPS
	Tiny-YOLOv3	CPU	80ms	17.22 FPS
		GPU	57ms	31.06 FPS
Notebook	YOLOv3	CPU	389ms	2.81 FPS
		GPU	271ms	4.13 FPS
	Tiny-YOLOv3	CPU	119ms	11.36 FPS
		GPU	94ms	13.57 FPS
Raspberry Pi 3 Model B	YOLOv3	CPU	≈14000ms	≈0.1 FPS
	Tiny-YOLOv3	CPU	1891ms	0.6 FPS

Der Desktop PC benötigte mit dem YOLOv3 Modell und der CPU durchschnittlich 304ms und mit der GPU 84ms für die Objekterkennung eines Bildes, mit dem Tiny-YOLOv3 Modell mit der CPU durchschnittlich 80ms und mit der GPU 57ms. Das Notebook benötigte mit dem YOLOv3 Modell und der CPU durchschnittlich 389ms und mit der GPU 271ms für die Objekterkennung eines Bildes, mit dem Tiny-YOLOv3 Modell mit der CPU durchschnittlich 119ms und mit der GPU 94ms. Der Raspberry Pi benötigte mit dem Tiny-YOLOv3 Modell mit der CPU durchschnittlich 1891ms. Eine Auswertung mit dem YOLOv3 Modell war nur teilweise möglich, da diese nach wenigen Bildern abgebrochen ist. Vor dem Abbruch hat der Raspberry Pi ungefähr 14000ms für die Objekterkennung eines Bildes benötigt.

Der Desktop PC hat die Objektverfolgung mit dem YOLOv3 Modell mit der CPU durchschnittlich mit 3.71 FPS und mit der GPU mit 17.14 FPS verarbeitet. Mit dem Tiny-YOLOv3 Modell hat der Desktop PC die Objektverfolgung mit der CPU durchschnittlich mit 17.22 FPS und mit der GPU mit 31.06 FPS verarbeitet. Das Notebook hat die Objektverfolgung mit dem YOLOv3 Modell mit der CPU durchschnittlich mit 2.81 FPS und mit der GPU mit 4.13 FPS verarbeitet. Mit dem Tiny-YOLOv3 Modell hat das Notebook die Objektverfolgung mit der CPU durchschnittlich mit 11.36 FPS und mit der GPU mit 13.57 FPS verarbeitet. Der Raspberry Pi hat die Objektverfolgung mit dem Tiny-YOLOv3 Modell mit der CPU durchschnittlich mit 0.6 FPS verarbeitet. Eine Objektverfolgung mit dem YOLOv3 Modell war ebenfalls nur teilweise möglich, da diese nach wenigen Bildern abgebrochen ist. Vor dem Abbruch hat der Raspberry Pi die Objektverfolgung mit ungefähr 0.1 FPS ausgeführt.

Insgesamt konnten die Geräte mit der GPU niedrigere Ausführungszeiten bei der Objekterkennung und höhere FPS während der Objektverfolgung erzielen. Zudem sind mit dem Tiny-YOLOv3 Modell im Vergleich zum YOLOv3 Modell die Ausführungszeiten bei der Objekterkennung niedriger und die FPS während der Objektverfolgung höher.

6.6 Diskussion

Die Analyse der COCO und Open Images Datensätze hat gezeigt, dass beide “Person” Datensätze aufgrund der zuvor aufgelisteten Auffälligkeiten Schwachstellen (unter anderem, dass Personen nicht mit Bounding Boxes umrandet waren) beinhalten. Somit sind beide Datensätze nicht ideal dafür geeignet, um festzustellen, wie gut die Vorhersagen anhand der unmodifizierten Test-Datensätze funktionieren. Die Qualität des COCO Datensatzes ist allgemein besser, da nur 22% der Daten angepasst werden mussten. Beim Open Images Datensatz mussten dagegen 54% der Daten angepasst werden. Daher hat dieser mehr Aufwand benötigt, um die Datenqualität zu verbessern. Grund hierfür könnte sein, dass durch die Klasse “Person” nicht alle Personen erfasst wurden, da sie einer Unterkategorie, wie z.B. “Mann” oder “Frau” zugeordnet worden sind.

