

Vorgelegt an der Universität Stuttgart
Institut für Energieübertragung und
Hochspannungstechnik



KI-basierte Analyse von Satellitendaten zur Bestimmung der Auswirkungen von Elektromobilität an Autobahnraststätten

**AI-based analysis of satellite data to determine
the impact of electromobility at highway rest
areas**

Fachpraktikum Energieübertragung

von

Patrick Ernst, 3388020

Diego Kuderna Melgar, 3683217

Tobias Schmalzried, 3679072

Beginn der Arbeit: 11.04.2023

Ende der Arbeit: 29.09.2023

Betreuer: Nelly-Lee Fischer, M. Sc.
Kathrin Walz, M. Sc.

Prüfer: Prof. Dr.-Ing. Krzysztof Rudion

Erklärung

Wir versichern, dass wir die Arbeit selbstständig durchgeführt und verfasst haben, abgesehen von den Anregungen, die uns von Seiten unserer Betreuerinnen Kathrin Walz und Nelly-Lee Fischer gegeben worden sind, und dass wir keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben.

Stuttgart, 28. September 2023

Ort, Datum

Patrick Ernst

Winterbach, 28. September 2023

Ort, Datum

Diego Kuderna Melgar



Dublin, 28. September 2023

Ort, Datum

Tobias Schmalzried

Kurzfassung

Das IEH untersucht im Rahmen des Projekts BANULA unter anderem die Auswirkungen der Elektrifizierung von Lastwagen auf das Übertragungsnetz. In diesem Zuge befasst sich diese Arbeit mit der Anwendung von Methoden des *Deep Learnings* auf Satellitendaten, um die Auslastung baden-württembergischer Autobahnraststätten zu analysieren. Dazu wurde ein YOLOv5-Modell realisiert, das LKW mit 89,5 % Genauigkeit erkennt. Die eingesetzten Satellitendaten aus Baden-Württemberg weisen ähnliche Auslastungsmuster wie bayrische Raststätten auf. Die Gewinnung von Zeitreihen aus Satellitendaten ist derzeit allerdings nicht möglich. Jedoch lassen sich die gewonnenen Daten künftig für weitere *Deep Learning* Ansätze nutzen.

Abstract

The IEH is investigating, among other things, the effects of the electrification of trucks on the transmission network, as part of the BANULA project. In this context, this work deals with the application of deep learning methods to satellite data to analyze the utilization of motorway service stations in Baden-Württemberg. For this purpose, a YOLOv5 model was implemented that detects trucks with 89.5 % accuracy. The satellite data used from Baden-Württemberg shows similar utilization patterns to Bavarian rest stops. However, obtaining time series from satellite data is currently not possible. However, the data obtained can be used for further deep learning approaches in the future.

Inhaltsverzeichnis

Kurzfassung	ii
Abstract	iii
Abbildungsverzeichnis	vi
Tabellenverzeichnis	ix
Formelzeichen und Darstellungskonventionen	xi
Formelzeichen	xi
Akronyme	xii
1. Einleitung	1
1.1. Einführung	1
1.2. Aufgabenstellung und Zielsetzung	2
1.3. Struktur der Arbeit	2
2. Grundlagen	4
2.1. Objekterkennung mithilfe von Deep Learning	4
2.2. Konvolutionsnetze	5
2.3. Rekurrente neuronale Netze	6
2.4. Intersection over Union und Jaccard-Index	6
2.5. Mean Average Precision	7
3. Verwendete Technologien	9
3.1. YOLOv5	9
3.1.1. Einführung in YOLO	10
3.1.2. Netzwerkarchitektur	10

3.1.3. Bounding Box-Vorhersagen	11
3.1.4. Trainingsprozess	12
3.2. Long Short-Term Memory	13
3.2.1. Aufbau einer LSTM-Zelle	13
3.2.2. LSTMs für Zeitreihen	14
3.3. Python-Frameworks für Deep Learning	15
3.3.1. PyTorch	15
3.3.2. TensorFlow und Keras	15
3.4. CubeSat Satelliten	16
3.4.1. PlanetScope	16
3.4.2. RapidEye	17
3.4.3. SkySat	18
3.4.4. Earthnet Programm	19
3.4.5. Google Earth	20
4. Praktische Umsetzung	21
4.1. Beschaffung der Satellitenbilder	21
4.2. Parkplatzbelegungsdaten aus Baden-Württemberg	25
4.3. Aufbereitung des Datensatzes für die Objekterkennung	27
4.3.1. Sichtung der Daten	28
4.3.2. Formatierung der Daten	28
4.3.3. Labeln der Daten	28
4.3.4. Aufteilung der Daten	30
4.3.5. Bestimmung von Hintergrundbildern	30
4.3.6. Generierung des Datensatzes	32
4.4. Objekterkennung der Fahrzeuge	33
4.4.1. Wahl eines Modells für die Objekterkennung	33
4.4.2. Training und Validierung des Netzes	34
4.4.3. Test des Modells	40
4.5. Berechnung eines Korrekturfaktors	42
4.6. LSTM für die Vorhersage von Belegungsdaten	44
4.6.1. Aufbereitung der Daten	44
4.6.2. Training des LSTM	45

5. Ergebnisse der Arbeit	51
5.1. Erwarteter zeitlicher Verlauf der Parkplatzauslastung	51
5.2. Bestimmung des Ladebedarfs einer durchschnittlichen Raststätte	53
5.2.1. Szenario 1: Laden in Lenk- und Ruhepause	54
5.2.2. Szenario 2: Laden in Standzeit	56
5.3. Fazit	59
6. Zusammenfassung und Ausblick	60
Literaturverzeichnis	63
Anhang	67
A. Unterstützende Grafik	67
B. Trainingsergebnisse	67

Abbildungsverzeichnis

2.1. Teilgebiete der KI	5
3.1. Architektur von YOLO [11]	10
3.2. Verschiedene Größen des YOLOv5-Modells [12]	11
3.3. Aufbau einer <i>LSTM</i> Zelle mit einem langanhaltenden (<i>f</i>), einem langanhaltenden (<i>i</i>) und einem <i>Output-Gate</i> (<i>o</i>) [16]	14
3.4. <i>PlanetScope</i> -Satellit [26]	17
3.5. <i>RapidEye</i> Satelliten [28]	18
3.6. <i>SkySat</i> Satelliten [30]	19
4.1. Beispiel einer Satellitenaufnahme der Autobahnraststätte <i>Kraichgau</i> am 24.07.2022 (<i>PlanetScope</i> -Satelliten)	22
4.2. Beispiel einer Satellitenaufnahme der Autobahnraststätte <i>Kraichgau</i> am 24.07.2022 (<i>SkySat</i> -Satelliten)	23
4.3. Beispiel einer Satellitenaufnahme der Autobahnraststätte <i>Kraichgau</i> am 05.07.2016, bereitgestellt von <i>Google Earth</i> <i>Pro</i>	24
4.4. Lage der Raststätten entlang der A5. Ausgewertet wurden Bad Bellingen (grau), Fischergrund (gelb), Blauenblick-West (blau), Streitkopf (Orange) und Neuenburg-West. Erstellt mit: <i>Google My Maps</i>	26
4.5. Pipeline für die Generierung eines Datensatzes	27
4.6. Auszug aus <i>makesense.ai</i> während der Erstellung der Labels	29
4.7. Hintergrundbild, das im Training verwendet wurde	31

4.8. Lage der Raststätten: Augsburg (beige), Bruchsal (rot), Burgauer See (blau), Denkendorf (schwarz), Ellwanger Berge (gelb), Gruibingen (hellblau), Hohenlohe (orange), Illetal (orange), Jagsttal (hellgrau), Leipheim (grau), Lonetal (ocker), Neckarburg (rosa), Pforzheim (grün), Satteldorf (dunkelblau), Sindelfinger Wald (violett) und Wunnenstein (weiß); Erstellt mit: <i>Google My Maps</i>	32
4.9. Beispiel einer Trainingsbatch, die aus augmentierten Bildern besteht. LKW werden mit 0 annotiert, PKW mit 1	36
4.10. Trainingsmetriken des besten Modells aufgetragen über die Trainingsepochen, links oben: <i>Precision</i> , rechts oben: <i>Recall</i> , links unten: <i>mAP mit Threshold = 0,5</i> , rechts unten: <i>mAP mit Threshold = 0,95</i>	37
4.11. Konfusionsmatrix der Trainingsdaten. LKW werden zu 92 % korrekt erkannt	38
4.12. Auszug aus der automatischen Validierung des YOLOv5-Netzes	40
4.13. Testergebnis der Raststätte <i>Fläming</i> mit YOLOv5	41
4.14. Resultierende Zeitreihe des Trainings im Vergleich zur originalen Zeitreihe	46
4.15. Mit dem Modell vorhergesagte Zeitreihe	47
4.16. Modifizierte Zeitreihe	48
4.17. Trainingsergebnisse mit der modifizierten Zeitreihe	49
4.18. Vorhersageergebnisse mit der modifizierten Zeitreihe	50
5.1. Durchschnittlicher Verlauf der 20 bayrischen Raststätten (blau), auf der x-Achse: Wochentag, auf der y-Achse: relative Auslastung des Parkplatzes, zusätzlich baden-württembergische Punkte (rot), in Anlehnung an [40]	52
5.2. Histogramm, auf der x-Achse: Ladeleistung in kW, auf der y-Achse: relative Häufigkeit	55
5.3. Boxplot für Szenario 1, auf der x-Achse: Durchschnittliche Raststätte in Baden-Württemberg, auf der y-Achse: Ladeleistung in kW	56

5.4. Histogramm für Szenario 2, auf der x-Achse: Ladeleistung in kW, auf der y-Achse: relative Häufigkeit	57
5.5. <i>Boxplot</i> für Szenario 2, auf der x-Achse: Durchschnittliche Raststätte in Baden-Württemberg, auf der y-Achse: Lade- leistung in kW	58
A.1. Grafische Darstellung der <i>Intersection over Union</i>	67
B.1. Trainingsmetriken des YOLOv5-Modells mit 320x320 Pixel Eingabeauflösung	68
B.2. Trainingsmetriken des YOLOv5 Modells mit 640x640 Pixel Eingabeauflösung	68
B.3. Trainingsmetriken des YOLOv5 Modells mit 960x960 Pixel Eingabeauflösung	69
B.4. Trainingsmetriken des YOLOv5 Modells mit 1080x1080 Pi- xel Eingabeauflösung	70
B.5. Trainingsmetriken des YOLOv5-Modells in der Größe <i>XLarge</i> mit 960x960 Pixel Eingabeauflösung	70
B.6. Labelmetriken des 960x960 Pixel Modells der Größe <i>XLarge</i>	71
B.7. Korrelogramm der Labels des 960x960 Pixel Modells der Größe <i>XLarge</i>	72
B.8. Trainingsmetriken des YOLOv5-Modells der Größe <i>Large</i> mit 960x960 Pixel Eingabeauflösung, Datensatz ausschließ- lich aus <i>SkySat</i> -Aufnahmen	73
B.9. <i>mAP</i> über den Verlauf der Epochen eines Modells ohne <i>Early Stopping</i>	73

Tabellenverzeichnis

4.1. Die Parkplatzbelegung der Raststätte <i>Denkendorf</i> von YO-L Ov5 und von Hand bestimmt	43
---	----

Formelzeichen und Darstellungskonventionen

Formelzeichen

Symbol	Beschreibung	Einheit
AP	Average Precision	%
$Confidence$	Vertrauenswert	
IoU	Intersection over Union	
J	Jaccard Index	%
mAP	Mean Average Precision	%
$Precisions$	Präzision	%
$Recalls$	Erkennungsrate	%
SSE	Summed squared error	

Akronyme

Akronym	Bedeutung
AP	Average Precision
API	Application Programming Interface
BANULA	BA rrierefreie und N utzerfreundliche L ademöglichkeiten schaffen
BW	B aden- W ürttemberg
CNN	C onvolutional N eural N etwork
CPU	C entral P rocessing U nit
ESA	E uropean S pace A gency
GPU	G raphics P rocessing U nit
IEH	I nstitut für E nergieübertragung und H ochspannungstechnik
IoU	I ntersection o ver U nion
KI	K ünstliche I ntelligenz
LKW	L astkraftwagen
LSTM	L ong S hort-Term M emory
mAP	m ean A verage P recision
NASA	N ational A eronautics and S pace A dministration
PKW	P ersonenkrafwagen
R-CNN	R egion-based C onvolutional N eural N etwork
RNN	R ekurrent N eural N etwork
RPN	R egional P roposal N etwork
SSD	S ingle S hot D etector
SVZ BW	S traßenverkehrszentrale B aden- W ürttemberg
YOLO	Y ou O nly L ook O nce

1. Einleitung

Dieses Kapitel 1 umfasst die Einführung in die Thematik in Kapitel 1.1 und stellt die Aufgabenstellung sowie Zielsetzung in Kapitel 1.2 vor. Die Erläuterung der Struktur der Arbeit in Kapitel 1.3 stellt den Abschluss des Kapitels dar.

