

Bildverarbeitung 2

Verkehrsschilderkennung

Gruppe 4



Studiengang Mechatronik und Robotik
Teilnehmer Marco Menner, Benedikt Seeger
Matr.-Nr. 208778, 208799
Zeitraum Sommersemester 2024

Inhaltsverzeichnis

1 Einführung	2
2 Grundlagen	3
2.1 Bildverarbeitung mit KI	3
2.1.1 Convolutional Neural Networks	3
2.1.2 Objekt Detektierung	4
2.1.3 Yolov8	4
3 Datensatz	6
3.1 Betrachtung vorhandener Datensätze	6
3.2 Erstellung eines Datensatzes	7
3.2.1 Verwendete Klassen	7
3.2.2 Definition des Hintergrunds	8
3.2.3 Augmentierung	8
3.3 Generierung	10
4 Training des Modells	11
4.1 Augmentierung	12
5 Evaluierung und Anpassungen	13
5.1 Anpassung der Datensatz Generierung	15
5.2 Negatives Sampling	15
5.3 Klassifizierung der Geschwindigkeitsschilder	16
6 Verarbeitung und Visualisierung	18
6.1 Frame-basierte Validierung und Caching	18
6.2 Geschwindigkeitsschilder	18
6.3 Schilderbox	19
7 Zusammenfassung	20
7.1 Herausforderungen	20
7.2 Ausblick	20
7.3 Fazit	20

1 Einführung

Im Rahmen der Vorlesung Bildverarbeitung 2 an der Hochschule Heilbronn haben wir uns intensiv mit der KI-basierten Erkennung von Verkehrsschildern beschäftigt. Für das autonome Fahren ist die Echtzeit-Erkennung von Verkehrszeichen ein kritischer Schritt, um Informationen über Vorfahrtsregeln, Gefahren und Geschwindigkeitsbeschränkungen bereitzustellen. Ziel unserer Arbeit ist es, ein System zu entwickeln, das Verkehrsschilder in Bildern zuverlässig erkennt und identifiziert. Dieses System soll auch unter schwierigen Bedingungen robust arbeiten und den Fahrer aktiv unterstützen. In einer realen Anwendung ist eine geringe Berechnungszeit pro Bild erforderlich, um rechtzeitig auf erkannte Verkehrsschilder reagieren zu können. Diese Anforderung gilt somit auch für unser System.



(a) Gealtertes Verkehrschild



(b) Durch Begrünung verdecktes Verkehrschild

Abbildung 1: Beispiele für Verkehrszeichen unter extremen Einflüssen

Die Erkennung von Verkehrszeichen in der realen Welt stellt diverse Herausforderungen dar. Schilder können teilweise durch Pflanzen oder andere Objekte verdeckt sein oder durch Umwelteinflüsse verblassen. Die Lichtverhältnisse ändern sich kontinuierlich durch den Tag-Nacht-Zyklus, und verschiedene Beleuchtungssituationen durch Schattenwurf und Wetterbedingungen erschweren die Erkennung zusätzlich. Da Verkehrsschilder reflektieren und der Kamerasensor nur eine begrenzte dynamische Spannweite zwischen dem hellsten und dunkelsten Punkt erfassen kann, kann es zu Sättigung und damit zu Detailverlusten kommen. Außerdem sind die Schilder im Straßenverkehr nicht immer perfekt ausgerichtet oder gerade. Hinzu kommt, dass die Umgebungen stark variieren und detailliert sein können.

Unser System muss daher in der Lage sein, unter diesen Bedingungen zuverlässig zu arbeiten und eine robuste Erkennung der Verkehrsschilder zu gewährleisten.

2 Grundlagen

2.1 Bildverarbeitung mit KI

Die Bildverarbeitung mit Künstlicher Intelligenz (KI) ermöglicht es, komplexe Muster und Merkmale in visuellen Daten zu erkennen und zu analysieren. Durch den Einsatz von neuronalen Netzen können Maschinen lernen, visuelle Informationen auf eine Weise zu interpretieren, die der menschlichen Wahrnehmung ähnelt. Dies führt zu Anwendungen in verschiedenen Bereichen wie der medizinischen Bildanalyse, der automatisierten Überwachung, der Gesichtserkennung und der Objekterkennung in Echtzeit.

2.1.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) sind neuronale Netzwerke, die besonders effektiv in der Verarbeitung von Bilddaten sind. Sie verwenden Filter (auch als Convolutional Kernels bekannt), um über die Bilder hinweg zu wandern und lokale Muster wie Kanten, Ecken und Texturen zu extrahieren. Der Hauptvorteil von CNNs liegt in ihrer Fähigkeit, durch das Training auf großen Datensätzen komplexe Merkmalsrepräsentationen zu erlernen, was zu besonders guten Leistungen in Aufgaben wie der Bildklassifikation, Objekterkennung, Segmentierung und anderen Bildverarbeitungsaufgaben führt. Durch das Wandern der Filter über das Bild werden außerdem Parameter wiederverwendet und Translationsinvarianz kann damit elegant gelöst werden. Translationsinvarianz ist die Fähigkeit ein Objekt im Bild unabhängig von seiner Position zu erkennen. Damit sind CNNs für die Bildverarbeitung im Vergleich zu herkömmlichen neuronalen Netzwerken effizienter und weniger anfällig für Overfitting.

