

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Künstliche Intelligenz

Studienarbeit Deep Vision

von

André Kestler

**Vergleich der YOLOX- und YOLOv8-Modelle für die
Objekterkennung im Kontext des Udacity Self Driving
Car-Datensatzes**

Bearbeitungszeitraum: von 21. Juni 2023
bis 19. Juli 2023

1. Prüfer: Prof. Dr. phil. Tatyana Ivanovska

Inhaltsverzeichnis

| | |
|---|-----------|
| 1 Einleitung | 1 |
| 1.1 Aufgabenstellung | 1 |
| 1.2 Übersicht | 1 |
| 2 Datensatz | 2 |
| 2.1 Beschreibung | 2 |
| 2.2 Klassenaufteilung | 2 |
| 2.3 Aufbau | 3 |
| 2.3.1 Rohdaten | 3 |
| 2.3.2 YOLOX | 4 |
| 2.3.3 YOLOv8 | 4 |
| 3 Modell 1: YOLOX | 5 |
| 3.1 Architektur | 5 |
| 3.1.1 Decoupled Head | 6 |
| 3.1.2 Anchor Free Prediction | 7 |
| 3.1.3 SimOTA Label Assignment | 7 |
| 3.1.4 Advanced Augmentation | 7 |
| 3.2 Verlustfunktion | 7 |
| 3.3 Modellauswertung | 7 |
| 4 Modell 2: YOLOv8 | 8 |
| 4.1 Architektur | 8 |
| 4.2 Verlustfunktion | 9 |
| 4.3 Modellauswertung | 9 |
| 5 Zusammenfassung und Ausblick | 10 |

| | |
|----------------------------------|-----------|
| Literaturverzeichnis | 11 |
| Abbildungsverzeichnis | 12 |
| 6 Anhang | 13 |
| 6.1 Ordnerstruktur | 13 |
| 6.2 Dateien: Datensatz | 13 |
| 6.3 Dateien: YOLOX | 13 |
| 6.4 Dateien: YOLOv8 | 13 |

Abkürzungsverzeichnis

| | |
|------|------------------------------|
| IOU | Intersection over Union |
| OTA | Optimal Transport Assignment |
| YOLO | You only look once |

Kapitel 1

Einleitung

1.1 Aufgabenstellung

Im Rahmen der Vorlesung Deep Vision ist ein Projekt im Themenbereich des Kurses zu bearbeiten. Die Bearbeitung erfolgt als Einzelarbeit. Als Projekt werden zwei YOLO (You only look once) Netzwerke mit dem Udacity Self Driving Car Datensatz trainiert und miteinander verglichen. In der folgenden Arbeit werden YOLOX und YOLOv8 verwendet.

1.2 Übersicht

In Kapitel 2 wird zunächst der Datensatz beschrieben. Dabei wird auf die Klasseneinteilung und die Datenstruktur eingegangen. In Kapitel 3 wird das YOLOX-Netzwerk vorgestellt. Dabei wird auf die verwendete Verlustfunktion, die Architektur und die Ergebnisse mit dem Datensatz eingegangen. Kapitel 4 beschreibt die verwendete Architektur von YOLOv8. Außerdem werden die Ergebnisse anhand des verwendeten Datensatzes präsentiert. Das letzte Kapitel Zusammenfassung und Ausblick befasst sich mit dem direkten Vergleich der Ergebnisse und gibt einen Ausblick über die weitere Bearbeitung. Im Anhang wird beschrieben, wie die angegebenen Skripte verwendet werden, um die Netzwerke selbst zu trainieren.

Kapitel 2

Datensatz

2.1 Beschreibung

Der Udacity Self Driving Car Dataset [5] ist eine umfangreiche Sammlung von Bildern, die von Kameras in Fahrzeugen aufgenommen wurden. Der Datensatz beinhaltet 15000 Samples mit einer Auflösung von 512x512 Pixeln. Er besteht aus Bildern und die zugehörigen Annotationen, die Informationen über die enthaltenen Objekte in der Umgebung enthalten.

Die Bilder in dem Datensatz umfassen verschiedene Szenarien im Straßenverkehr. Darunter befinden sich Stadt- und Landstraßen. Die Bilder wurden bei unterschiedlichen Lichtbedingungen aufgenommen, um ein breite Vielfalt an Situationen in den Daten abzudecken.