1.1. Einführung

Die Elektrifizierung von Lastkraftwagen (LKW) sowie Personenkraftwagen (PKW) stellt eine Möglichkeit zur Erreichung langfristiger Klimaschutzziele dar, welche sich unter [1] finden lassen. Dabei muss das bestehende Stromnetz auf den erhöhten Ladebedarf vorbereitet werden, damit ein zuverlässiges Aufladen der künftigen Fahrzeuge gewährleistet werden kann. Überdies stellt die Integration eines hohen Anteils erneuerbarer Energien ein erstrebenswertes Ziel dar.

Unterbrechungen von kurzer Dauer im Fernverkehr fordern gesteigerte Ladeleistungen an Autobahnraststätten, was zu starken Belastungen der dortigen Netzanschlüsse führt. Aufgrund der noch geringen Durchdringung von Elektrofahrzeugen fehlt den Netzbetreibern jedoch eine Datenbasis zur Abschätzung des Ladebedarfs für die Netzplanung. Außerdem erforscht das IEH im Kontext des Projekts BANULA die Auswirkungen der Ladevorgänge auf das Übertragungsnetz.

1.2. Aufgabenstellung und Zielsetzung

Zurzeit ist die Netzplanung aufgrund der geringen Verfügbarkeit von Daten nur eingeschränkt möglich. Im Rahmen dieser Arbeit wird die Verfügbarkeit zeitlich aufgelöster Satellitendaten von Autobahnraststätten untersucht und gegebenenfalls eine Datenbasis erstellt. Daneben werden Methoden zur Bilderkennung von LKW sowie PKW auf den Satellitenbildern analysiert. Dazu wird ein Bilderkennungsalgorithmus auf Basis einer künstlichen Intelligenz (KI) zur automatisierten Identifikation von parkenden PKW und LKW an einer Raststätte umgesetzt. Jener Algorithmus soll auf zeitlich aufgelöste Datensätze von Satellitenbildern anwendbar sein, um Parkplatzbelegungskurven ableiten zu können.

Im Anschluss an die Validierung der Methodik mithilfe von vorhandenen Parkplatzbelegungsdaten erfolgt die Anwendung auf andere Standorte in Baden-Württemberg sowie Bayern. Ein elementares Ziel dieser Arbeit ist somit zu untersuchen, ob sich mithilfe von Satellitendaten Zeitreihen für den Tagesverlauf baden-württembergischer Raststätten erstellen lassen. Die ermittelten Parkplatzbelegungen lassen sich schließlich zur Bestimmung des zeitlich aufgelösten Ladebedarfs von PKW und LKW an der spezifischen Autobahnraststätte einsetzen. Daneben lassen sich Netzplanungsansätze zur Auslegung des Netzanschlusses einer Autobahnraststätte ableiten sowie verschiedene Ladeleistungen und die Integration von erneuerbaren Energien untersuchen.

Außerdem wird die Fragestellung beantwortet, ob neben Satellitendaten alternative Quellen für Parkplatzbelegungsdaten baden-württembergischer Autobahnraststätten existieren.

1.3. Struktur der Arbeit

Zu Beginn dieser Arbeit wird in Kapitel 2 auf die theoretischen Grundlagen künstlicher Intelligenz eingegangen. In diesem Kontext werden die Teilgebiete der KI aufgezeigt und verschiedene Netzwerkarchitekturen

vorgestellt. Überdies werden Metriken für die Bewertung der Leistung von Objekterkennungsmodellen diskutiert.

Im darauffolgenden Kapitel 3 wird der Stand der Technik der zurzeit beliebtesten Modelle für Objekterkennung sowie für Zeitreihenvorhersagen eingegangen. Zusätzlich werden Werkzeuge für die Implementierung von neuronalen Netzen präsentiert. Abschließend werden Satellitenmissionen eingeführt, die Daten für diese Arbeit zur Verfügung stellen.

In Kapitel 4 folgt die praktische Umsetzung der Arbeit. Dazu wird zunächst auf die Beschaffung der Satellitendaten, die Aufbereitung eines Datensatzes für ein Modell für die Objekterkennung sowie dessen Training eingegangen. Darauf folgt die Untersuchung der Möglichkeiten, künstliche Zeitreihen mithilfe von *Long Short-Term Memory* (LSTM) zu erhalten. Schließlich erfolgt die energetische Betrachtung der ermittelten Datenpunkte baden-württembergischer Autobahnraststätten.

Abschließend werden in Kapitel 5 die Ergebnisse der Arbeit präsentiert, bevor Kapitel 6 die Arbeit zusammenfasst und einen Ausblick umfasst.

2. Grundlagen

In diesem Kapitel werden die für diese Arbeit relevanten Grundlagen erläutert. Hierzu wird näher auf die Objekterkennung mittels Deep Learning in Kapitel 2.1 sowie auf Konvolutionsnetze und rekurrente neuronale Netze in den Kapitel 2.2 und 2.2 eingegangen. Schließlich werden Metriken für Objekterkennungsmodelle in den Kapiteln 2.4 sowie 2.5 erläutert.

2.1. Objekterkennung mithilfe von Deep Learning

Einen umfassenden Überblick über die Techniken der Objekterkennung bietet [2]. Die Disziplin der Objekterkennung verfolgt das Ziel, gewünschte Objekte auf Bildern sowie Videos zu erkennen und zu lokalisieren. Dazu müssen die Positionen und Grenzen der Objekte auf dem entsprechenden Bild oder Video identifiziert und das Objekt einer Klasse zugewiesen werden. Deswegen stellt die Objekterkennung eine signifikant größere Herausforderung als die Bilderkennung dar, wobei lediglich ein Objekt klassifiziert werden muss.

Bekannte Modelle sind *You Only Look Once* (YOLO), *Single Shot Detector* (SSD), *RetinaNet* und *Faster R-CNN*, welche abhängig von der vorliegenden Aufgabenstellung Anwendung finden. Daneben stellen diese Ansätze Erweiterungen von bereits existierenden Bildklassifizierungsmodellen dar. Weitere Informationen über SSD sowie *Faster R-CNN* lassen sich in [3] sowie [4] finden. Nähere Beschreibungen über *RetinaNet* und YOLO sind in [5] und Kapitel 3.1 zu finden.

Die sogenannte *Small Object Detection* ist ein Teilgebiet der Objekterkennung, die sich mit der Identifizierung kleiner Objekte befasst. Im Jahr 2016 wurden die Herausforderungen der Erkennung kleiner Gesichter behandelt. Jene Arbeit stellt Techniken zur Anpassung der Netzwerkarchitektur sowie der Trainingsdaten vor. [6] Die geringe Auflösung sowie geringe Größe der Objekte erschweren diese Aufgabe, ebenso wie Abdeckung der Objekte und Schwankungen der Lichtverhältnisse.[7] Für weitere Informationen über die Erkennung kleiner Objekte sind [6] und [7] empfehlenswert.

Auf der folgenden Abbildung 2.1 ist die Aufteilung der KI abgebildet.

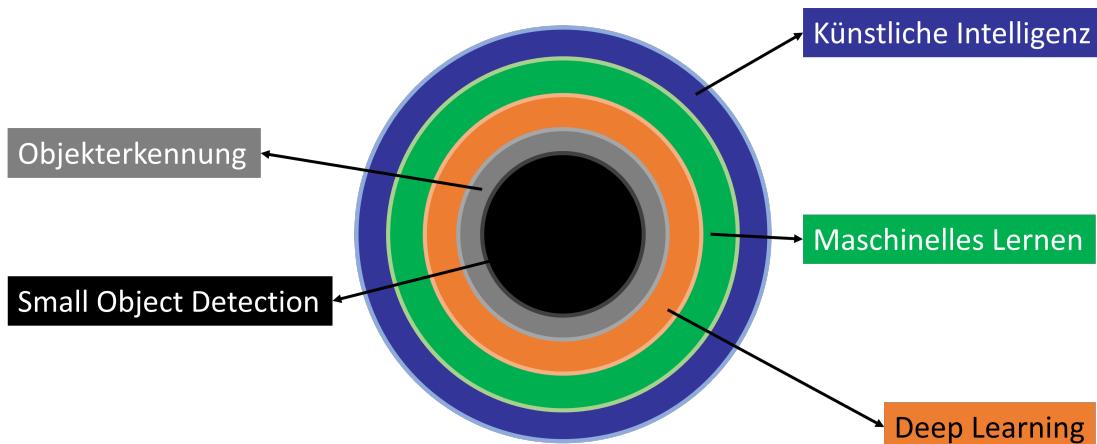


Abbildung 2.1.: Teilgebiete der KI

2.2. Konvolutionsnetze

Bei Konvolutionsnetzen (CNN) handelt es sich um eine Art neuronaler Netze, die überwiegend in der maschinellen Verarbeitung von Bild- sowie Videodateien eingesetzt werden. Dabei basieren die Schichten dieser Netze auf der mathematischen Faltungsoperation. Daneben sind so genannte *Pooling*-Schichten Voraussetzung für Konvolutionsnetze. Diese Operation reduziert die Bildgröße, wodurch sich Ressourcen des Rechners einsparen lassen. Auf Abbildung 3.1 auf Seite 10 ist eine exemplarische

Architektur eines Konvolutionsnetzes sichtbar. Hierbei sind neben Konvolutionsschichten ebenfalls *Pooling*-Schichten abgebildet. Eine umfassende Einführung in das Gebiet des *Deep Learning* stellt [8] dar und sollte bei Bedarf nach weiteren Informationen bezüglich Konvolutionsnetzen zurate gezogen werden.

2.3. Rekurrente neuronale Netze

Rekurrente neuronale Netze zeichnen sich dadurch aus, dass sie im Gegensatz zu klassischen *Feedforward*-Netzen und Konvolutionsnetzen Schichten mit Rückkopplungen besitzen. Dies ermöglicht die Erinnerung an vorangegangene Werte, weshalb diese Art von Netzen für die Verarbeitung natürlicher Sprache und sequenzieller Daten eingesetzt wird. In Kapitel 3.2 auf Seite 13 wird auf *LSTM* eingegangen, was eine Art rekurrenter neuronaler Netze darstellt. Hier lässt sich ebenfalls für weitere Informationen bezüglich rekurrenter neuronaler Netze [8] empfehlen.

2.4. Intersection over Union und Jaccard-Index

Intersection over Union ist die bildhafte Beschreibung des sogenannten *Jaccard-Indexes*. Der *Jaccard-Index* wurde von dem Mathematiker Paul Jaccard aufgestellt. Mit dem dem *Jaccard-Index* wird die Ähnlichkeit zweier oder mehrerer Mengen beschrieben. Dazu wird die Überschneidung mit der Vereinigung der Mengen ins Verhältnis gesetzt. Für zwei Mengen A und B wird der *Jaccard-Index* wie folgt berechnet:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \quad (2.1)$$

Um nur die Überschneidung zu verwenden, kann die Formel umgeformt werden:

$$J(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \quad (2.2)$$

Dabei gilt nach [9]:

$$0 \leq J \leq 1. \quad (2.3)$$

In Objekterkennungsalgorithmen wird oft der *Jaccard-Index* verwendet, um vorhergesagte *Bounding Boxen* mit den annotierten zu vergleichen. Ein Index von $J = 1$ beschreibt dabei eine perfekte Übereinstimmung, $J = 0$ gar keine. Nach dieser Funktion lassen sich nun die *Bounding Boxen* beurteilen und optimieren. Nach [10] wird dieser simple Index häufig um komplexe Kompensationsterme erweitert. Auf Abbildung A.1 auf Seite 67 ist die *IoU* grafisch dargestellt.