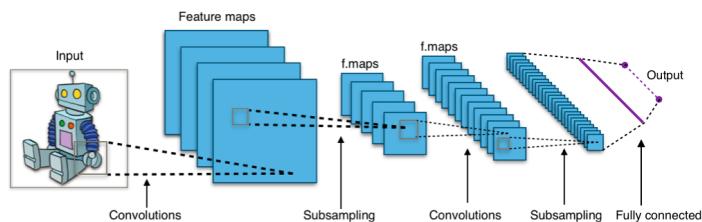


Abbildung 2: Übersicht einer CNN Architektur

2.1.2 Objekt Detektierung

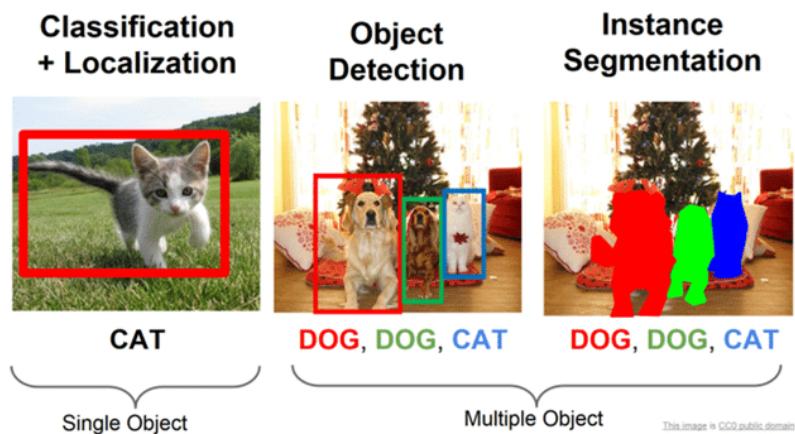


Abbildung 3: Die Aufgaben der Objekterkennung

Es gibt verschiedene Ansätze zur Erkennung von Objekten in Bildern, darunter Detektion, Lokalisierung, Klassifizierung und Segmentierung. Detektion umfasst die Identifikation und Lokalisierung von Objekten innerhalb eines Bildes durch Begrenzungsrahmen (Bounding Boxes). Klassifizierung bezieht sich auf die Zuordnung eines gesamten Bildes oder einzelner Objekte zu vordefinierten Klassen. Segmentierung ordnet jedes Pixel eines Bildes einer bestimmten Klasse zu und ermöglicht eine präzise Abgrenzung der Formen und Konturen der Objekte. Diese Techniken ermöglichen es, nicht nur zu identifizieren, welche Objekte in einem Bild vorhanden sind, sondern auch, wo sie sich befinden und wie sie genau geformt sind. Für unser Projekt ist es entscheidend, zu identifizieren, welche Objekte zu unseren spezifischen Klassen gehören und deren genaue Positionen in der Szene zu bestimmen. Zudem ist es wichtig, dass mehrere Objekte gleichzeitig in einem Bild erkannt werden können. Deshalb wurde die Objektdetektion gewählt. Obwohl auch die Segmentierung möglich gewesen wäre, ist sie insgesamt rechenaufwendiger und die genaue Form der Objekte ist für unser Projekt nicht relevant.

2.1.3 Yolov8

YOLOv8 (You Only Look Once) ist eine der neuesten Versionen einer Reihe von Echtzeit-Objektdetektionsmodellen, die darauf abzielen, schnelle und präzise Erkennungen und Lokalisierungen von Objekten in Bildern und Videos zu ermöglichen. Das YOLO-Framework ist bekannt für seine Fähigkeit, in einem einzigen Durchlauf (Single Shot) durch das Netzwerk sowohl die Klassifizierung als auch die Lokalisierung von Objekten durchzuführen, was es besonders effizient macht.