2.2 Klassenaufteilung

Die ursprüngliche Klasseneinteilung ist in Abbildung 2.2 dargestellt. Dort lässt sich erkennen, dass insbesondere die Aufteilung der Klasse *Ampel* in Unterkategorien unterrepräsentiert



Abbildung 2.1: Beispielbilder aus dem Datensatz. Quelle: [5]

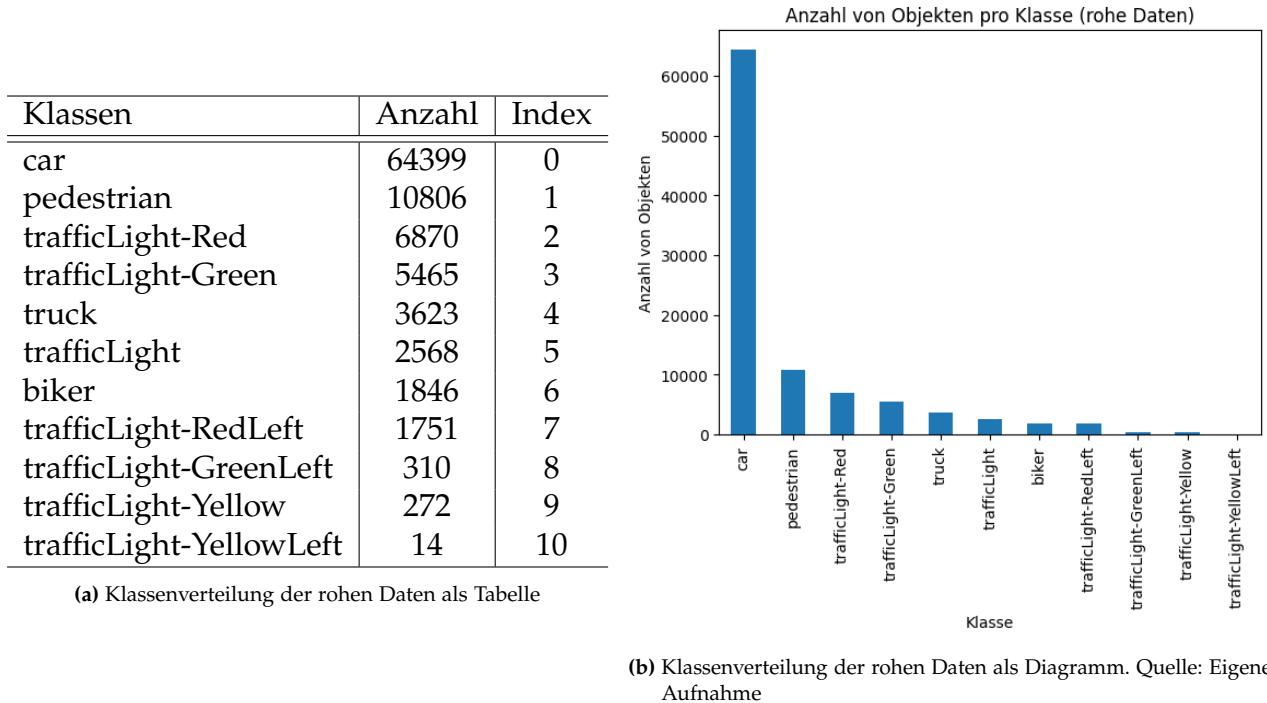


Abbildung 2.2: Klassenverteilung der rohen Daten

ist. Um dieses Problem zu umgehen, wurden die *Ampel*-Klassen zu einer Klasse *trafficLight* zusammengeführt. Für das Auswerten des Datensatzes wurde dieser mit dem passenden Skript (*datasetPreprocessing.ipynb*) in einen Trainings-, Validierungs- und Testdatensatz aufgeteilt. Diese Aufteilung kann aus den farblichen Säulen in Abbildung 2.3 entnommen werden. Eine weitere Bearbeitung des Datensatzes wurde nicht vorgenommen, um zu überprüfen, wie die verwendeten Netzwerke mit einem unbalancierten Datensatz umgehen.

2.3 Aufbau

2.3.1 Rohdaten

Die Rohdaten sind in einem Ordner gespeichert. Dieser Ordner enthält die Bilder im .jpg-Format und eine .csv-Datei in der alle Annotationen enthalten sind. Der Header der Datei enthält den Dateinamen des Bildes, die Breite und Höhe, die Klasse und die vier Bounding Box Koordinaten zu jedem Objekt.