2.5. Mean Average Precision

Die *Average Precision (AP)* stellt eine Möglichkeit dar, die Genauigkeit und damit "Richtigkeit" eines Netzwerkes zu bestimmen. Der *AP*-Wert gibt an, wie genau eine Klasse erkannt wurde. Dazu wird zu nächst die Genauigkeit (*Precision*) einer Klasse bestimmt:

$$\text{Precisions}(k) = \frac{\text{TruePositives}(k)}{\text{TotalPredictions}}. \quad (2.4)$$

Recalls bezeichnet einen Wert, der die tatsächlich richtig erkannten Klassen misst:

$$\text{Recalls}(k) = \frac{\text{TruePositives}(k)}{\text{GroundTruth}}. \quad (2.5)$$

Damit lässt sich der *AP*-Wert folgendermaßen berechnen:

$$AP = \sum_{k=0}^{k=n-1} (Recalls(k) - Recalls(k+1) \cdot Precisions(k)). \quad (2.6)$$

Dabei stellt n die Anzahl der *Thresholds* dar (hier: *Intersection over Union*).

Außerdem lassen sich die Modelle anhand der Metrik *mean Average Precision (mAP)* bewerten. Die Berechnung des *mAP*-Wertes erfolgt folgendermaßen:

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k. \quad (2.7)$$

n ist hier die Anzahl der Klassen, während AP_k der *AP*-Wert der Klasse k darstellt.

mAP ist die Standardmetrik zur Beurteilung von Bilderkennungsalgorithmen. Je höher der *mAP*-Wert, desto höher die Qualität des Algorithmus.

3. Verwendete Technologien

In diesem Kapitel wird auf die verwendeten Technologien eingegangen. Dazu wird zunächst in Kapitel 3.1 das Modell *YOLO* für die Objekterkennung eingegangen. Schließlich wird in Kapitel 3.2 das *LSTM* erläutert. Im Anschluss werden die in dieser Arbeit eingesetzten Werkzeuge wie *TensorFlow* und *PyTorch* in Kapitel 3.3 beschrieben. Abschließend werden *Cubesat*-Satelliten in Kapitel 3.4 präsentiert, welche die Satellitenaufnahmen für diese Arbeit liefern.

3.1. YOLOv5

Im Allgemeinen unterscheiden sich die Bilderkennungsarchitekturen in ihrer Funktionsweise in der Anzahl der Schritte in ihrem Prozess. Sogenannte *Two-Stage* Detektoren verwenden zwei Schritte, *One-Stage* Detektoren einen. Bei *Two-Stage* Detektoren werden zunächst die Regionen analysiert, die Objekte enthalten können. Daraus wird ein *Regional Proposal Network* (*RPN*) erstellt, in welchem schließlich die eigentliche Objekterkennung stattfindet.

Auf der anderen Seite lernen *Single-Stage* Detektoren direkt über Regression die Wahrscheinlichkeit für Objekte an bestimmten Stellen. Dadurch erzielen *Two-Stage* Detektoren höhere Genauigkeiten, während bei *One-Stage* Detektoren der Trainingsaufwand und die Erkennungsgeschwindigkeit deutlich geringer sind.

Mit *YOLOv5* wird hier ein *Single-Stage* Detektor vorgestellt.

3.1.1. Einführung in YOLO

Im Jahre 2016 wurde die erste Version des Modells YOLO vorgestellt [11]. Die YOLO-Architektur basiert auf Konvolutionsnetzen, die auf Faltungen beruhen. Hierbei werden Klassenwahrscheinlichkeiten sowie *Bounding Boxen* vorhergesagt. Die aktuelle Version, YOLOv5, wurde von *Ultralytics* im *Framework PyTorch* (siehe Kapitel 3.3.1 auf Seite 15) implementiert [12]. Diese Version nutzt das Konvolutionsnetz *Darknet53* als *Backbone*, welche sich für die Extraktion von Details wie beispielsweise Ecken und Kanten einsetzen lassen. Informationen zu diesem Netz lassen sich unter [11] finden.

3.1.2. Netzwerkarchitektur

Auf Abbildung 3.1 ist die Architektur des Modells YOLO abgebildet.

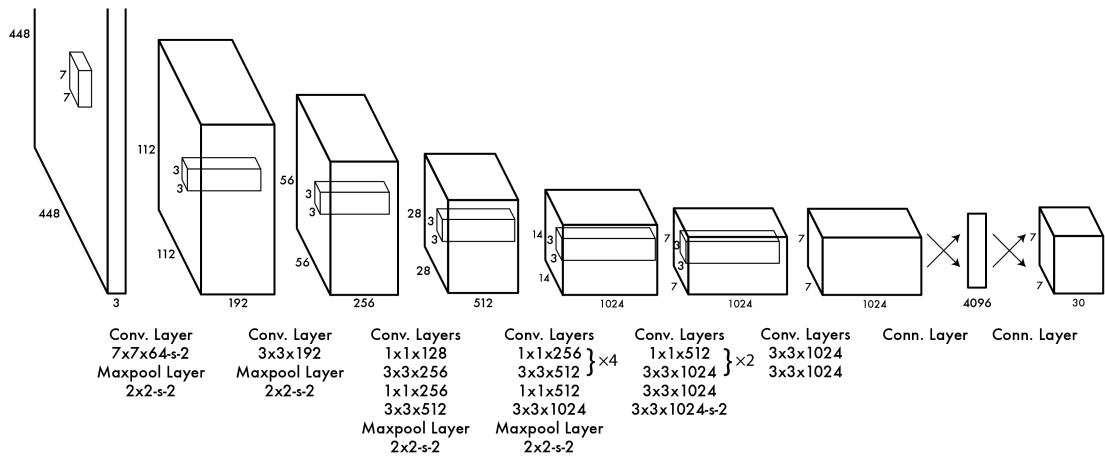


Abbildung 3.1.: Architektur von YOLO [11]

Zu Beginn wird das zu betrachtende Bild auf eine Größe von 448 x 448 Pixel skaliert, welches als Eingabe des Konvolutionsnetzes dient. Die Archi-

tektur besteht aus 24 Faltungsschichten, gefolgt von zwei vollvermaschten Schichten.

Die aktuelle Version von YOLO, YOLOv5, verfügt über vier verschiedene Größen: YOLOv5s (*Small*, am kleinsten), YOLOv5m (*Medium*), YOLOv5l (*Large*) und YOLOv5x (*XLarge*, am größten). Die Größe bezieht sich hier auf die Anzahl und Größe der Faltungsschichten. Auf der folgenden Abbildung 3.2 sind die unterschiedlichen Modellgrößen von YOLOv5 abgebildet.

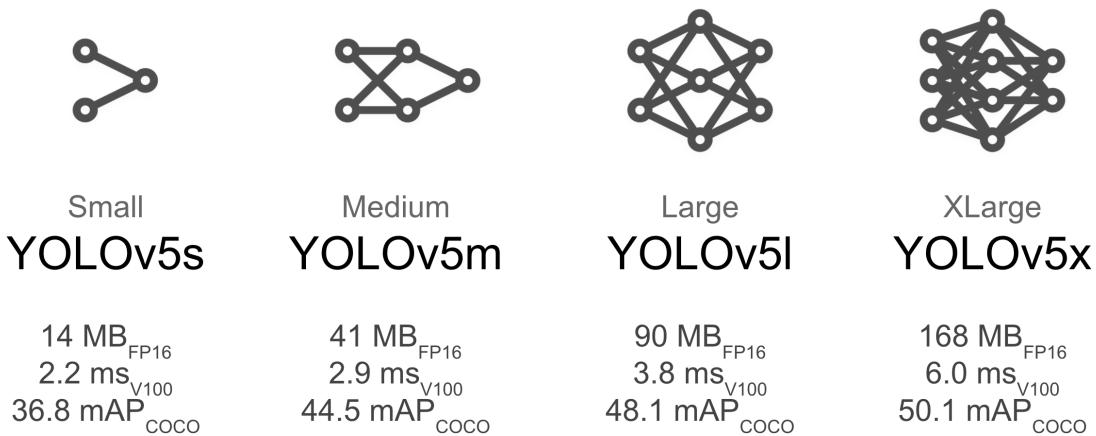


Abbildung 3.2.: Verschiedene Größen des YOLOv5-Modells [12]

Zusätzlich ist der Speicherbedarf in MB, die Klassifizierungsgeschwindigkeit in ms sowie die *mAP* jeder Modellgröße angegeben. Mit zunehmender Anzahl an Faltungsschichten steigen sämtliche Größen.

3.1.3. Bounding Box-Vorhersagen

YOLO betrachtet die Objekterkennung als Regressionsproblem, wobei erkannte Objekte durch *Bounding Boxen* eindeutig bestimmt werden. YOLO zählt dabei zu den sogenannten *One-Stage*-Detektoren, die in einem Schritt alle *Bounding Boxen* erkennen. Das Eingabebild wird in ein Raster

aus $S \times S$ Zellen aufgeteilt, wobei jede Zelle prüft, ob und wie viele Objekte sie erkennt. Jede *Bounding Box* B wird durch Koordinaten (x, y, w, h) und einen *Confidence Score* beschrieben. Dabei stellen x und y die Koordinaten des Zentrums der *Bounding Box* dar, während w und h die Breite und Höhe beschreiben. Außerdem bildet der *Confidence Score* ab, ob die *Bounding Box* ein Objekt enthält und wie stark die Deckung ist.

Es gilt:

$$\text{Confidence} = \text{Pr}(\text{Object}) \cdot \text{IoU}. \quad (3.1)$$

$\text{Pr}(\text{Object})$ beschreibt dabei die Wahrscheinlichkeit, dass sich ein Objekt in der Zelle befindet. Im Anschluss wird die *Intersection over Union* (IoU) angewandt. Dabei wird die Überlappung der von Modell bestimmten *Bounding Box* und der *Ground Truth Bounding Box* durch deren Vereinigung geteilt. Mit der Klassenanzahl C und der Anzahl an *Bounding Boxen* in einer Zelle B wird mithilfe der Bildseitenlänge S der Ergebnistensor berechnet:

$$S \times S \times (B * 5 + C). \quad (3.2)$$

3.1.4. Trainingsprozess

Für das Training des YOLO-Modell wird eine große Anzahl ab Trainingsdaten benötigt, die manuell mit *Ground Truth* versehen werden müssen. *Ground Truth* beschreibt die tatsächliche Position von Objekten und deren *Bounding Box* in einem Eingabebild. YOLO verwendet für das Training als Fehlerfunktion die Summe der Fehlerquadrat nach

$$SSE = \sum_{i=0}^{i=n} (x_i - \bar{x})^2. \quad (3.3)$$

Für die Aktivierung in der letzten Schicht wird die *Leaky Rectified Linear* Funktion mit

$$\phi(x) = \begin{cases} x & , \text{ wenn } x > 0 \\ 0.1x & , \text{ sonst} \end{cases} \quad (3.4)$$

verwendet.

3.2. Long Short-Term Memory

Im Jahr 1997 wurde *LSTM* erstmals in [13] präsentiert. Dabei handelt es sich um ein rekurrentes neuronales Netz mit Gedächtnis. Diese Art von Daten eignet sich für die Verarbeitung von sequenziellen Daten wie beispielsweise Zeitreihen, natürliche Sprache sowie handgeschriebene Texte. *LSTMs* sind in der Lage, Abhängigkeiten über lange Zeiträume zu erkennen. Der Vorteil zu konventionellen rekurrenten Netzen ist das Verhindern von verschwindenden Gewichten im Netz. Bei sehr großen Datensätzen, die eine hohe Abhängigkeit ihrer Daten aufweisen, wie zum Beispiel bei langen Zeitreihen, von Ereignissen, die miteinander korrelieren, verschwinden oft Gewichte in großen Modellen (*Vanishing Gradient Problem*). Dadurch werden ganze Schichten nutzlos und die Komplexität des Modells reicht nicht mehr aus, um die Zeitreihe adäquat zu beschreiben. *LSTM* versucht dieses Problem zu lösen. [13]–[15]

3.2.1. Aufbau einer LSTM-Zelle

In einem *LSTM* ersetzt eine *LSTM-Zelle* ein herkömmliches Neuron. Auf Abbildung 3.3 ist eine *LSTM-Zelle* abgebildet. Zu sehen sind das *Input-*, *Forget-* sowie das *Output-Gate*, die die selektive Erinnerung an Erfahrungen ermöglichen. Dies lässt die Realisierung eines langanhaltenden Kurzzeitgedächtnisses zu.