Die Abbildungen der Trainingsverläufe haben gezeigt, dass das Training mit den Datensatz Varianten gut verlaufen ist, da der Loss-Wert über die Epochen gesunken ist und nur minimal ansteigt. Einzig die C5 und O5 Varianten sind stärker angestiegen, was auf ein Overfitting deutet. Die mittels Data Augmentation erstellten Bilder waren sich

wahrscheinlich zu ähnlich. Es konnte zwischen den YOLOv3 und Tiny-YOLOv3 Loss-Verläufen kein wesentlicher Unterschied festgestellt werden. Die Loss-Kurven verliefen ähnlich.

Die Ergebnisse der Objekterkennung haben Unterschiede zwischen den Modellen der Datensatz Varianten gezeigt. Sowohl die C3 als auch die O3 YOLOv3 und Tiny-YOLOv3 Modelle haben im mAP Vergleich des jeweiligen Test-Datensatzes am besten abgeschnitten, die C5 und O5 YOLOv3 und Tiny-YOLOv3 Modelle haben überwiegend am schlechtesten abgeschnitten. Dies lässt sich mit dem Overfitting während des Trainings erklären, da diese Modelle bei unbekannten Bildern schlechter funktioniert haben. Dies hat zudem dazu geführt, dass die C5 und O5 Modelle die niedrigste Anzahl an Vorhersagen getätigt haben. Positiv an den Modellen ist jedoch, dass diese im Verhältnis weniger FPs hatten. Besonders auffällig sind die O1 YOLOv3 und Tiny-YOLOv3 Modelle, da diese einen sehr niedrigen mAP Wert im Vergleich zu den anderen Varianten aufweisen. Dies deutet darauf hin, dass der unbearbeitete Open Images Test-Datensatz nicht ideal für die Evaluation genutzt werden kann, da die O2 YOLOv3 und Tiny-YOLOv3 Modelle mit dem modifizierten Test-Datensatz einen deutlich höheren mAP Wert haben. Unter Betrachtung des kombinierten COCO und Open Images Datensatzes hat das C3 Modell sowohl in der YOLOv3 als auch in der Tiny-YOLOv3 Ausführung am besten abgeschnitten. Durch das manuelle Beschriften der Datensätze konnte der mAP-Wert erhöht werden, jedoch hat dies ebenfalls zu einer Erhöhung der False Positives geführt.

Anhand der Vorhersagen des Beispielbildes sind ebenfalls Unterschiede zu sehen. Die Vorhersagen der YOLOv3 Modelle sind im Vergleich zu den Tiny-YOLOv3 Modellen viel präziser. Die Tiny-YOLOv3 Modelle haben zudem teilweise eine der Personen als mehrere Personen identifiziert. Dies könnte an den verwendeten Datensätzen liegen, da die Personen oftmals nicht vollständig im Bild dargestellt waren. Anhand des Beispielbildes haben die Modelle der COCO Varianten etwas besser abgeschnitten als die Modelle der Open Images Varianten.

Die Ergebnisse der Objektverfolgung haben ebenfalls Unterschiede zwischen den Modellen der Datensatz Varianten gezeigt. Anhand der Ergebnisse lässt sich jedoch nicht das beste Modell bestimmen. Die Werte sind einerseits stark davon abhängig, um was für eine Art von Video es sich während der Objektverfolgung handelt. Hinsichtlich der HOTA, DetA, AssA und LocA Werte mit den YOLOv3 Modellen hat C3 beim view-HC4 Video am besten abgeschnitten, beim view-IP1 Video konnte jedoch O3 die besseren Ergebnisse erzielen. Bei den Tiny-YOLOv3 Modellen sind die besten Werte auf mehr

Modelle verteilt, sodass bei diesen Varianten das beste Modell von den Anforderungen der Objektverfolgung abhängig ist.