Die folgenden Umwandlungen in die Formate werden mit dem Skript *datasetConversion.ipynb* gemacht. Die jeweiligen Ordnerstrukturen können in Kapitel 6 nachgelesen werden.

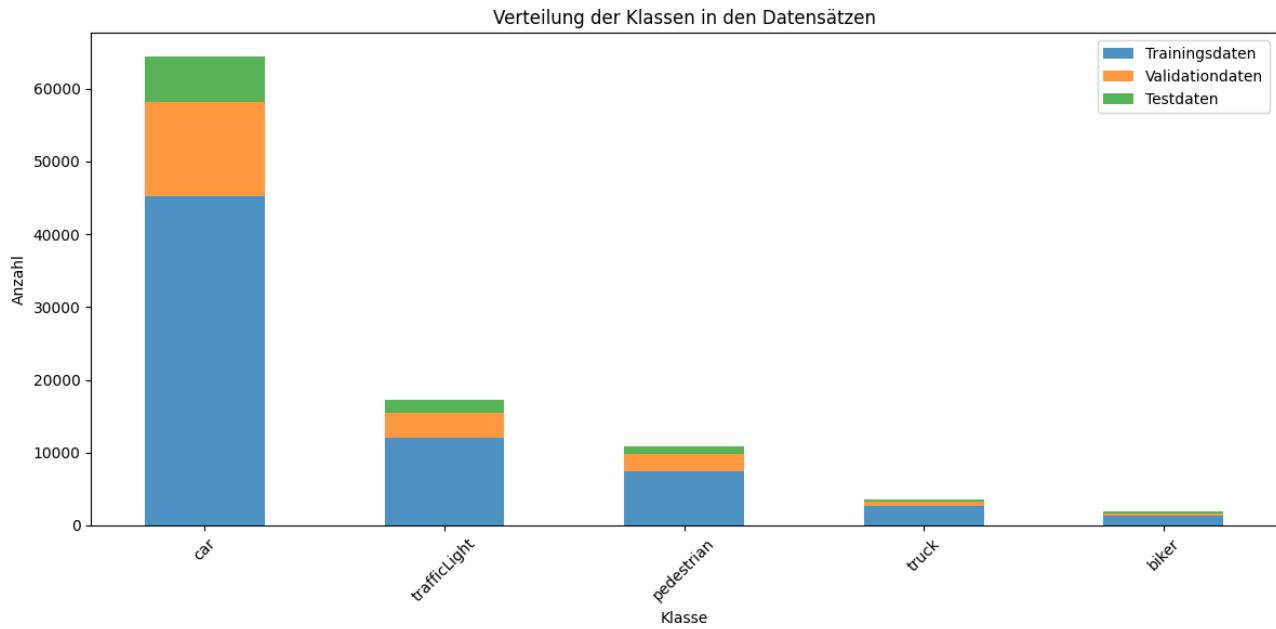


Abbildung 2.3: Aufteilung in Trainings-, Validation- und Testdaten. Quelle: Eigene Aufnahme

2.3.2 YOLOX

Das Netzwerk kann das COCO-Format und das PASCAL VOC-Format verarbeiten. In der folgenden Arbeit wird das COCO-Format verwendet. Zu diesem Zweck werden die Annotationen in einem separaten Ordner und die jeweiligen Bilddateien für Training, Validierung und Test ebenfalls in einem separaten Ordner abgelegt. Die Annotationen zu den drei Teildatensätzen liegen im .json-Format vor. Dabei wird jeder Klasse, jedem Bild und jeder Annotation eine ID zugewiesen, die die Zuordnung der Objekte zu den jeweiligen Bildern ermöglicht.

2.3.3 YOLOv8

Dieses Netzwerk verwendet das YOLO-Format. Dabei werden die Bilder für Training, Validierung und Test in einem separaten Ordner gespeichert. Die dazugehörigen Labels stehen in einzelnen .txt-Dateien. Jedes Bild erhält eine zugehörige Textdatei mit dem gleichen Namen wie das Bild und der Struktur Klasse, x-Zentrum, y-Zentrum, Breite und Höhe. Die Koordinaten müssen auf die Bildgröße normiert sein. Die Labels liegen in einer korrespondierenden Ordnerstruktur. Die zugehörige Konfigurationsdatei gibt anschließend den Pfad zu den Datensatz und die Anzahl der Klassen an.