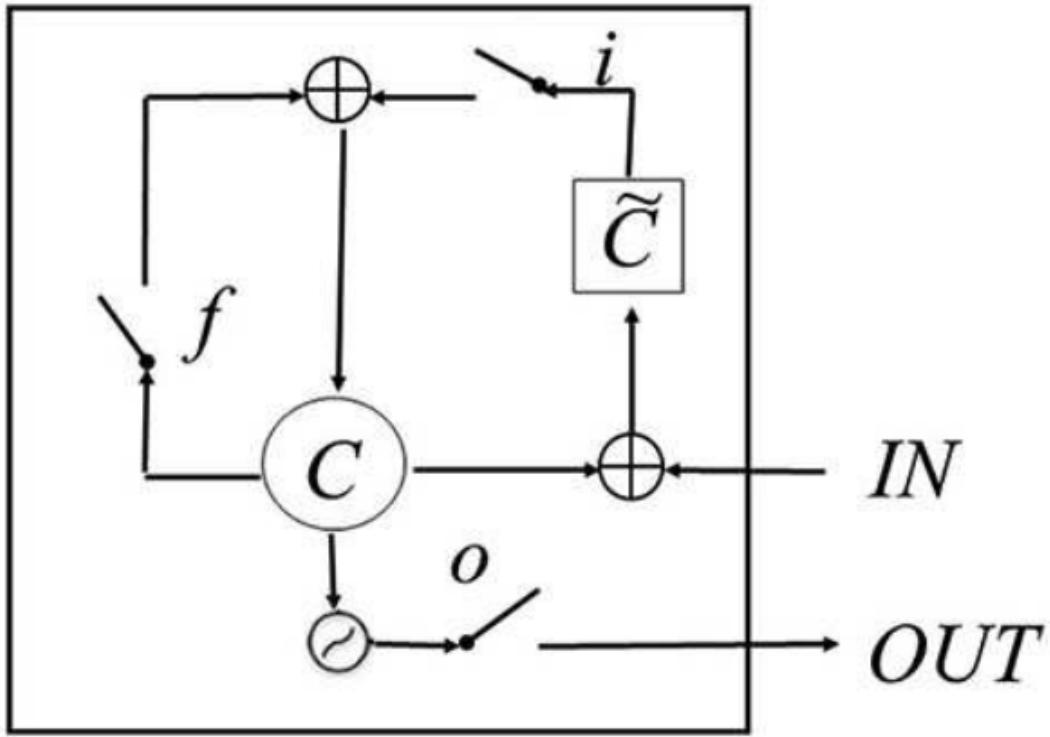


Abbildung 3.3.: Aufbau einer *LSTM* Zelle mit einem langanhaltenden (f), einem langanhaltenden (i) und einem *Output-Gate* (o) [16]

Dabei lernt das *LSTM* den Mechanismus zum Öffnen und Schließen der Tore automatisch und selbstständig in Abhängigkeit der Ein- und Ausgabedaten. Mit all diesen Maßnahmen erhöht sich auch die Komplexität des Trainings im Vergleich zu konventionellen Netzen stark.

3.2.2. LSTMs für Zeitreihen

Mit den beschriebenen Charakteristiken eignet sich *LSTM* perfekt für das Vorhersagen von simplen Zeitreihen über einen langen Zeitraum. *LSTMs* können zu bidirektionalen *LSTMs* erweitert werden. Hierbei wird das Netzwerk sowohl vorwärts als auch rückwärts durchlaufen. Das Netz erhöht damit seine Fähigkeit Zeitabhängigkeiten zu erkennen. [17], [18]

3.3. Python-Frameworks für Deep Learning

PyTorch sowie *TensorFlow* sind Python-Bibliotheken aus dem Anwendungsgebiet *Deep Learning*, die *Central Processing Units (CPUs)* sowie *Graphics Processing Units (GPUs)* nutzen. Für weiterreichende Informationen über die *Frameworks* lassen sich [19] sowie [20] empfehlen. In diesem Abschnitt werden die in der Arbeit verwendeten *Frameworks* kompakt vorgestellt.

3.3.1. PyTorch

PyTorch wurde 2016 von Facebook als Alternative zu *TensorFlow* auf Basis der *Torch-Library*, die 2002 veröffentlicht wurde, entwickelt. Ziel war es ein *Framework* zu erstellen, das sowohl schnell als auch simpel bedienbar ist. Das *Framework PyTorch* stellt Funktionen für die *GPU* beschleunigte Berechnung von Tensoren zur Verfügung und wurde speziell für Forschende und wissenschaftliches Arbeiten entwickelt. [21]

3.3.2. TensorFlow und Keras

TensorFlow wurde 2015 von dem Unternehmen *Google* entwickelt. Seither entwickelte sich *TensorFlow* zum beliebtesten *Framework* für maschinelles Lernen und *Deep Learning*. Im Gegensatz zu vorherigen Bibliotheken verwendet *TensorFlow* einen graphbasierten Datenfluss Ansatz, der aus zwei Schritten besteht. Zunächst wird ein Graph aus Anweisungen aufgebaut, der in der zweiten Phase abgearbeitet wird. Jeder Knoten des Graphen verarbeitet dabei Tensoren als Eingabe und gibt auch ebenso einen Tensor aus. Die Knoten greifen dabei auf eine *Application Programming Interface (API)* zurück, deren Funktion in performanteren Programmiersprachen implementiert sind, meist in C oder C++. Für den Entwickelnden stellt *TensorFlow* zahlreiche vorgefertigte Funktionen wie *Loss-Funktionen* und *Gradienten-Berechnung* zu Verfügung. [22], [23]

Keras ist eine *Deep Learning*-Bibliothek und *API* die direkt auf *TensorFlow* aufsetzt. Die Bibliothek ermöglicht den unkomplizierten Aufbau von *Deep Learning*-Modellen in geringer Zeit. Somit wird *TensorFlow* durch *Keras* mit implementierten Modellen und anpassbaren Schichten erweitert. Ein weiterer Vorteil ist die Modularität von *Keras*, die sich einerseits für den Aufbau von traditionellen Netzen einsetzbar ist und auf der anderen Seite *Computer Vision* für die Bilderkennung implementiert. [24]

3.4. CubeSat Satelliten

Bei einem *CubeSat* handelt es sich um einen würfelförmigen Satelliten mit einer Seitenlänge von mindestens 10 cm, die sich für den Aufbau von größeren Satelliten einsetzen lassen. [25]

In den folgenden Unterkapiteln wird auf die Satelliten-Missionen eingegangen, deren Aufnahmen in dieser Arbeit Verwendung finden.

3.4.1. PlanetScope

Die *PlanetScope*-Satellitenkonstellation wird von der Firma *Planet Labs* betrieben und besteht aus circa 130 *CubeSat*-Satelliten. Dabei sind sie in der Lage die gesamte Landoberfläche der Erde täglich mit einer Auflösung von $3 \frac{\text{m}}{\text{pixel}}$ abzubilden. Außerdem befinden sich die Satelliten in einer Höhe von 475 - 525 km. Auf der folgenden Abbildung 3.4 ist ein Satellit der *PlanetScope*-Mission abgebildet.

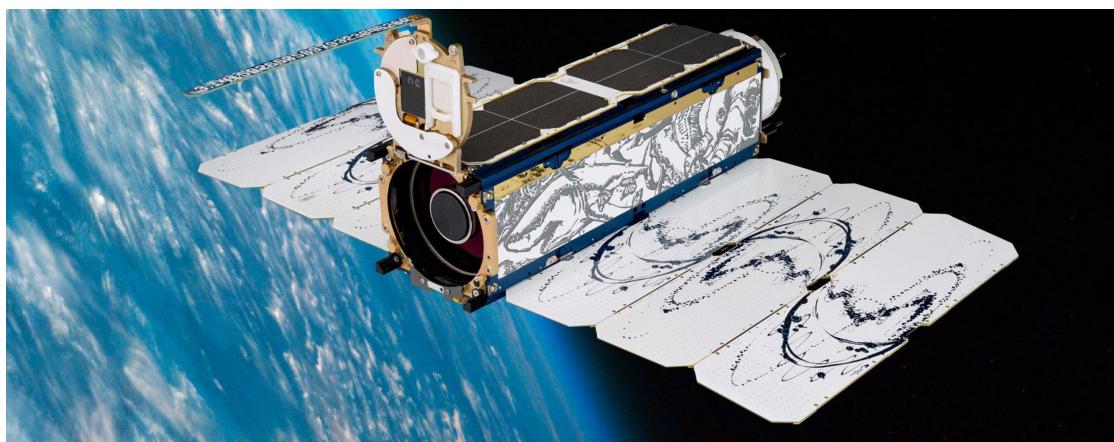


Abbildung 3.4.: *PlanetScope*-Satellit [26]

Überdies lassen sich unter [20] und [27] weiterreichende Informationen zu *PlanetScope* finden. Auf Abbildung 4.1 auf Seite 22 ist ein beispielhaftes Satellitenbild der *PlanetScope*-Mission zu sehen.

3.4.2. RapidEye

RapidEye war eine Mission von *Planet*, die vom 29. August 2008 bis zum 31. März 2020 lief und aus fünf identischen Satelliten bestand. Diese Satellitenkonstellation eignet sich für den Zugriff auf historische Aufnahmen aus diesem Zeitraum. Auf der folgenden Abbildung 3.5 sind *RapidEye*-Satelliten abgebildet.

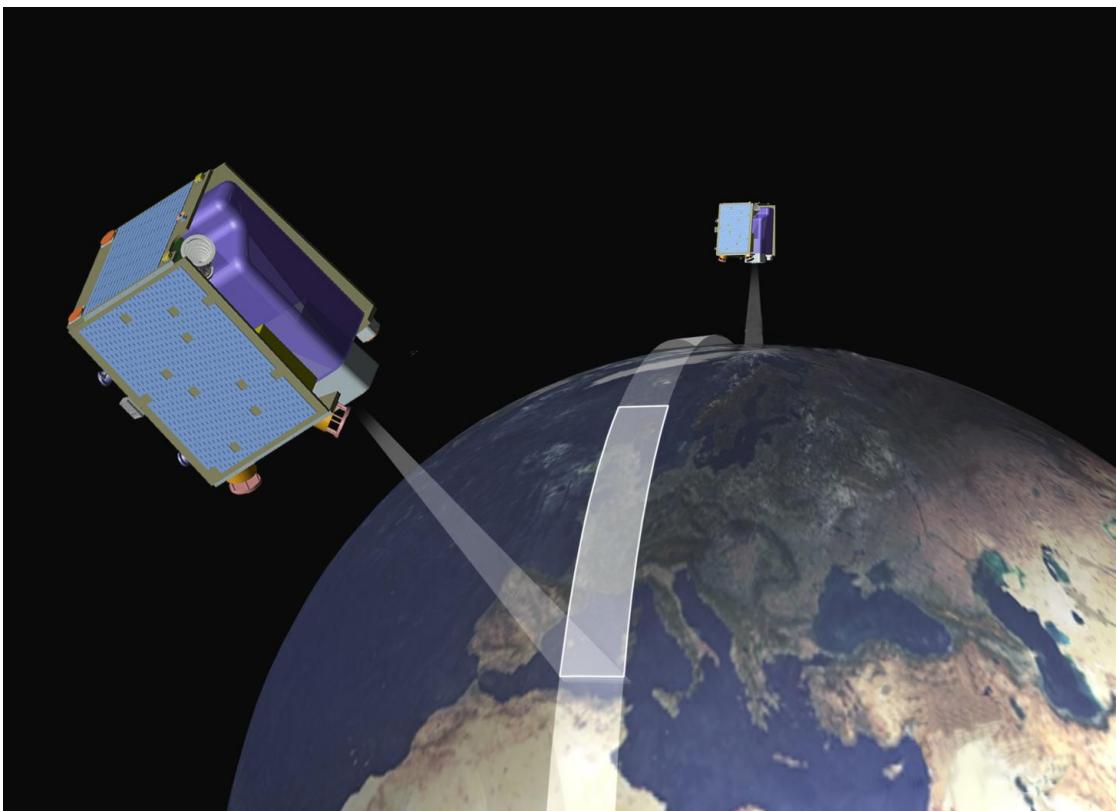


Abbildung 3.5.: *RapidEye* Satelliten [28]

Die Satelliten befanden sich in einer Höhe von 630 km und lieferte zwischen täglichen und allen fünf Tagen Satellitenaufnahmen der Erde. Dabei beträgt die Auflösung der Satellitenbilder $5 \frac{\text{m}}{\text{pixel}}$. Weitere Informationen über *RapidEye* sind unter [29] sowie [27] auffindbar.

3.4.3. SkySat

Die SkySat-Mission von *Planet Labs* startete am 21. November 2013 und dauert bis heute an. Dabei umfasst die Mission 21 Satelliten, die sich in einer Bahnhöhe zwischen 475 - 575 km befinden.