Die Ergebnisse der Performanceanalyse haben gezeigt, dass es einen großen Performanceunterschied zwischen des YOLOv3 Modells und des Tiny-YOLOv3 Modells und der Nutzung einer CPU oder GPU gibt. Insgesamt konnten die Objekterkennungen mit dem Tiny-YOLOv3 Modell deutlich schneller erfolgen. Außerdem konnten die FPS während der Verarbeitung der Objektverfolgung ebenfalls durch das Tiny-YOLOv3 Modell erhöht werden. Der Grund hierfür ist, dass das Tiny-YOLOv3 Netzwerk deutlich kleiner als das YOLOv3 Netzwerk ist. Der Vergleich der Art der Hardware hat zudem bestätigt, dass durch die Benutzung einer GPU die durchschnittliche Zeit für Objekterkennungen reduziert und die durchschnittlichen FPS während der Objektverfolgung erhöht werden konnten. Dies trifft sowohl auf die Benutzung des YOLOv3 Modells als auch auf die des Tiny-YOLOv3 Modells zu. Auf dem Raspberry Pi waren die Objekterkennung und Objektverfolgung mit dem Tiny-YOLOv3 Modell möglich, jedoch ist die durchschnittliche Ausführungszeit der Objekterkennung deutlich höher und die FPS während der Verarbeitung der Objektverfolgung deutlich geringer, im Vergleich zu dem Desktop PC und dem Notebook. Mit dem YOLOv3 Modell ist der Raspberry Pi an seine Grenzen gestoßen, da die Prozesse der Objekterkennung und Objektverfolgung nach wenigen Bildern abgebrochen sind. Gründe dafür könnten nicht ausreichend Arbeitsspeicher und die Größe des YOLOv3 Modells sein.

Allgemein lässt es sich behaupten, dass die Personen mit den Objekterkennungen und den Objektverfolgungen mit den YOLOv3 Modellen im Vergleich zu den Tiny-YOLOv3 Modellen genauer vorhergesagt werden. Mit den Tiny-YOLOv3 Modellen ist jedoch eine zeitliche Ersparnis bei den Objekterkennungen und eine schnellere Verarbeitung während der Objektverfolgung möglich, sodass diese auch auf einem Einplatinencomputer, wie einem Raspberry Pi möglich ist. Die Auswahl des Modells ist abhängig von den Anforderungen, da nicht das beste Modell unter den Varianten festgestellt werden konnte. Die manuelle Beschriftung führte zu einer Erhöhung der mAP, jedoch auch zu einer höheren Anzahl an False Positives. Die Vervielfältigung der Bilder hat nicht zu einer Verbesserung der Vorhersagen geführt, stattdessen wurden diese im Vergleich schlechter. Dies ist anhand des Loss-Verlaufs und der Ergebnisse der Objekterkennung und Objektverfolgung ersichtlich. Die Ergebnisse haben außerdem gezeigt, dass sich beide Datensätze für das Training eignen, wobei der Open Images Datensatz deutlich größer ist und mehr Objektklassen beinhaltet. Die Ergebnisse der Performanceanalyse konnten nicht die Ausführungszeiten der Abbildung 3.1 erreichen, da für die Evaluation eine leistungs-

schwächere GPU verwendet wurde und weil die ursprüngliche YOLOv3 Implementation von Redmon et al. [25] hinsichtlich der Performance höchstwahrscheinlich optimierter ist als die verwendete Implementation.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein System entwickelt, welches ein Training eines neuronalen Netzwerkes anhand des COCO oder Open Images Datensatzes ermöglicht, mit dem eine Objekterkennung und Objektverfolgung beliebiger Objektklassen möglich ist.