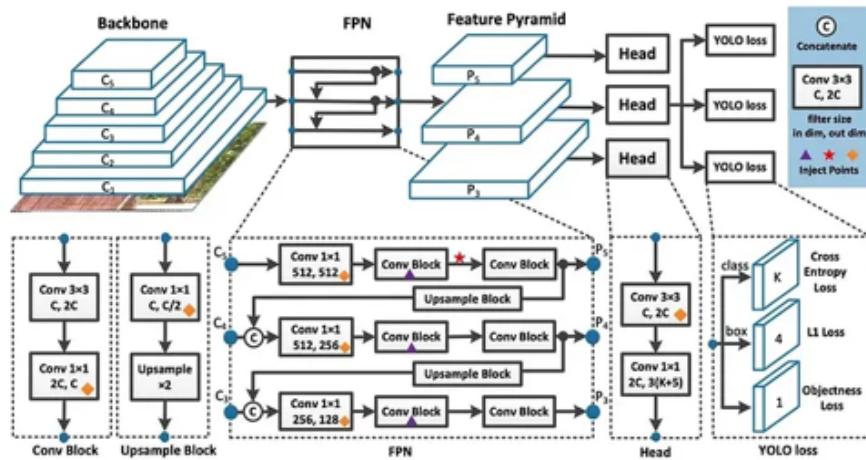


Abbildung 4: Schematischer Aufbau der YOLOv8 Architektur

Für unsere Anwendung haben wir uns entschieden, YOLOv8n zu verwenden. YOLOv8n ist die Nano-Variante der YOLOv8-Reihe und bietet eine Balance zwischen Leistung und Ressourcenbedarf, was es ideal für den Einsatz in Umgebungen mit begrenzten Hardware-Ressourcen macht. Diese Variante zeichnet sich durch ihre geringe Größe und schnelle Inferenzzeiten aus, was sie besonders für Echtzeitanwendungen geeignet macht, bei denen eine schnelle Verarbeitung erforderlich ist. YOLOv8n wurde auf einer Vielzahl von Objektklassen trainiert, um eine breite Palette von Objekten erkennen zu können. Um das Modell nun auf das Problem der Verkehrsschilderkennung anwendbar zu machen wird das Konzept des Transfer Learning benutzt, bei dem das Modell zunächst auf einem großen, allgemeinen Datensatz vortrainiert wurde, um grundlegende Merkmale zu lernen. Anschließend wurde es auf spezifischere Datensätze feinabgestimmt, um die Leistung in der Zielanwendung zu optimieren. Dieses Verfahren ermöglicht es dem Modell, von den bereits gelernten allgemeinen Merkmalen zu profitieren und sich schneller und effizienter an neue Aufgaben anzupassen. Ein wesentlicher Fortschritt von YOLOv8 im Vergleich zu früheren YOLO-Architekturen ist die Einführung der Mosaik-Augmentierung. Diese Technik verbessert die Generalisierungsfähigkeit des Modells, indem sie Bilder aus mehreren zufällig ausgewählten Trainingsbeispielen kombiniert und zu einem neuen Bild zusammenfügt. Dadurch wird die Variation der Trainingsdaten erhöht, was zu einer besseren Erkennungsleistung führt. Darüber hinaus wurden auch Verbesserungen an der Netzarchitektur und den Verlustfunktionen vorgenommen, um die Genauigkeit und Effizienz weiter zu steigern.

Insgesamt bietet YOLOv8 durch diese Verbesserungen eine erheblich bessere Leistung und Genauigkeit im Vergleich zu seinen Vorgängern, was es zu einer ausgezeichneten Wahl für moderne Objektdetectionsaufgaben macht.

3 Datensatz

3.1 Betrachtung vorhandener Datensätze

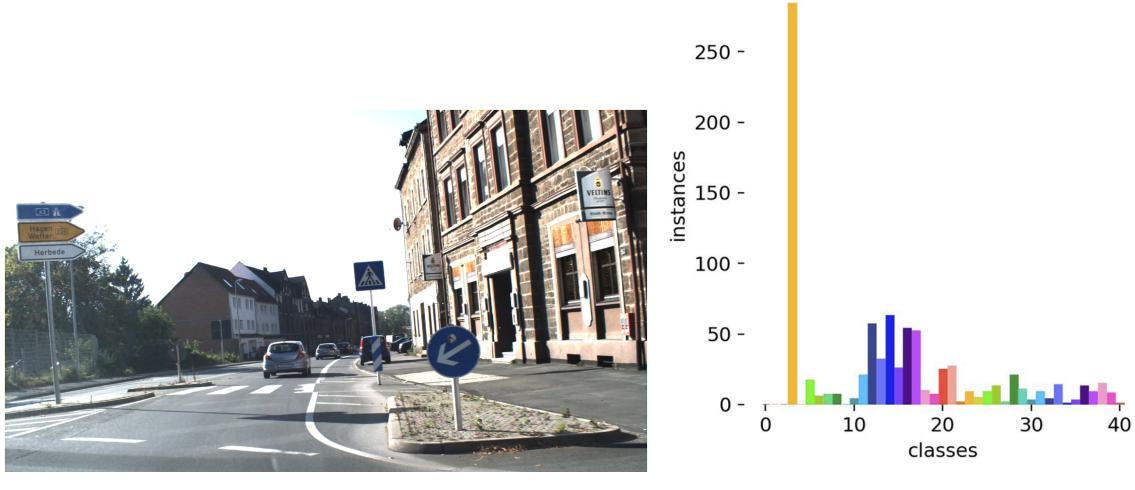
Es gibt für deutsche Verkehrssituationen und Schilder zwei öffentliche Datensätze, welche als Maßstab gelten. Diese wurden im Rahmen eines Wettbewerbs von der Universität Bochum veröffentlicht.

GTSRB Der German Traffic Sign Recognition Benchmark(GTSRB) ist ein gelabelter Datensatz mit Bildern von deutschen Verkehrszeichen. Er besteht aus mehr als 50000 Bildern in 43 Klassen. Die Bilder sind 15 x 15 bis 250 x 250 Pixel groß. Darauf sind die Verkehrszeichen mit verschiedenen Beleuchtungssituationen und Hintergründen sichtbar. Dieser Datensatz eignet sich für das Training eines Klassifizierungsmodells. Für unsere Anwendung haben diese Bilder leider zu wenig Hintergrund und eine Lokalisation in einer Szene wird damit nicht ausreichend erlernt. Jedoch ist dieser für seine Bekanntheit in diesem Gebiet erwähnenswert.