Abbildung 3.6.: *SkySat* Satelliten [30]

Abbildung 3.6 bilden Satelliten der *SkySat*-Mission ab. Jene Satelliten liefern täglich Satellitenaufnahmen der Erde mit einer Auflösung von bis zu $50 \frac{\text{cm}}{\text{pixel}}$, was sich für die Erkennung von Fahrzeugen eignet. Zusätzliche Daten zu *SkySat* finden sich unter [27] und [31]. Abbildung 4.2 auf Seite 23 zeigt ein exemplarisches Satellitenbild der *SkySat*-Satelliten.

3.4.4. Earthnet Programm

Das *Earthnet* Programm der *European Space Agency (ESA)* ermöglicht den kostenfreien Zugang zu Satellitendaten der *PlanetScope*-, *SkySat*- sowie *RapidEye*-Mission. Das Programm wurde im Jahre 1976 in das Leben gerufen und verfolgt das Ziel, Zugang zu Satellitendaten der Erde zu ermöglichen. Heutzutage bietet *Earthnet* Zugang zu Daten von Drittmissionen, wie bereits erwähnt, *PlanetScope*-, *SkySat*- sowie *RapidEye*. Für zusätzliche Informationen über das *Earthnet* Programm der *ESA* lässt sich [29] empfehlen.

Für Forschungszwecke werden diese Daten nach einer erfolgreichen Bewerbung kostenlos zur Verfügung gestellt.

3.4.5. Google Earth

Neben den Satelliten des Unternehmens *Planet Labs* dient *Google Earth Pro* als zusätzliche Informationsquelle. Die Satellitendaten von *Google Earth Pro* stammen aus unterschiedlichen Quellen. Hierzu gehören beispielsweise die *Landsat*-Satelliten der *National Aeronautics and Space Administration (NASA)*, *Sentinel*-Satelliten der *ESA* sowie Satelliten des Unternehmens *DigitalGlobe* [32]. Ein beispielhaftes Satellitenbild von *Google Earth Pro* ist auf Abbildung 4.3 auf Seite 24 abgebildet.

Daneben existieren weitere Unternehmen wie *Maxar*, die hochauflösende Satellitendaten der *WorldView*-Mission oder des ehemaligen Unternehmens *GeoEye* kommerziell anbieten.

4. Praktische Umsetzung

Dieses Kapitel umfasst die praktische Umsetzung der Arbeit. Dazu wird zunächst die Verfügbarkeit von Satellitendaten sowie weiterer Parkplatzbeliegsungsdaten von Raststätten in Baden-Württemberg (BW) in den Kapiteln 4.1 und 4.2 analysiert. In den Kapiteln 4.3 bis 4.5 wird die Realisierung des Objekterkennungsmodells *YOLO* thematisiert. Schließlich wird in Kapitel 4.6 auf die Erstellung künstlicher Daten mittels *LSTM* erläutert. Im Anschluss daran wird der Ladebedarf einer durchschnittlichen Raststätte in BW in Kapitel 5.2 bestimmt.

4.1. Beschaffung der Satellitenbilder

Neben bei dem Unternehmen *Planet Labs* käuflich erworbenen Daten, wurden die kostenfreien Aufnahmen des *Earthnet* Programms genutzt. Jene Daten standen nach erfolgreicher Bewerbung für das *Earthnet* Programm zur Verfügung. Darüber hinaus stellt *Planet Labs* für Studierende zu akademischen Zwecken ein kostenloses monatliches Kontingent von *PlanetScope*-Aufnahmen bereit.

Auf der folgenden Abbildung 4.1 ist die Satellitenaufnahme der Raststätte *Kraichgau* der *PlanetScope*-Satelliten abgebildet.



Abbildung 4.1.: Beispiel einer Satellitenaufnahme der Autobahnreststätte Kraichgau am 24.07.2022 (*PlanetScope*-Satelliten)

Im Vergleich dazu ist auf der Abbildung 4.2 die Aufnahme der *SkySat*-Satelliten zu sehen.

Dabei sind die unterschiedlichen Auflösungen der beiden Satellitenaufnahmen ersichtlich. Die Aufnahmen der *SkySat*-Mission eignen sich aufgrund der höheren Auflösung im Vergleich zu den *PlanetScope*-Aufnahmen für die Erkennung kleiner Objekte. Jedoch unterscheiden sich die Missionen ebenso in ihrer Frequenz der Aufnahmen, da *SkySat*-Satelliten eine niedrigere Frequenz von Aufnahmen der gleichen Gebiete besitzt. Im Gegensatz dazu, sind *PlanetScope*-Satelliten in der Lage, häufiger Aufnahmen der gleichen Gebiete bereitzustellen. Aus diesen Eigenschaften ergeben sich unterschiedliche Anwendungsbereiche der Satellitenmissionen. *PlanetScope*-Satelliten eignen sich aufgrund der Häufigkeit ihrer Aufnahmen sowie der geringen Bildauflösung für die Überwachung großer Gebiete.

Daher eignen sie sich exemplarisch für die Beobachtung landwirtschaftlich genutzter Gebiete oder die Überwachung der Infrastrukturentwicklung. Im Kontrast dazu lassen sich *SkySat*-Satelliten für detaillierte und präzise Aufnahmen einsetzen. Beispielhafte Anwendungen sind die Erkennung von LKW sowie weitere Fahrzeuge wie Schiffe oder Flugzeuge.

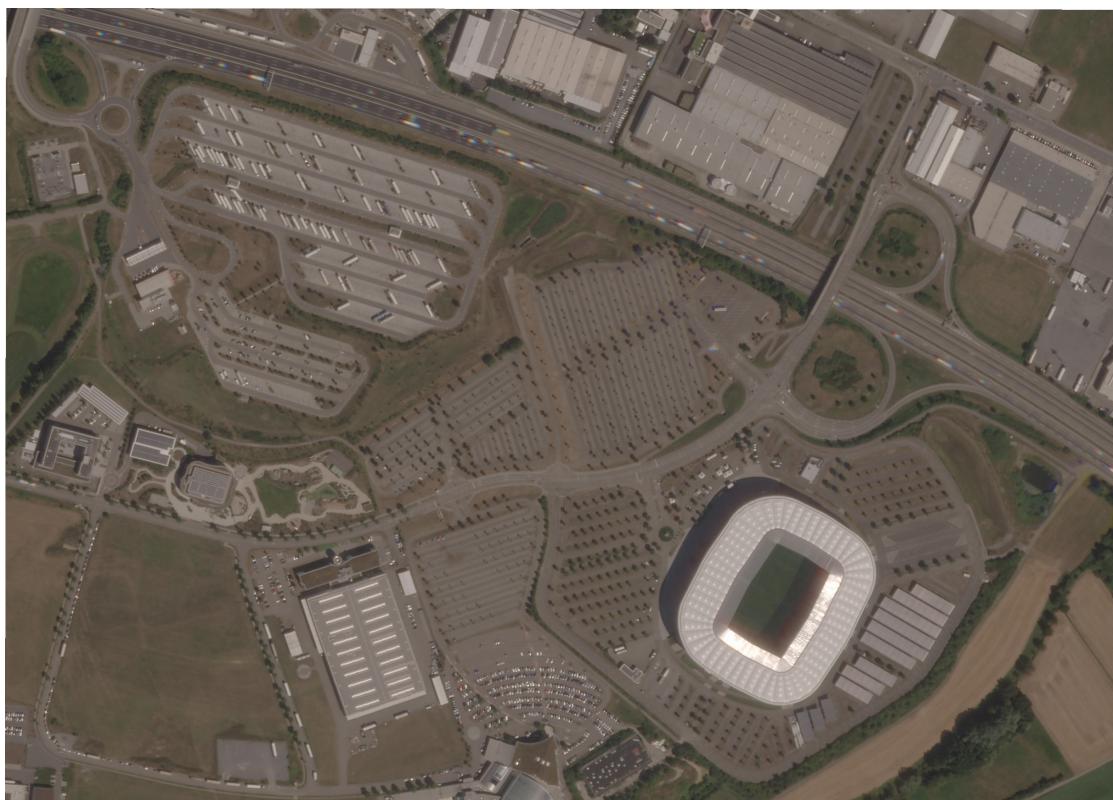


Abbildung 4.2.: Beispiel einer Satellitenaufnahme der Autobahnreststätte Kraichgau am 24.07.2022 (*SkySat*-Satelliten)

Allerdings stehen keine Archivdaten der *RapidEye*-Mission für dieses Gebiet in diesem Zeitraum zur Verfügung, weshalb keine Aufnahmen dieser Mission in der Arbeit keine Verwendung findet.

Nun folgt eine Aufnahme jener Raststätte von *Google Earth Pro* (sichtbar auf Abbildung 4.3).



Abbildung 4.3.: Beispiel einer Satellitenaufnahme der Autobahnreststätte Kraichgau am 05.07.2016, bereitgestellt von *Google Earth Pro*

Dabei wird deutlich, dass ausschließlich die Bilder von *Google Earth Pro* sowie der *SkySat*-Mission aufgrund der hohen Auflösung einsetzbar sind. Lediglich die Satelliten der *SkySat*-Mission weisen eine zureichende Auflösung für die Erkennung kleiner Objekte auf, weshalb jene Daten in dieser Arbeit verwendet wurden.

Dabei bestehen allerdings einige Herausforderungen, da lediglich eine geringe Anzahl an hochauflösten Satellitendaten der *SkySat*-Satelliten zur Verfügung stehen, welche zudem zu ähnlichen Uhrzeiten aufgenommen wurden. Überdies existieren keine Aufnahmen bei Nacht, was die Erstellung von Zeitreihen mithilfe von Satellitenaufnahmen erschwert. Eine Möglichkeit, um an Aufnahmen zu unterschiedlichen Tageszeiten zu gelangen, stellt der Zukauf von Satellitendaten von weiteren Unternehmen aus dem Bereich der Erdbeobachtung dar. *Maxar* stellt ebenfalls hochauflöste Daten zur Verfügung, allerdings sind diese Aufnahmen kostspielig und bietet nur eine marginale Verbesserung des Bestands der Daten. Aus

diesen Gründen wurde diese Möglichkeit nicht weiterverfolgt.

Im Kontext dieser Arbeit sind Aufnahmen mit der Auflösung der *SkySat*-Satelliten in Kombination mit der Aufnahmefrequenz der *PlanetScope*-Satelliten wünschenswert, um Zeitreihen zu erstellen.

4.2. Parkplatzbelegungsdaten aus Baden-Württemberg

Neben Satellitenaufnahmen von Baden-Württemberg stellt die Straßenverkehrszentrale Baden-Württemberg (SVZ BW) eine weitere Quelle für Parkplatzbelegungsdaten aus BW dar. Entlang der A5 in Baden-Württemberg in Fahrtrichtung Schweiz wurden zum Zeitpunkt dieser Arbeit bereits fünf Raststätten mit einem System zur Detektion von LKW ausgestattet. Zu den Raststätten gehören: Bad Bellingen (37 + 40 Stellplätze), Fischergrund (6), Streitkopf (6), Blauenblick-West (6) und Neuenburg-West (20) (sichtbar auf Abbildung 4.4 auf Seite 26). [33] Als nächstes sollen die Raststätten in Baden-Baden sowie Bühl mit diesem System ausgestattet werden. [33]