Dazu wurden in Kapitel 2 zunächst die Grundlagen der Objektidentifizierung und Qualitätsmaße beschrieben. In Kapitel 3 wurden die verwendeten Algorithmen YOLO und DeepSORT erläutert, außerdem wurden die verwendeten Datensätze COCO und Open Images beschrieben. In Kapitel 4 wurde das Gesamtsystem beschrieben, welches einerseits aus der Datenbeschaffung, der Datenvorbereitung, dem Training, der Evaluation der Objekterkennung, der Evaluation der Objektverfolgung, der Objekterkennung und Objektverfolgung und andererseits aus den Tools zur Anpassung der Datensätze (Manuelles Filtern der Datensätze, Manuelle Beschriftung der Datensätze und Vervielfältigung der Daten) besteht. In Kapitel 5 wurde die Durchführung des Systems beschrieben und in Kapitel 6 wurden die Ergebnisse vorgestellt und diskutiert.

Die Ergebnisse der Auswertungen haben gezeigt, dass das entwickelte System für die Objekterkennung und Objektverfolgung genutzt werden kann. Zur Verbesserung des Systems könnten weitere Datensätze in die Datenbeschaffung eingebunden werden, mit denen möglicherweise noch bessere Resultate erzielt werden könnten. Das Tool zur Vervielfältigung der Daten sollte weiterhin in Betracht gezogen werden, durch eine Anpassung der verwendeten Transformationen bzw. durch eine Reduzierung der Anzahl der neu erstellten Bilder könnten bessere Ergebnisse erzielt werden.

Für eine bessere Genauigkeit und Performance könnte beispielsweise das YOLOv4 Netzwerk genutzt werden. Das implementierte System verwendet vordefinierte Anchor Boxes, welche individuell mittels k-Means-Algorithmus anhand des verwendeten Datensatzes angepasst werden könnten, um noch bessere Resultate zu erzielen. Zusätzlich könnten Anpassungen an der Größe der Eingabebilder durchgeführt werden, um entweder die Genauigkeit oder die Performance zu verbessern.

Das System verwendet beim DeepSORT Algorithmus für den Deep Appearance Descriptor ein Modell, welches mit einem Personendatensatz trainiert wurde, sodass dieser mit Personen am besten funktioniert. Hierfür könnte ein weiterer Prozess für die Erstellung eines passenden Deep Appearance Descriptors erstellt werden, falls ein Training mit anderen Objektklassen durchgeführt werden soll, um die Genauigkeit der Objektverfolgung für weitere Objektklassen zu gewährleisten.

Der DeepSORT Algorithmus könnte mit einem Algorithmus zur Detektion und Beschreibung lokaler Merkmale in Bildern, wie dem SIFT Algorithmus ergänzt werden, mit dem Übereinstimmungen von Objektmerkmalen überprüft werden könnten. Dies würde eine Zuordnung von Objekten ermöglichen, die bereits bekannt sind, aber eine längere Zeit aus dem Bild verschwunden sind, wodurch die Anzahl der ID-Wechsel reduziert werden könnte. Die Einbindung des SIFT Algorithmus könnte sich jedoch negativ auf die Performance auswirken, sodass die Objektverfolgung möglicherweise nicht in Echtzeit durchgeführt werden kann. In Abbildung 7.1 sind Übereinstimmungen von SIFT-Merkmalen exemplarisch anhand von zwei Personen dargestellt.

(a) Identische Personen. (b) Unterschiedliche Personen.

Abbildung 7.1: Übereinstimmende SIFT-Merkmale anhand von Personen.

Die Abbildung 7.1a enthält übereinstimmenden SIFT-Merkmale, da es sich den Abbildungen um identische Personen handelt. In Abbildung 7.1b dagegen wurden unterschiedliche Personen betrachtet, sodass keine übereinstimmende SIFT-Merkmale festgestellt werden konnten.