Kapitel 3

Modell 1: YOLOX

3.1 Architektur

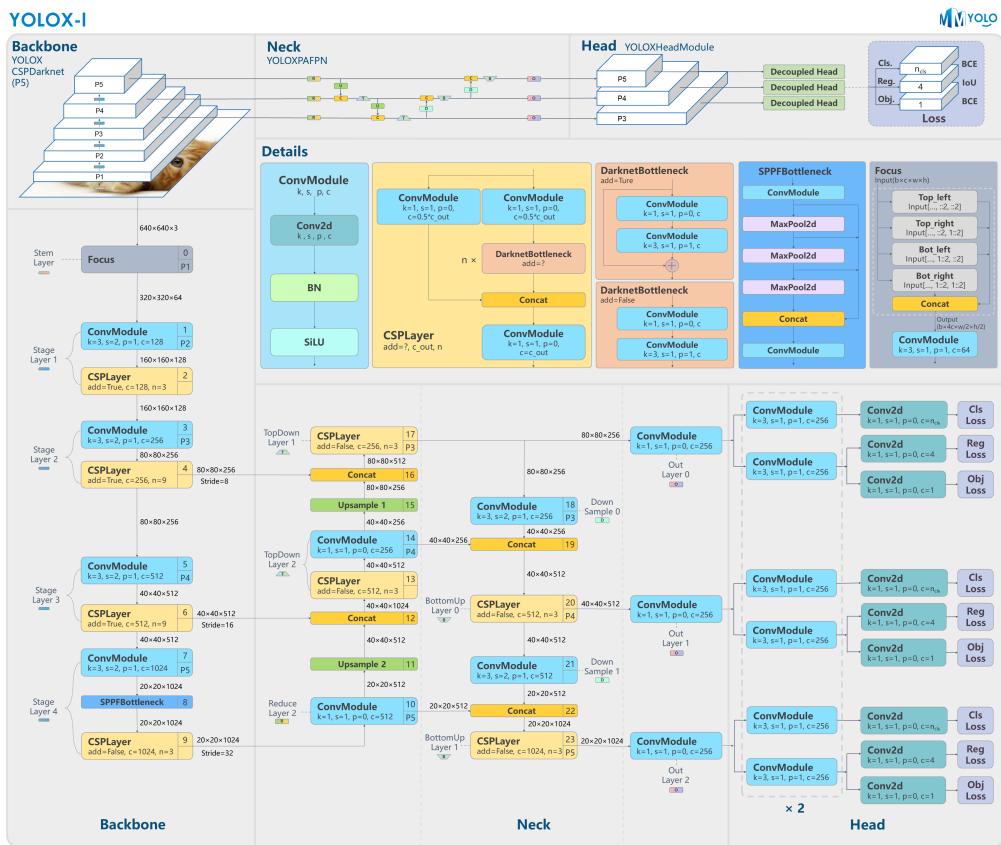


Abbildung 3.1: Übersicht über die Architektur von YOLOX. Quelle: [1, 2, 3]

Die YOLOX Architektur besteht aus einem Backbone-Netz, dem Neck und einem Head.

Das **Backbone**-Netzwerk ist für die Extraktion der Merkmale aus dem Bild verantwortlich. YOLOX verwendet das CSPDarknet als Backbone, um Merkmale auf 3 verschiedenen Maßstäben zu extrahieren. Die erste Ausgabe hat eine Dimension von $(H/8 \times W/8 \times 256)$, die Zweite eine Dimension von $(H/16 \times W/16 \times 512)$ und die Letzte eine Größe von $(H/32 \times W/32 \times 1024)$. Die beiden Parameter H und W sind die Höhe und Breite der Eingabe. Diese drei Ausgänge werden an das Neck weitergegeben. Durch die unterschiedlichen Skalierungen kann das Netz Merkmale für verschiedene Größen erzeugen. Der 256-Kanal-Ausgang extrahiert Merkmale mit einer kleineren Skalierung, während der 1024-Kanal-Ausgang Merkmale mit einer größeren Skalierung extrahiert. Mit zunehmender Tiefe des Netzes werden die Feature-Maps kleiner. Dadurch bleibt weniger Information von dem Originalbild erhalten. Die steigende Anzahl der Kanäle soll diesem Effekt entgegenwirken.