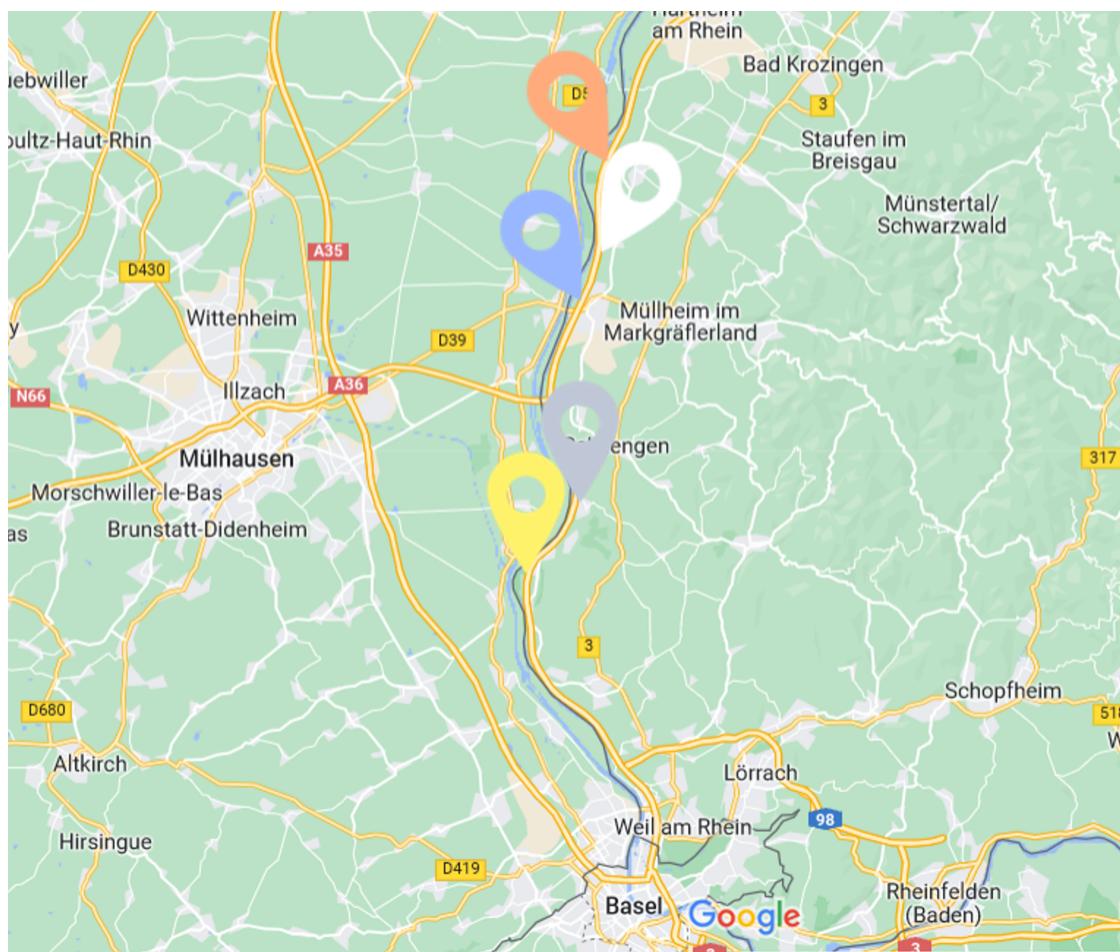


Abbildung 4.4.: Lage der Raststätten entlang der A5. Ausgewertet wurden Bad Bellingen (grau), Fischergrund (gelb), Blauenblick-West (blau), Streitkopf (Orange) und Neuenburg-West.
Erstellt mit: *Google My Maps*

Aufgrund seltener Aktualisierungen der Daten seitens der SVZ BW ist diese Quelle ebenfalls nicht für die Erstellung von Zeitreihen einsetzbar. Die Daten lassen sich unter [34] abrufen. Aus vorangegangenen Arbeiten existiert ein System, um Belegungsdaten aus Bayern automatisiert abzurufen und zu sichern. Daher ist eine erhöhte Frequenz der Aktualisierung der Daten seitens der SVZ BW wünschenswert, damit jenes System einsetzbar ist. Dies stellt eine Möglichkeit dar, authentische Daten aus Baden-Württemberg zu erhalten.

4.3. Aufbereitung des Datensatzes für die Objekterkennung

Nun wird die Aufbereitung des Datensatzes für den Objekterkennungsalgorithmus beschrieben. Unabhängig des Datensatzes werden sämtliche Bilder in der nachfolgenden Pipeline aufbereitet. Aus diesem Grund werden im Folgenden die Schritte explizit, aber nicht auf einen Datensatz zugeschnitten erläutert.

Grundsätzlich sind bei der Generierung eines Datensatzes aus Bildern folgende Schritte zu befolgen:

1. Sichtung der Daten
2. Formatierung der Daten
3. Labeln der Daten
4. Aufteilung der gelabelten Daten in Trainings-, Validierungs- und Testdatensatz
5. Bestimmung von Hintergrundbildern und gegebenenfalls Synthese von Hintergrundbildern
6. Generierung des Datensatzes in gewünschter Form



Abbildung 4.5.: Pipeline für die Generierung eines Datensatzes

Auf Abbildung 4.5 ist graphisch die Reihenfolge der Arbeitsschritte für die Generierung eines Datensatzes abgebildet.

4.3.1. Sichtung der Daten

Zu Beginn wird der Datensatz nach dem Augentest gefiltert, wobei sämtliche Bilder manuell angesehen und auf Relevanz und Auflösung geprüft werden. Eine wechselnde Bildauflösung kann dazu führen, dass das trainierte Modell gemein verbesserte Ergebnisse liefert. Jedoch wird eine größere Anzahl an Trainingsdaten benötigt und das Modell könnte angewandt auf einen Datensatz mit gleichbleibender Auflösung merklich minderwertigere Ergebnisse liefern. Viele Bilderkennungsmodelle reagieren zudem äußerst empfindlich auf eine Änderung der Auflösung. [35]

4.3.2. Formatierung der Daten

Viele Satellitenbilder werden im *GEOTIFF* Format bereitgestellt. In diesem Format ist zusätzlich zum eigentlichen Bild noch eine Referenz zu seinem Aufnahmepunkt hinterlegt. Damit können Bilder genau ihrem realen Standort zugeordnet werden, um zum Beispiel mehrere Bilder zu einer Übersicht über ein Gebiet zusammenzufügen. Im vorliegenden Fall ist diese Information jedoch überflüssig. Deshalb werden die Bilder in diesem Schritt ins *PNG*-Format umgewandelt, das vom Modell als Eingabe genutzt werden kann. Ein weiterer Vorteil ist die Reduktion der Größe. *PNG*-Bilder sind im Allgemeinen eine merklich kleinere Größe als *GEOTIFF* Bilder auf und beanspruchen daher beim Training weniger Grafikspeicher.

4.3.3. Labeln der Daten

Ein elementarer Arbeitsschritt der Aufbereitung eines Datensatzes ist das *Labeln* der Bilder. Hier werden, meist mit Hilfe einer grafischen Oberfläche, die Objekte, die es zu erkennen gilt, auf dem Bild markiert. Die Markierungen werden im Fall von *YOLO* im sogenannten *YOLO*-Format gespeichert. Hier markiert ein Rechteck das Objekt im Kontrast zu einigen anderen gängigen Algorithmen, welche ein Polygon verwenden (siehe Abbildung 4.6). Dabei wird stets das linke obere Eck des Rechtecks gespeichert. Von

dort aus wird jeweils die Höhe und Breite gemessen, sodass sich das gesamte Rechteck sichern lässt.

Für unerfahrene Nutzer eignet sich *Makesense.ai* hervorragend, da es sich dabei um eine simple Oberfläche handelt. Überdies erlaubt sie ein schnelles Labeln und bietet die Möglichkeit, eigene Modelle zur Objekterkennung zu verwenden. Zudem lässt sich der Prozess des Labelns teilweise automatisieren. *MakeSense.ai* lässt sich lokal ausführen werden und sendet keine Daten an den Entwickler der Plattform. [36] Aus diesen Gründen wurde in dieser Arbeit *MakeSense.ai* für das Labeln der Satellitenaufnahmen verwendet.

Für die Erkennung von LKW und PKW wurden die rohen Daten mit "truck" und "car" gelabelt. Bei dem vorliegenden Projekt werden Bilder von großen Rastplätzen mit verhältnismäßig kleinen Objekten, den PKWs und LKWs, analysiert.

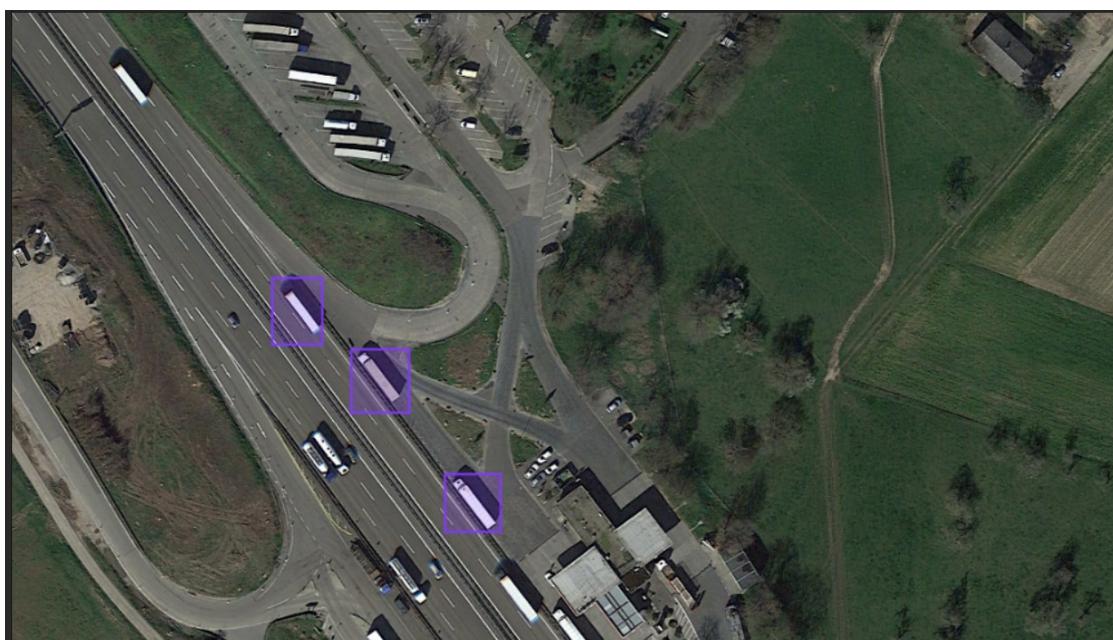


Abbildung 4.6.: Auszug aus *makesense.ai* während der Erstellung der Labels

Auf Abbildung 4.6 ist ein Ausschnitt des Tools *makesense.ai* während des Labeling-Vorgangs sichtbar. Dabei stellen die Rechtecke in der Farbe lila die Markierungen der LKW dar.

4.3.4. Aufteilung der Daten

Um die Objekte nicht mit dem Hintergrund verschwimmen zu lassen, wurden die Bilder in Quadrate einer geringeren Größe geteilt. Dazu wurden zuerst die Bilder aufgeteilt und schließlich die Koordinaten der Labels entsprechend angepasst. Im Training zeigte sich, dass durch Variation der Größe der Quadrate Trainingsunterschiede von bis zu 25% Genauigkeit auftraten. Ein weiterer Vorteil ist die Beschleunigung des Trainings, da nicht stets ein großes Bild betrachtet werden muss, sondern eine erhebliche Anzahl an kleinen Bildern. Dadurch lässt sich die Auslastung des Grafikspeichers signifikant reduzieren.

Der Datensatz wird im folgenden Schritt aufgeteilt, sodass circa 70 % der Daten für das Training verwendet werden, 20 % für den Test des Trainings und 10 % für die Validierung des Modells. Werden ausschließlich jene gelabelten Bilder für das Training eingesetzt, wird das Modell auf sämtlichen Bildern versuchen, LKW sowie PKW zu erkennen.

4.3.5. Bestimmung von Hintergrundbildern

In einer tatsächlichen Datenquelle wird nicht immer ein Objekt auf einem Eingabebild vorhanden sein. Deshalb werden nun im nächsten Schritt so genannte Hintergrundbilder eingefügt, auf denen kein Objekt vorhanden ist. Dazu sollten Bilder gewählt werden, die eine ähnliche Beschaffenheit wie die gelabelten Bilder aufweisen. In dem vorliegenden Projekt wurden dazu Bilder von leeren Parkplätzen benutzt, wie in 4.7 beispielhaft dargestellt.