Die Ergebnisse der Performanceanalyse haben gezeigt, dass die Objekterkennung und Objektverfolgung ebenfalls mit einem Raspberry Pi ausgeführt werden kann. Der verwendete Raspberry Pi 3B stößte allerdings beim YOLO Modell wegen seiner 1 GB Arbeitsspeicher an seine Grenzen, sodass hierfür mindestens ein Raspberry Pi 4 mit 2 GB Arbeitsspeicher genutzt werden sollte. Da der Raspberry Pi 4 eine leistungsstärkere CPU besitzt, ist eine bessere Performance zu erwarten. Außerdem wäre eine Performanceanalyse mit einem Jetson Nano interessant, mit dem eine deutliche Performancesteigerung aufgrund der vorhandenen GPU zu erwarten wäre.

Für einen zusätzlichen Vergleich der Genauigkeit der Modelle könnten existierende Challenges, wie die COCO Challenge oder die MOT Challenge genutzt werden. Dadurch könnte eine Aussage darüber getroffen werden, wie gut die Modelle im Vergleich zu anderen Modellen abschneiden.

Literaturverzeichnis

[1] Aidouni, Manal E.: Evaluating Object Detection Models: Guide to Performance Metrics. (2019). – URL https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html

[2] Babb, Tim: How a Kalman filter works, in pictures. (2015). – URL http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/

[3] Brownlee, Jason: *A Gentle Introduction to Object Recognition With Deep Learning*. https://machinelearningmastery.com/object-recognition-with-deep-learning/. 2019

[4] Ciaparrone, Gioele ; Luque Sánchez, Francisco ; Tabik, Siham ; Troiano, Luigi ; Tagliaferri, Roberto ; Herrera, Francisco: Deep learning in video multi-object tracking: A survey. In: *Neurocomputing* 381 (2020), Mar, S. 61–88. – URL http://dx.doi.org/10.1016/j.neucom.2019.11.023. – ISSN 0925-2312

[5] Dendorfer, Patrick ; Rezatofighi, Seyed H. ; Milan, Anton ; Shi, Javen ; Cremers, Daniel ; Reid, Ian D. ; Roth, Stefan ; Schindler, Konrad ; Leal-Taixé, Laura: CVPR19 Tracking and Detection Challenge: How crowded can it get? In: *CoRR* abs/1906.04567 (2019). – URL http://arxiv.org/abs/1906.04567

[6] Dimitrievski, Martin ; Veelaert, Peter ; Philips, Wilfried: Behavioral Pedestrian Tracking Using a Camera and LiDAR Sensors on a Moving Vehicle. In: *Sensors* 19 (2019), Nr. 2. – URL https://www.mdpi.com/1424-8220/19/2/391. – ISSN 1424-8220

[7] Fang, Wei ; Wang, Lin ; Ren, Peiming: Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments. In: *IEEE Access* 8 (2020), S. 1935–1944

[8] Grace A. Lewis, B. Craig M.: A Model Problem Approach to Measurement-to-Track Association. (2003)

[9] Hall, David ; Dayoub, Feras ; Skinner, John ; Zhang, Haoyang ; Miller, Dimity ; Corke, Peter ; Carneiro, Gustavo ; Angelova, Anelia ; Sünderhauf, Niko: *Probabilistic Object Detection: Definition and Evaluation*. https://arxiv.org/abs/1811.10800. 2020

[10] He, Jiawei ; Huang, Zehao ; Wang, Naiyan ; Zhang, Zhaoxiang: *Learnable Graph Matching: Incorporating Graph Partitioning with Deep Feature Learning for Multiple Object Tracking*. 2021

[11] Hochreiter, Sepp: Untersuchungen zu dynamischen neuronalen Netzen. (1991)

[12] Jurić, Darko: Object Tracking: Kalman Filter with Ease. (2015). – URL https://www.codeproject.com/articles/865935/object-tracking-kalman-filter-with-ease

[13] Kalman, Rudolf E.: A New Approach to Linear Filtering and Prediction Problems. (1960)

[14] Kanjee, Ritesh: DeepSORT - Deep Learning applied to Object Tracking). (2020). – URL https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104

[15] Kathuria, Ayoosh: How to Implement a YOLO (v3) Object Detector from Scratch in PyTorch. (2018). – URL https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