Die Ausgaben aus dem Backbone werden im **CSP-Pyramid-Neck** zu einer Merkmalspyramide verarbeitet, um Objekte unterschiedlicher Größen zu erkennen. Die drei verschiedenen Ausgaben werden wieder miteinander verrechnet (Faltung, Upsampling, Konkatenieren), um sie zu verbinden. Anschließend werden sie mit einem **CSP-Layer** auf drei unterschiedliche Dimensionen reduziert, um die Ergebnisse in den Head zu geben.

Der **Head** ist die Ausgabeschicht des Netzwerks und enthält die Detektionskomponente. YOLOX verwendet einen Decoupled Head, der aus zwei Teilen besteht. Dieser Mechanismus ist mit YOLOX neu eingeführt worden und wird in Kapitel 3.1.1 beschrieben.

3.1.1 Decoupled Head

Der Decoupled Head trennt die Vorhersage von Objekten und Bounding Boxes in zwei Zweige auf. Zusätzlich zum Pfad der Bounding-Box wird dort der Konfidenzwert vorhergesagt. Bei den herkömmlichen YOLO-Netzwerken wird die Vorhersage (Klasse, Bounding Box und Konfidenzwert) in einer einzigen Vorhersage gemacht. Dies kann zu Schwierigkeiten bei der Erkennung von Objekten unterschiedlicher Größe führen.

Wie in der Abbildung 3.2 (unten) zu sehen ist, wird im Decoupled Head die Dimension des Eingangs durch eine 1×1 -Faltung reduziert und anschließend in zwei Pfade aufgeteilt. Das bedeutet, dass das Modell zuerst die Präsenz von Objekten vorhersagt und in einem parallelen Zweig die Bounding-Box-Koordinaten und den Objektscore für die erkannten Objekte berechnet. Dieser Head wird für jede drei FPN-Merkmale ausgeführt. [4]

Die drei Tensorausgaben von YOLOX enthalten die gleichen Informationen wie die Ausgänge des großen Tensors von YOLOv3:

- Cls: Die Klasse jeder Bounding Box
- Reg: Die 4 Teile der Bounding Box
- Obj: Wie sicher ist das Netzwerk, dass innerhalb der Bounding Box ein beliebiges Objekt ist

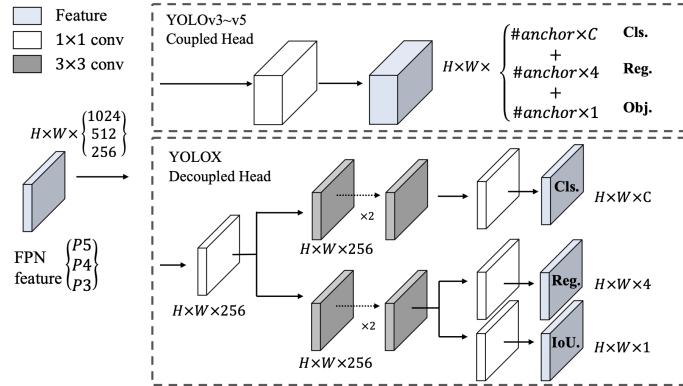


Abbildung 3.2: Illustration des Unterschieds zwischen dem YOLOv3-Head und dem neuen Decoupled-Head Quelle: [2]