Abbildung 5: Auszug aus dem GTSRB Datensatz

GTSDB Der German Traffic Sign Detection Benchmark(GTSDB) ist ein gelabelter Datensatz mit Szenen aus deutschen Straßen. Diese enthalten bis zu sechs Verkehrszeichen, welche alle gelabelt wurden. Der Datensatz besteht aus 600 Bildern und die Verkehrszeichen darin variieren in der Größe von 16x16 bis 128x128 Pixel. Die Beleuchtungssituationen und Wetterbedingungen der Szenen sind stark unterschiedlich und versuchen die Variation der Realität widerspiegeln. Dieser Datensatz zielt darauf ab, Verkehrsschilder in einer Szene ausfindig machen zu können. Die Klassenhäufigkeit des Datensatzes ist ungleich verteilt und macht es für das Training komplexer Klassen mit nur wenig Bildern schwer.



(a) Auzug aus dem GTSDB Datensatz

(b) Klassenhäufigkeit

Abbildung 6: Übersicht des GTSDB Datensatzes

3.2 Erstellung eines Datensatzes

Um die Datenlage zu verbessern wurde sich dafür entschieden synthetische Trainingsdaten zu generieren, die für notwendigen Anforderungen zugeschnitten sind. Dafür werden Bilder von Verkehrsschildern auf einen zufälligen Hintergrund kopiert und die dazugehörigen Bounding Boxes generiert.

3.2.1 Verwendete Klassen

Insgesamt wird zwischen 45 Klassen entschieden. Für jede Klasse gibt es eine Grafik und mindestens ein Bild aus einem realen Foto. Die Bilder wurden so ausgeschnitten, dass der Hintergrund transparent ist. Damit können die Bilder auf einen Hintergrund kopiert werden und es ist kein Bildrand sichtbar.

Tabelle 1: Übersicht der Verkehrsschildern

Klassen Art	Schilder Anzahl	Beispiel
Geschwindigkeitsschilder	12	50 kmh
Richtungsschilder	8	Gerade aus
Gefahrenschilder	14	Steinschlag
Verbotsschilder	7	Überholverbot
Vorfahrtsschilder	4	Vorfahrtsstraße
Gesamt	45	

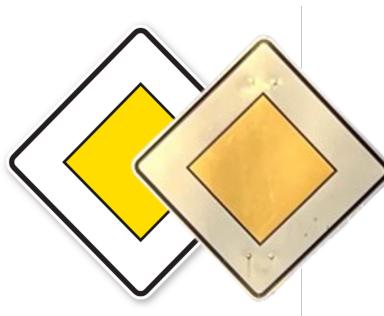


Abbildung 7: Darstellung beider Bildarten, links eine Vektorgrafik, rechts ein Bild ohne Hintergrund.

3.2.2 Definition des Hintergrunds

Für das Training ist es wichtig, dass der Hintergrund komplex ist. Ein einfarbiges Bild für den Hintergrund würde das Modell nicht auf eine reale Anwendung vorbereiten. Des Weiteren soll das Modell keine Eigenschaften des Hintergrunds erlernen. Für unser Datensatz werden Bilder des COCO (Common Objects in Context) Datensatzes verwendet.



Abbildung 8: Auszug aus dem COCO Datensatz

3.2.3 Augmentierung

Die Bilder der Verkehrsschilder werden vor dem Kopieren auf den Hintergrund noch verändert. Damit werden unterschiedliche, in der Realität auftretende Effekte simuliert. Jeder dieser Effekte wird zufällig in verschiedener Intensität angewandt.

1. Ausgangsbild ohne Bearbeitung
2. Reduzierte Schärfe
3. Perspektivische Verzerrung
4. Veränderte Helligkeit und Kontrast
5. Farbverschiebung

Nach der Bearbeitung muss der Alpha Kanal wieder als Maske über das Bild gelegt werden, da manche der Bearbeitungsalgorithmen keinen Transparenzkanal zulassen oder diesen überschreiben. Dieser wird vor der Augmentierung separiert.

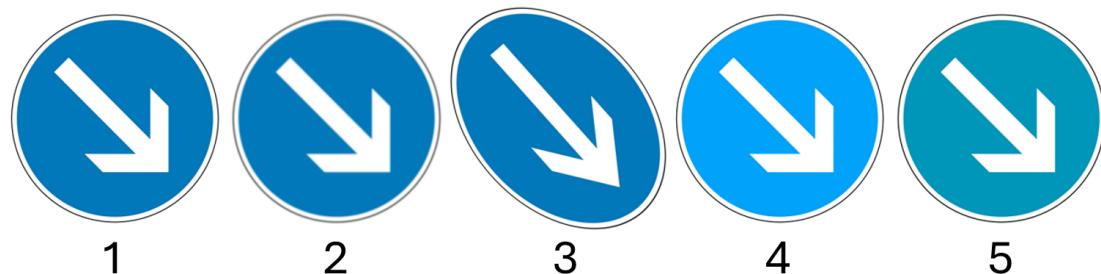


Abbildung 9: Übersicht der verschiedenen Bearbeitungsschritte auf ein Bild angewandt

3.3 Generierung

Für die Erstellung von Trainingsbildern werden zufällig 5 bis 10 augmentierte Verkehrs-schilder auf einem Hintergrundbild platziert. Größe und Position der Schilder sind dabei zufällig gewählt. Zu jedem kopierten Schild wird auch die zugehörige Bounding Box generiert. Dadurch wird in jedem Bild eine unterschiedliche Straßensituation simuliert. Obwohl dies für den Menschen nicht unmittelbar erkennbar ist, sind diese Trainingsdaten für das Modell sehr wertvoll. Das Modell lernt, welche Schilder an welchen Positionen im Bild vorhanden sind. Während der Generierung wurde darauf geachtet, dass sich die Schilder nicht überlappen.