[16] Kuznetsova, Alina ; Rom, Hassan ; Alldrin, Neil ; Uijlings, Jasper ; Krasin, Ivan ; Pont-Tuset, Jordi ; Kamali, Shahab ; Popov, Stefan ; Malloci, Matteo ; Kolesnikov, Alexander ; al. et: The Open Images Dataset V4. In: *International Journal of Computer Vision* 128 (2020), Mar, Nr. 7, S. 1956–1981. – URL http://dx.doi.org/10.1007/s11263-020-01316-z. – ISSN 1573-1405

[17] Li, Guofa ; Yang, Yifan ; Qu, Xingda: Deep Learning Approaches on Pedestrian Detection in Hazy Weather. In: *IEEE Transactions on Industrial Electronics* 67 (2020), Nr. 10, S. 8889–8899

[18] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Bourdev, Lubomir ; Girshick, Ross ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Zitnick, C. L. ;

Dollár, Piotr: *Microsoft COCO: Common Objects in Context*. https://arxiv.org/abs/1405.0312. 2015

[19] Liu, Wei ; Liao, Shengcai ; Ren, Weiqiang ; Hu, Weidong ; Yu, Yinan: High-Level Semantic Feature Detection: A New Perspective for Pedestrian Detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019

[20] Luiten, Jonathon: How to evaluate tracking with the HOTA metrics. (2021). – URL https://autonomousvision.github.io/hota-metrics/

[21] Ma, Haojie ; Liu, Yalan ; Ren, Yuhuan ; Yu, Jingxian: Detection of Collapsed Buildings in Post-Earthquake Remote Sensing Images Based on the Improved YOLOv3. In: *Remote Sensing* 12 (2020), Nr. 1. – URL https://www.mdpi.com/2072-4292/12/1/44. – ISSN 2072-4292

[22] Mantripragada, Manogna: Digging deep into YOLO V3 - A hands-on guide. (2020). – URL https://towardsdatascience.com/digging-deep-into-yolo-v3-a-hands-on-guide-part-1-78681f2c7e29

[23] Meel, Vidushi: What is the COCO Dataset? What you need to know in 2021. (2021). – URL https://viso.ai/computer-vision/coco-dataset/

[24] Meel, Vidushi: YOLOv3: Real-Time Object Detection Algorithm (What's New?). (2021). – URL https://viso.ai/deep-learning/yolov3-overview/

[25] Redmon, Joseph ; Farhadi, Ali: *YOLOv3: An Incremental Improvement*. https://arxiv.org/abs/1804.02767. 2018

[26] Riggio, Christopher: What's the deal with Accuracy, Precision, Recall and F1? (2019)

[27] Rosebrock, Adrian: Intersection over Union (IoU) for object detection. (2016). – URL https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[28] Shah, Deval: The Surveillance phenomenon you must know about : Multi Object Tracking. (2020). – URL https://medium.com/visionwizard/object-tracking-675d7a33e687

[29] Singh, Aishwarya: Selecting the Right Bounding Box Using Non-Max Suppression (with implementation). (2020). – URL https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

[30] Soviany, Petru ; Ionescu, Radu T.: Optimizing the Trade-off between Single-Stage and Two-Stage Object Detectors using Image Difficulty Prediction. In: *CoRR* abs/1803.08707 (2018). – URL http://arxiv.org/abs/1803.08707

[31] Wojke, Nicolai ; Bewley, Alex ; Paulus, Dietrich: *Simple Online and Realtime Tracking with a Deep Association Metric*. 2017

[32] Xiaotian Gonga, Li M. ; Ouyang, Hangkong: An improved method of Tiny YOLOV3. (2020)

[33] Yoshua Bengio, Patrice S. ; Frasconi, Paolo: Learning Long-Term Dependencies with Gradient Descent is Difficult. (1994)

[34] Świeżewski, Jędrzej: What is YOLO Object Detection? (2020). – URL https://appsilon.com/object-detection-yolo-algorithm/

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