3.1.2 Anchor Free Prediction

3.1.3 SimOTA Label Assignment

3.1.4 Advanced Augmentation

3.2 Verlustfunktion

3.3 Modellauswertung

Kapitel 4

Modell 2: YOLOv8

4.1 Architektur

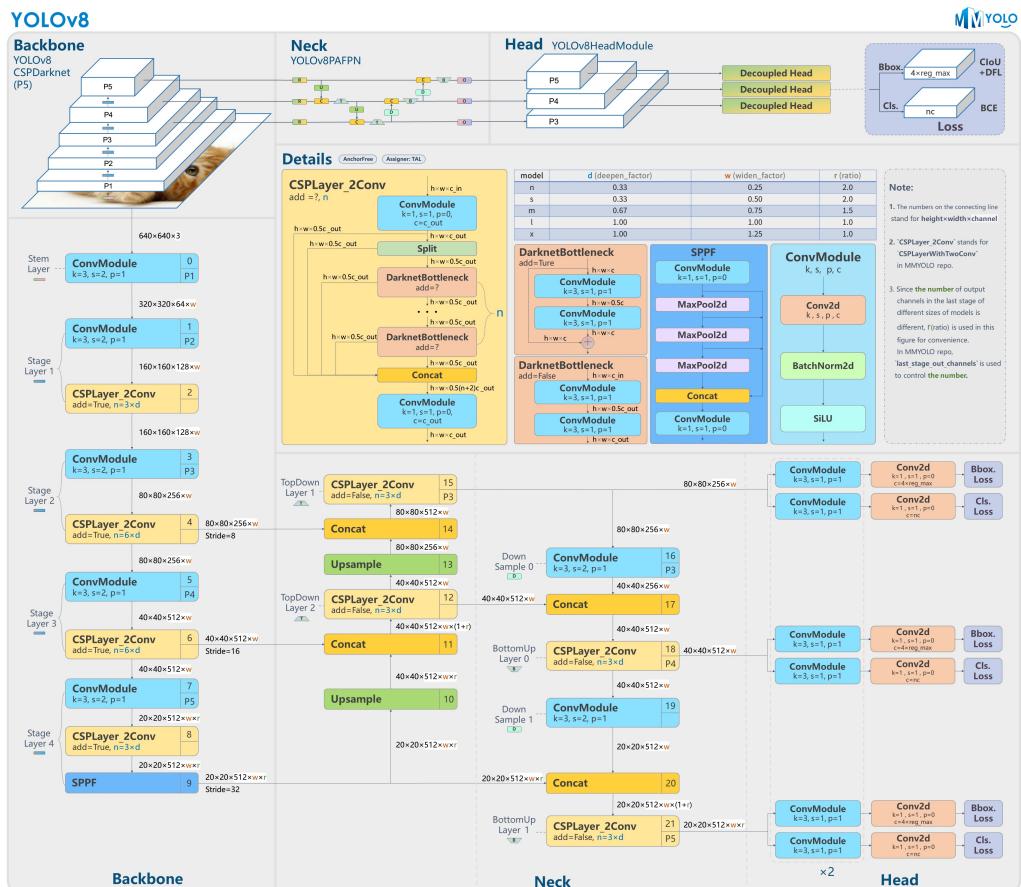


Abbildung 4.1: Übersicht über die Architektur von YOLOv8. Quelle: [1]

4.2 Verlustfunktion

4.3 Modellauswertung

Kapitel 5

Zusammenfassung und Ausblick

Literaturverzeichnis

- [1] CONTRIBUTORS, MMYOLO: *Udacity Self Driving Car Dataset*. <https://github.com/open-mmlab/mmyolo>, Abruf: 10. Juli 2023
- [2] GE, Zheng ; LIU, Songtao ; WANG, Feng ; LI, Zeming ; SUN, Jian: YOLOX: Exceeding YOLO Series in 2021. In: *arXiv preprint arXiv:2107.08430* (2021)
- [3] MEGVII-BASEDETECTION: YOLOX. <https://github.com/Megvii-BaseDetection/YOLOX>. Version: 2021
- [4] MONGARAS, Gabriel: *YOLOX Explanation — How Does YOLOX Work?* <https://medium.com/mlearning-ai/yolox-explanation-how-does-yolox-work-3e5c89f2bf78>, Abruf: 12. Juli 2023
- [5] ZHANG, Edward: *Udacity Self Driving Car Dataset*. <https://www.kaggle.com/datasets/sshikamaru/udacity-self-driving-car-dataset>, Abruf: 23. Juni 2023

Abbildungsverzeichnis

| | | |
|-----|---|---|
| 2.1 | Beispielbilder aus dem Datensatz | 2 |
| 2.2 | Klassenverteilung der rohen Daten | 3 |
| 2.3 | Aufteilung in Trainings-, Validation- und Testdaten | 4 |
| 3.1 | Übersicht über die Architektur von YOLOX | 5 |
| 3.2 | Illustration des Unterschieds zwischen dem Yolov3-Head und dem neuen Decoupled-Head | 7 |
| 4.1 | Übersicht über die Architektur von YOLOv8 | 8 |

Kapitel 6

Anhang

6.1 Ordnerstruktur

6.2 Dateien: Datensatz

6.3 Dateien: YOLOX

6.4 Dateien: YOLOv8