Abbildung 10: Beispiel eines Trainingsbildes

4 Training des Modells

Nachdem ein Skript zur Generierung eines geeigneten Datensatzes erstellt wurde, beginnt das eigentliche Training des Modells. Dabei werden verschiedene Trainingsdatengrößen getestet, um deren Einfluss auf das Ergebnis zu ermitteln. Zudem wird die Anzahl der Epochen mehrfach variiert. Die Epochenzahl gibt an, wie oft das Modell den gesamten Trainingsdatensatz durchläuft, um seine Parameter zu optimieren. Jede Epoche verbessert die Fähigkeit des Modells, Muster zu erkennen und Vorhersagen zu treffen. Eine zu geringe Epochenzahl führt zu Unteranpassung, während zu viele Epochen Überanpassung verursachen können.

Letztlich wurde eine Datensatzgröße von 3000 Bildern gewählt. Dieser Datensatz wurde im Verhältnis 70/30 auf Trainings- und Testdatensatz aufgeteilt. Dabei wurden die 600 Bilder des GTSDB (German Traffic Sign Detection Benchmark) dem Testdatensatz beigemischt, um auch gegen reale Bilder zu testen. Für die finale Validierung wurden echte Videosequenzen verwendet. Im Folgenden wird erläutert, wie sich diese Datensätze unterscheiden und warum sie für die Evaluierung eines Machine Learning Modells notwendig sind:

Trainingsdatensatz: Dieser Teil des Datensatzes wird verwendet, um das Modell zu trainieren. Das Modell lernt aus diesen Daten und passt seine Parameter an, um die bestmöglichen Vorhersagen zu treffen.

Validierungsdatensatz: Dieser separate Teil des Datensatzes wird verwendet, um die Leistung des Modells während des Trainings zu überwachen und Hyperparameter zu optimieren. Er hilft dabei, das Modell zu validieren und Überanpassung zu verhindern.

Testdatensatz: Nach dem Training und der Optimierung wird dieser Teil des Datensatzes verwendet, um die endgültige Leistung des Modells zu bewerten. Er dient dazu, die Generalisierungsfähigkeit des Modells auf neue, unbekannte Daten zu überprüfen.

4.1 Augmentierung

Für das eigentliche Training wurden die Trainingsbilder zusätzlich augmentiert. Dabei lag der Fokus nicht wie zuvor darauf, die einzelnen Schilder in verschiedenen Varianten darzustellen, sondern die gesamte Szene, in der sich die Schilder befinden, zu verändern. Dies sollte das Modell weiter robuster gegenüber unterschiedlichen Umgebungen machen.