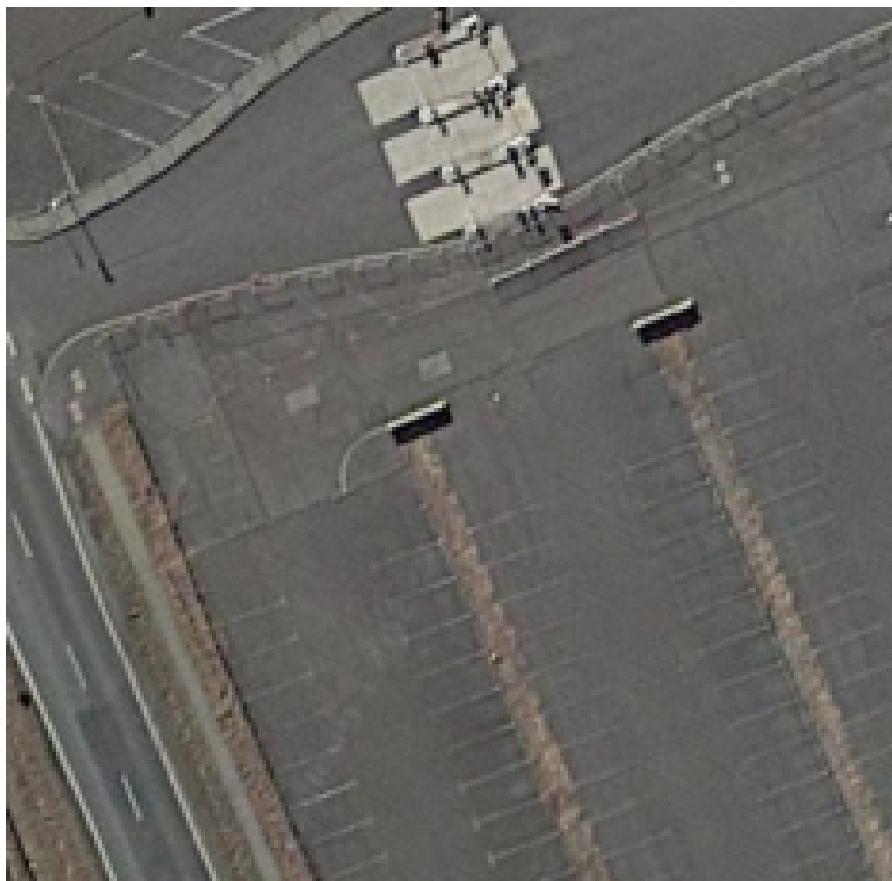


Abbildung 4.7.: Hintergrundbild, das im Training verwendet wurde

Diese Abbildungen benötigen keine Labels, da keine zu erkennenden Objekte vorhanden sind. Bei der Menge der Bilder sollte darauf geachtet werden, dass diese nicht überproportional mehr sind, als die Trainingsdaten, um die Trainingsqualität zu erhalten. Aus Erfahrungswerten ergibt sich eine ungefähre Anzahl von maximal 10 % der Trainingsdaten, wobei sich die genaue Menge dem Trainingsergebnis nach anpassen lässt. Im eingesetzten Modell ist eine Menge von 1 % genügend, da durch die Auftrennung der Bilder weitere Hintergrundbilder entstehen. Die Verwendung von Hintergrundbildern führt zu einer erheblichen Senkung der *False-Positive-Rate*.

4.3.6. Generierung des Datensatzes

Nun wird aus den fertig aufbereiteten Daten ein Datensatz generiert. Dazu wird eine YAML-Datei erstellt, die die Struktur des Datensatzes beschreibt. In ihr werden die gelabelten Klassen ihren Abkürzungen zugeordnet und die Speicherorte der Sets in Bezug auf die YAML-Datei angegeben. Die Informationen werden anschließend im Training verwendet, um den Einsatz der erstellten Sets zu automatisieren. In diesem Schritt lässt sich zudem die Struktur des Datensatzes an das zu trainierende Modell anpassen.

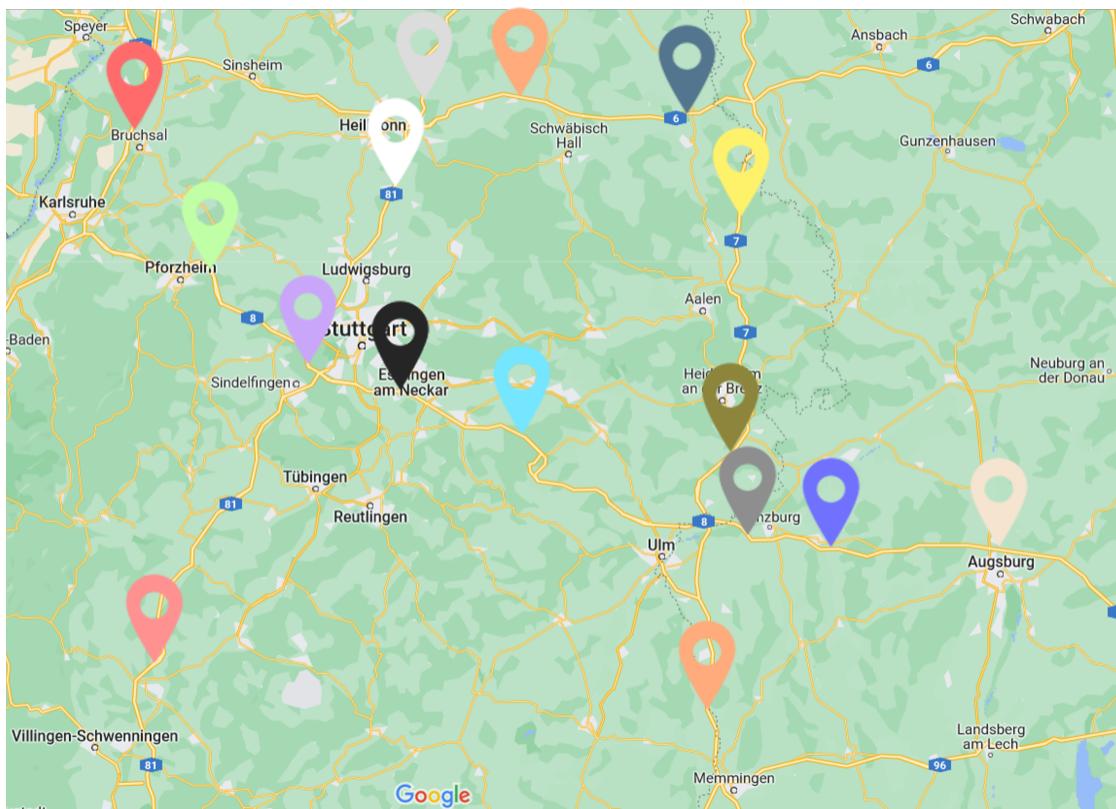


Abbildung 4.8.: Lage der Raststätten: Augsburg (beige), Bruchsal (rot), Burgauer See (blau), Denkendorf (schwarz), Ellwanger Berge (gelb), Gruibingen (hellblau), Hohenlohe (orange), Illertal (orange), Jagsttal (hellgrau), Leipheim (grau), Lontal (ocker), Neckarburg (rosa), Pforzheim (grün), Satteldorf (dunkelblau), Sindelfinger Wald (violett) und Wuppenstein (weiß); Erstellt mit: Google My Maps

Nun wird auf den konkreten Datensatz des Objekterkennungsmodells auf Satellitendaten eingegangen. Jener Datensatz besteht aus Satellitenaufnahmen von *Google Earth Pro* sowie von *SkySat*-Satelliten und umfasst 80 verschiedene Bilder von 16 Autobahnraststätten in Bayern sowie Baden-Württemberg. Die Raststätten sind in Abbildung 4.8 dargestellt.

4.4. Objekterkennung der Fahrzeuge

Für die Erkennung von Fahrzeugen wird zunächst eine adäquate Modellarchitektur bestimmt, welche im Anschluss trainiert und ausgewertet wird.

4.4.1. Wahl eines Modells für die Objekterkennung

Bei der Wahl der Architektur müssen verschiedene Kriterien beachtet werden. Allgemein werden Objekterkennungsmodelle nach ihrer Genauigkeit, ihrer Zeit für das Erkennen von Objekten, ihres Trainingsaufwandes und ihrer Größe beurteilt. In den meisten Fällen gilt es dabei einen Kompromiss zwischen Genauigkeit und Schnelligkeit zu finden. Im vorliegenden Fall ist lediglich die Genauigkeit entscheidend, da keine Echtzeitanforderungen bestehen. In dieser Arbeit besteht lediglich die Anforderung, dass die Erkennungszeit geringer sein muss als die Frequenz der Satellitenbilder, die mindestens einen Tag pro Bild beträgt. Deshalb lässt sich eine Architektur mit hoher Genauigkeit auswählen.

Die ideale Lösung für die Erkennung von kleineren Objekten, wie im vorliegenden Fall, wäre die Verwendung eines *Two-Stage* Detektors. Allerdings wären diese auf der Hardware während des Projektverlaufs zwar ausführbar gewesen, doch die Zeit und Ressourcen, die das Training in Anspruch nehmen, stehen in keinem Verhältnis zu der zu erwartenden Genauigkeitsverbesserung.

Die gängigsten Modelle unter den *Single-Stage* Detektoren sind die Modelle *SSD* und *YOLO*, die ähnliche Ergebnisse bei der Erkennung von

kleinen Objekten liefern. Daneben stellt YOLO zurzeit mit seinen iterativen Verbesserungen den de facto Standard dar. Da allerdings das Training von YOLO auf der verwendeten GPU deutlich schneller ist, wurde YOLO als Erkennungsalgorithmus gewählt. [37]

Für das Training wird die aktuelle Version 5 verwendet, die zum Zeitpunkt der Arbeit die höchste Genauigkeit aufweist. [38] YOLO lässt sich dabei in vier verschiedenen Ausführungen nutzen, die sich jeweils in der Größe ihres Modells (sichtbar auf Abbildung 3.2 auf Seite 11) unterscheiden.

Mit steigender Modellgröße erhöht sich die Qualität der Bilderkennung. Allerdings steigt mit der Größe des Modells ebenso die Zeit zum Erkennen von Objekten an. Für das Training werden die zwei größten Modelle "Large" (l) und "XLarge" (xl) verwendet und miteinander verglichen.

4.4.2. Training und Validierung des Netzes

Die gewählten Modelle werden im Folgenden, mit dem in Kapitel 4.3 erstellten Datensatz trainiert und verifiziert.

Training

Während des Trainingsprozesses werden die Bildgröße, die Epochenzahl und die Zusammensetzung des Datensatzes variiert. Dies dient dem Auffinden einer Kombination, die ein optimales Trainingsergebnis liefert.

Satellitenbilder besitzen üblicherweise eine erhebliche Größe, da die abgedeckte Fläche häufig sehr groß ist. Werden diese Bilder direkt im Netz zum Training verwendet, ist der Bedarf von Grafikspeicher immens, da jedes Bild in den Speicher der Grafikkarte eingelesen werden muss. Weiterhin besitzen bei Verwendung der Originalgröße auch die im Verhältnis kleinen Objekten kaum Relevanz auf dem Bild. Dies erschwert die Erkennung von kleinen Objekten.

Deshalb werden die Bilder vor der Eingabe in das Netz durch Auftrennung auf eine Größe zugeschnitten. Aus einem Bild der Auflösung 1080x1080

Pixel werden somit zum Beispiel 16 Bilder der Auflösung 320x320 Pixel generiert. Dabei werden die Randstücke möglichst ähnlich groß zu der anvisierten Größe gehalten. Damit ergibt sich eine homogene Trainingsmenge sowie eine geringere Überlappungen, die zu mehrfach Erkennungen führen können. Als Auflösungen werden dabei Vielfache von 320 gewählt, da YOLO nativ mit einer Auflösung von 320x320 Pixeln arbeitet.

Andere Eingabeauflösungen werden auf 320x320 Pixel vom *Input Layer* des Netzes skaliert. Nach dem Training lassen sich die Bilder zur verbesserten Visualisierung wieder zusammensetzen.

Allgemein werden die Daten vor der Eingabe in das Netz von *Albumentation* augmentiert. *Albumentation* ist eine Bibliothek, die von *Yolov5* wie ein *Plug-in* benutzt wird. Mit der Verwendung von *Albumentation* wird der Datensatz augmentiert (4.9) und erweitert, indem künstlich neue Bilder und Labels aus dem bereits vorhandenen Datensatz generiert werden. Dies geschieht durch Rotation, Veränderung der Sättigung, des Kontrastes und Neukomposition der Bilder. Dadurch wird die Datenmenge signifikant erhöht und ein umfangreicheres Training möglich. Des Weiteren verhindert eine große Anzahl an Trainingsbeispielen das Auswendiglernen der Daten. Auf Abbildung 4.9 ist ein beispielhafter Stapel von augmentierten Bildern zu sehen. Dabei sind exemplarisch die Neukomposition sowie Rotation einiger Trainingsbilder sichtbar.