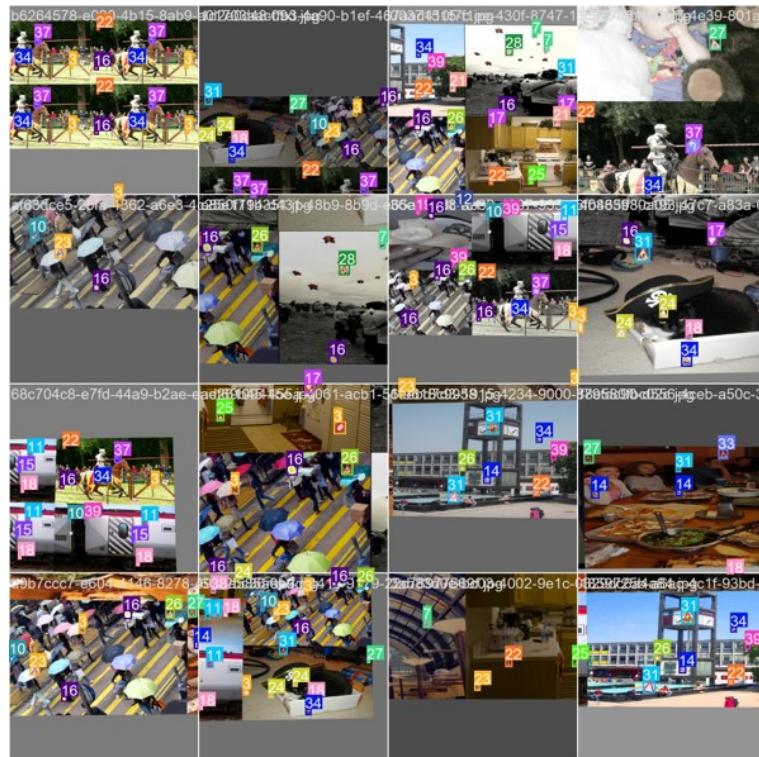


Abbildung 11: Augmentierung der Traingsdaten

Anhand des obigen Bildes wird die angewendete Augmentierung deutlich sichtbar. Zunächst wurde die in den Grundlagen erwähnte Mosaik-Augmentierung eingesetzt, bei der mehrere Bilder zu einem zusammengesetzt wurden. Zusätzlich kam die Copy-Paste-Funktion zum Einsatz, bei der Bilder zufällig in andere Bilder eingefügt werden. Diese Bilder wurden mithilfe der Scale-Augmentierung in verschiedenen Größen platziert. Um den Winkel der Bilder zu verändern, wurde die Perspective-Methode verwendet, die die Perspektive des Bildes transformiert. Abschließend wurde eine HSV-Transformation (Hue, Saturation, Value) angewendet, um Farbton, Sättigung und Helligkeit zu verändern.

5 Evaluierung und Anpassungen

Für das Training wurden sowohl die Confusion Matrix als auch ein Auszug der Verkehrsschilder zur visuellen Inspektion der Modellperformance verwendet. Unten ist die Confusion Matrix des finalen Modells dargestellt.

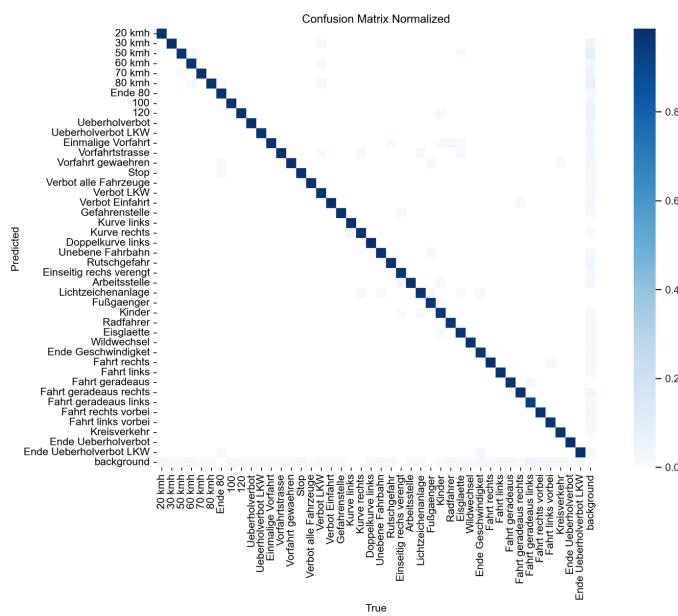


Abbildung 12: Confusion Matrix der Ergebnisse

Eine Confusion Matrix ist in folgende Fälle eingeteilt:

True Positives (TP): Fälle, in denen das Modell ein positives Ereignis korrekt vorhergesagt hat.

False Negatives (FN): Fälle, in denen das Modell ein positives Ereignis fälschlicherweise als negativ vorhergesagt hat.

False Positives (FP): Fälle, in denen das Modell ein negatives Ereignis fälschlicherweise als positiv vorhergesagt hat.

True Negatives (TN): Fälle, in denen das Modell ein negatives Ereignis korrekt vorhergesagt hat.

Die Performance ist optimal, wenn die Confusion Matrix eine diagonale Linie von oben links nach unten rechts bildet, da dies bedeutet, dass alle Objekte korrekt erkannt wurden. In diesem Fall gibt es keine Fehlklassifizierungen oder Falsch-Negativ-Erkennungen. Es ist

jedoch leicht zu erkennen, dass das Modell manchmal Schilder im Hintergrund erkennt, obwohl keine vorhanden sind. Diese Fehler werden jedoch durch geringfügige Anpassungen des Modells im weiteren Verlauf behoben.



Abbildung 13: Resultate der Detektion auf die Testbilder

In dem obigen Bild ist einmal die Erkennung des Modells auf die Testdaten dargestellt. Dabei wird gut sichtbar das, dass Modell mit hoher Wahrscheinlichkeit auf die dargestellten Schilder vorhersagt. Dabei sind auch keine Erkennung von Schildern ersichtlich die keine sind.

F1-Confidence Kurve des Trainings

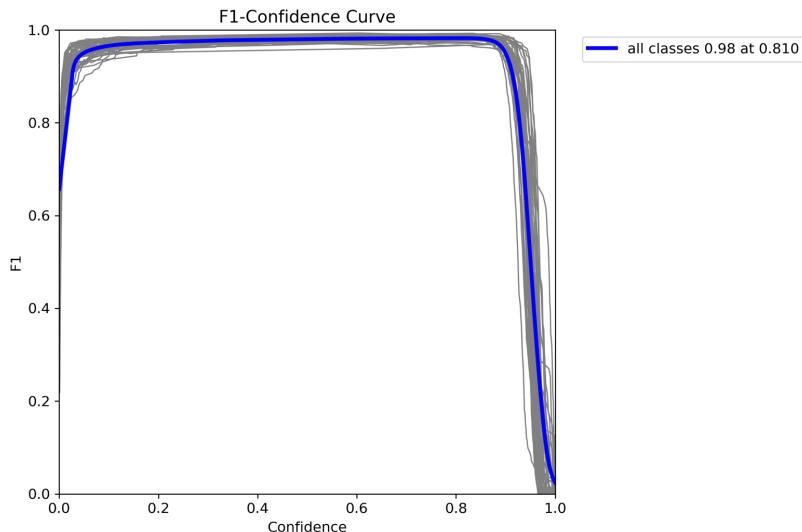


Abbildung 14: F-Confidence Kurve des Trainings

Die Abbildung 14 zeigt die F1-Confidence-Kurve des trainierten Bilddetektionsmodells. Die x-Achse repräsentiert die Confidence-Werte der Modellvorhersagen, die von 0 bis 1 reichen, während die y-Achse den F1-Score darstellt, welcher ebenfalls zwischen 0 und 1 variiert. Der F1-Score ist eine etablierte Metrik, die das harmonische Mittel aus Präzision und Recall darstellt und somit die Balance zwischen diesen beiden Metriken misst. Die grauen Linien im Hintergrund der Abbildung repräsentieren die F1-Confidence-Kurven für die einzelnen Klassen, die das Modell unterscheidet. Jede dieser Linien zeigt die Abhängigkeit des F1-Scores von den Confidence-Schwellenwerten für eine spezifische Klasse. Die dicke blaue Linie hingegen zeigt den durchschnittlichen F1-Score über alle Klassen hinweg in Abhängigkeit von den Confidence-Werten.

Besonders hervorzuheben ist der Punkt auf der blauen Linie, der bei einem Confidence-Wert von 0.810 einen durchschnittlichen F1-Score von 0.98 erreicht. Dieser hohe F1-Score deutet darauf hin, dass das Modell bei einem Confidence-Schwellenwert von 0.810 eine außerordentlich hohe Genauigkeit aufweist. Dies impliziert, dass das Modell in der Lage ist, die meisten seiner Vorhersagen mit hoher Präzision und hohem Recall zu treffen, wenn es nur Vorhersagen berücksichtigt, die mit mindestens 81% Confidence gemacht wurden. Die Konsistenz des Modells über verschiedene Klassen hinweg wird durch die Nähe der grauen Linien zur blauen Linie verdeutlicht. Dies zeigt, dass das Modell nicht nur im Durchschnitt, sondern auch in der Vorhersage einzelner Klassen eine hohe und stabile Leistung erbringt.

Der nächste Schritt war nun das Modell mit den Validierungsvideos zu evaluieren. Die dabei erkannten Probleme wurden im weiteren schrittweise behoben.

5.1 Anpassung der Datensatz Generierung

Der erste Punkt, der anhand der Validierungsvideos deutlich wurde, war die späte Erkennung der Bilder. Um dieses Problem zu beheben, wurde die Größe der Bilder, in der sie zufällig platziert wurden, reduziert. Dadurch sollte das Modell lernen, die Bilder früher und zuverlässiger zu erkennen.

5.2 Negatives Sampling

Ein weiteres Problem, das während der Evaluierung auftrat, war die fehlerhafte Erkennung von Schildern, die nicht im Datensatz enthalten waren, als eine der bekannten Klassen des Modells. Zudem wurden gelegentlich auch die Rückseiten von Schildern detektiert. Um dieses Verhalten zu unterbinden, wurden Bilder von diesen Schildern ebenfalls in den

Trainingsdatensatz aufgenommen und zufällig platziert. Im Gegensatz zu den anderen Schildern wurde jedoch keine Bounding Box definiert, wodurch der Hintergrund entsprechend für das Modell angepasst wurde.



Abbildung 15: Negatives Sampling der Verkehrsschilder

Hier sind zwei Beispiele der verwendeten Schilder, zum einen die Rückseite eines Schildes. Zum anderen ein Parken verboten Schild, welches für unser Problem nicht als relevant definiert wurde und deshalb nicht im Datensatz enthalten ist.

5.3 Klassifizierung der Geschwindigkeitsschilder

Weiterhin hatte das Modell Probleme die doch sehr ähnlichen Geschwindigkeitsschilder von einander zu unterscheiden. Aus diesem Grund wurde sich dazu entschieden bei diesem Schildertyp einen extra Schritt einzufügen.



Abbildung 16: Klassifizierung der Geschwindigkeitsklasse

Als erstes wurden alle Geschwindigkeitsschilder zu einer Klasse zusammengefasst. Sobald eines der Schilder dann erkannt wurde, schneidet das Programm die Boundingbox um das Schild aus. Im nächsten Schritt wird dann dieses Schild weiter klassifiziert und der tatsächlichen Geschwindigkeit zugeordnet. Für die Klassifizierung wurde in diesem Schritt

wieder ein Yolov8n Modell verwendet, das jedoch für Klassifizierungsaufgaben optimiert ist.

Der Grund, warum für den Klassifizierungsschritt nicht klassische Bildverarbeitungsmethoden verwendet wurden, liegt in der höheren Genauigkeit und Robustheit moderner neuronaler Netze. Klassische Bildverarbeitungsmethoden sind oft anfällig für Variationen in Beleuchtung, Perspektive und teilweise verdeckte Schilder. Ein neuronales Netz kann dagegen diese Variationen besser handhaben und liefert zuverlässigere Ergebnisse. Die Verwendung eines modernen YOLOv8n Modells gewährleistet somit eine präzisere und konsistenter Klassifizierung der Geschwindigkeitsschilder.

Natürlich führt dieser zusätzliche Klassifizierungsschritt zu einer etwas langsameren Erkennung. Allerdings ist diese geringere Geschwindigkeit durch die hohe Relevanz der korrekten Erkennung und Klassifizierung von Geschwindigkeitsschildern gerechtfertigt. In vielen Anwendungen, wie zum Beispiel im autonomen Fahren, ist die genaue Erkennung der Geschwindigkeitsschilder von entscheidender Bedeutung für die Sicherheit und die Einhaltung von Verkehrsregeln. Daher überwiegt der Vorteil der erhöhten Genauigkeit und Zuverlässigkeit die leichte Verzögerung in der Verarbeitung.

6 Verarbeitung und Visualisierung

Nachdem das Modell nun weiterhin optimiert wurde sollte im nächsten Schritt eine Programm zur Visualisierung und Verarbeitung der erkannten Schilder geschrieben werden. Insgesamt dauert die Inferenz der Schilder nun zwischen 0.060 und 0.070 Sekunden. Bei Geschwindigkeitsschildern durch die Klassifizierung 0.080 bis 0.090 Sekunden.

6.1 Frame-basierte Validierung und Caching

Um die Erkennung weiterhin robuster zu machen, wurde in dem Programm ein Caching implementiert. Damit wird ein Schild erst angezeigt, wenn es dreimal hintereinander detektiert wurde. Durch diesen Zusatzschritt werden einmalige Falscherkennungen vermieden und nur Schilder visualisiert, welche mehrfach übereinstimmend erkannt wurden. Dieser Cache wird nach ab dem vierten Frame wieder gelöscht.

6.2 Geschwindigkeitsschilder



Abbildung 17: Visualisierung der erkannten Schilder

Da wie schon zuvor erwähnt die Geschwindigkeitsschilder eine Sicherheitsrelevante Schilderart ist soll diese größer als die restlichen Schilder dargestellt werden. Weiterhin soll dieses Bild nach einer Erkennung konstant angezeigt werden bis ein neues Geschwindigkeitsschild erkannt wird.

6.3 Schilderbox



Abbildung 18: Augmentierung der Verkehrsschilder

Die restlichen Schilder werden in einer Box darstellt die sich fortlaufend aktualisiert wenn ein neues Schild erkannt wird. Sofern ein bereits angezeigtes Schild nochmals detektiert wird, kommt dieses wieder an den Anfang. Insgesamt werden immer die letzten 6 Schilder dargestellt.

7 Zusammenfassung

Zu Beginn des Projekts wurde ein Datensatz erstellt, indem Bilder von verschiedenen Schildtypen auf zufällige Hintergründe mit unterschiedlicher Augmentierung und Größe eingefügt wurden. Anschließend wurde die Yolov8-Architektur trainiert. Durch wiederholtes Training und Evaluieren wurde das Modell schrittweise angepasst, bis eine zufriedenstellende Genauigkeit erreicht war. Danach wurde ein Programm zur Verarbeitung und Visualisierung der Ergebnisse entwickelt, das einen Cache verwendet, um Falscherkennungen weiter zu minimieren. Die Geschwindigkeit des Programms wird permanent angezeigt und ist höher als die restlichen Detektionen. Alle anderen Schilder werden in einer Box mit sechs Plätzen angezeigt.

7.1 Herausforderungen

Das Projekt verlief insgesamt erfolgreich, doch der Prozess war sehr iterativ, bis der Datensatz genügend Variationen enthielt, um das Modell effizient trainieren zu können. Mit einem bereits bestehenden Datensatz wäre dieser Schritt deutlich einfacher gewesen, sodass mehr Fokus auf die Entwicklung weiterer Features gelegt werden konnte. Diese Erfahrung war dennoch wertvoll, da in Zukunft häufiger mit dem Fehlen eines Datensatzes gerechnet werden muss und so eine Methode gefunden wurde, um dennoch einen Datensatz zu erstellen, ohne die Szene mehrfach aufnehmen zu müssen.

7.2 Ausblick

Zur Verbesserung des Programms könnten weitere Schildtypen hinzugefügt werden, die derzeit noch nicht im Datensatz enthalten sind. Außerdem wäre es denkbar, den Zustand von Ampeln zu tracken und ebenfalls im Overlay darzustellen. Interessant wäre es auch, eine Region-of-Interest zu definieren, sodass nur relevante Schilder für den Fahrer angezeigt werden, beispielsweise wenn ein Rechtsabbiegen geplant ist. Eine weitere Optimierung zur Steigerung der Modellschnelligkeit wäre die Implementierung der Geschwindigkeitserkennung oder sogar der gesamten Verkehrsschilderkennung in C++. Dadurch könnte die Erkennungsrate weiter erhöht werden.

7.3 Fazit

Das Projekt bot eine wertvolle Gelegenheit, Einblicke in die Objektdetektion und die Arbeit mit Convolutional Neural Networks zu gewinnen. Besonders wichtig war die Vorverarbeitung der Daten und die Wahl des richtigen Ansatzes. Zudem konnten weitere

Kompetenzen im strukturierten Arbeiten und in der Durchführung eines Forschungsprojekts für zukünftige Arbeiten gesammelt werden.