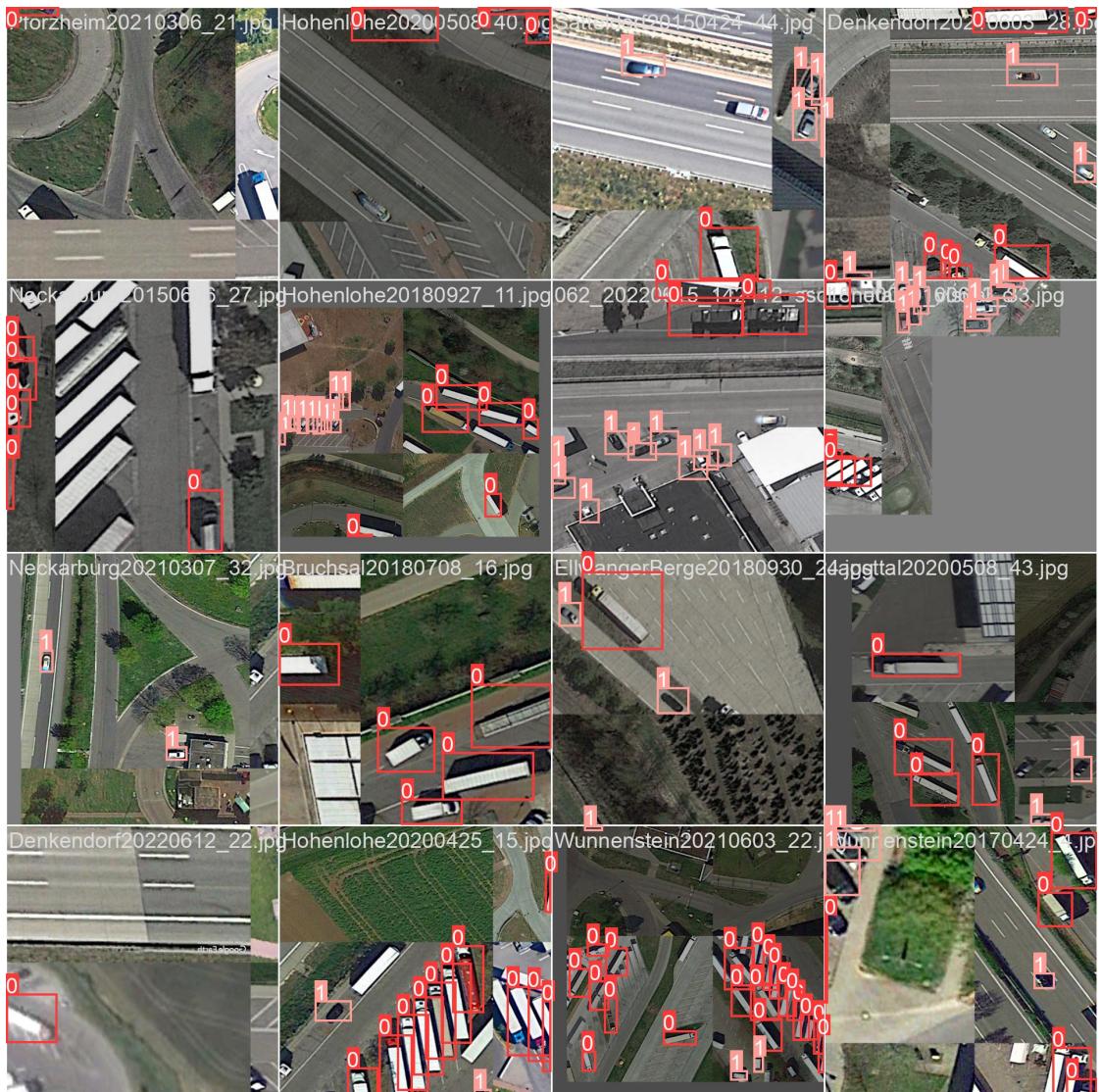


Abbildung 4.9.: Beispiel einer Trainingsbatch, die aus augmentierten Bildern besteht. LKW werden mit 0 annotiert, PKW mit 1

Zudem sind die Ergebnisse der Trainingsversuche im Anhang (Kapitel 6) auf Seite 67 zu finden.

Das präziseste Ergebnis liefert ein "l"-Modell mit der Bildgröße 960x960 Pixel. Hier beträgt die *mAP* 89,5 % und damit der Fehler 10,5 %. In der Literatur wurde auf ein ähnliches Problem eine *Precision* von 85,1 % mit YOLOv3 erzielt. [39] In Abbildung 4.10 sind die Trainingsmetriken des besten Modells für jede Trainingsiteration abgebildet.

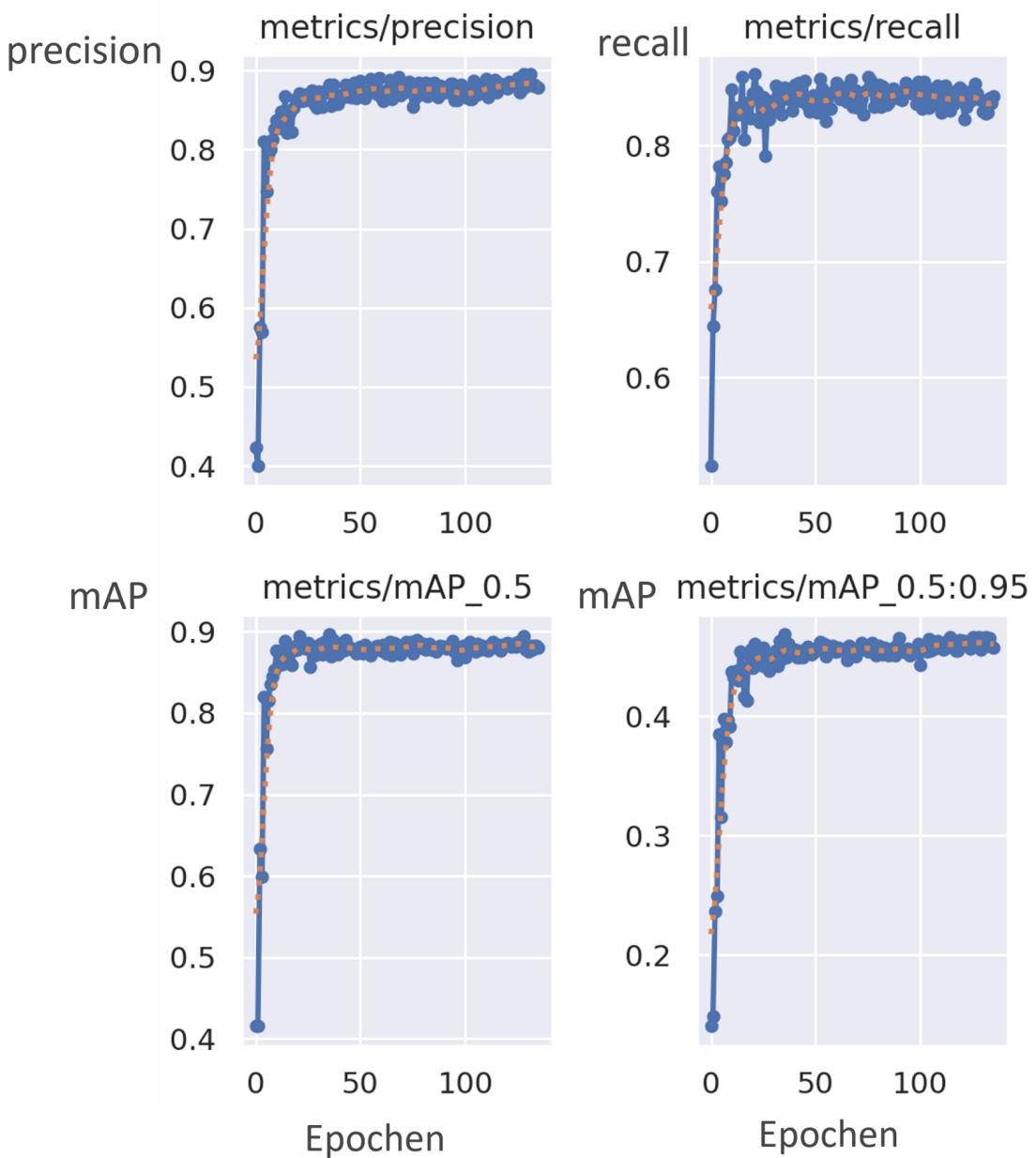


Abbildung 4.10.: Trainingsmetriken des besten Modells aufgetragen über die Trainingsepochen, links oben: *Precision*, rechts oben: *Recall*, links unten: *mAP* mit *Threshold* = 0,5, rechts unten: *mAP* mit *Threshold* = 0,95

Bei genauerer Betrachtung der Trainingsresultate in 4.10 in der Konfusionsmatrix in 4.11 fällt auf, dass die Erkennung von LKW besser gelingt als

die Erkennung von PKW.

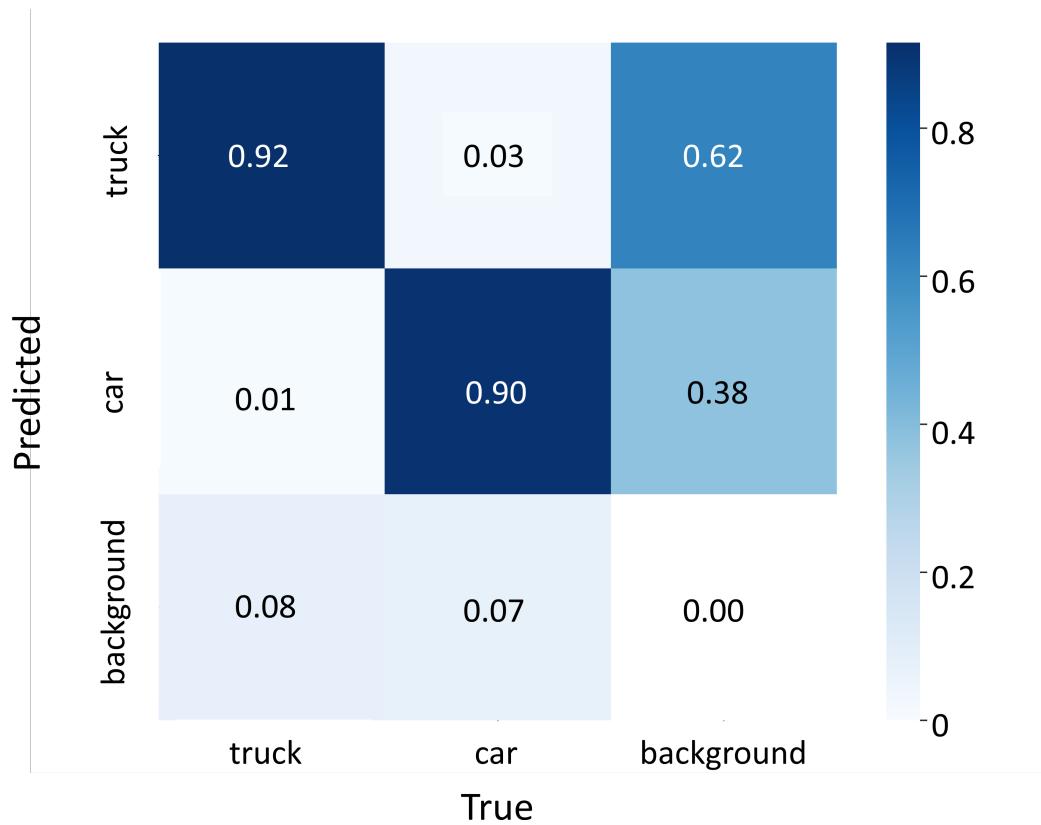


Abbildung 4.11.: Konfusionsmatrix der Trainingsdaten.
LKW werden zu 92 % korrekt erkannt

Während des Trainings wurde der Datensatz variiert. So wurde zum Beispiel ausschließlich mit *SkySat* Bildern oder alleinig mit *Google Earth* Bildern trainiert. Das Modell mit dem besten Trainingsergebnis wurde mit einer Kombination von *Google Earth* Bildern und *SkySat* Bildern trainiert. Die Batchsize wurde dabei immer so groß wie möglich gewählt, um möglichst viele verschiedene Bilder in jeden Trainingsdurchlauf einfließen zu lassen. Das Modell wurde auf unterschiedlichen Grafikkarten (Nvidia P5000, A4000, RTX4000, RTX5000 und P6000) in einer Cloud trainiert, der Grafikspeicher der verwendeten Grafikkarten bestimmt somit direkt die

Batchsize. Allerdings lässt sich feststellen, dass die Erhöhung der Batchsize ab einer Größe von 8 nur noch einen vernachlässigbaren Einfluss auf das Ergebnis erzielt, da mit verhältnismäßig vielen Epochen trainiert wird. So konnte der Einfluss von unterschiedlicher Hardware auf das Modell reduziert werden.

Validierung

Zur Validierung werden 18 Bilder herangezogen, die nicht während des Trainings des Modells eingesetzt wurden. Auf diesen werden alle Objekte von Hand gezählt und im Anschluss mit den Ergebnissen des Modells verglichen. So lässt sich die Genauigkeit des Modells zeitnah erkennen und ebenso ein Korrekturfaktor berechnen (siehe Kapitel 4.5 auf Seite 42). Auf der folgenden Abbildung ist ein Auszug aus der automatischen Validierung des trainierten YOLOv5-Modells.

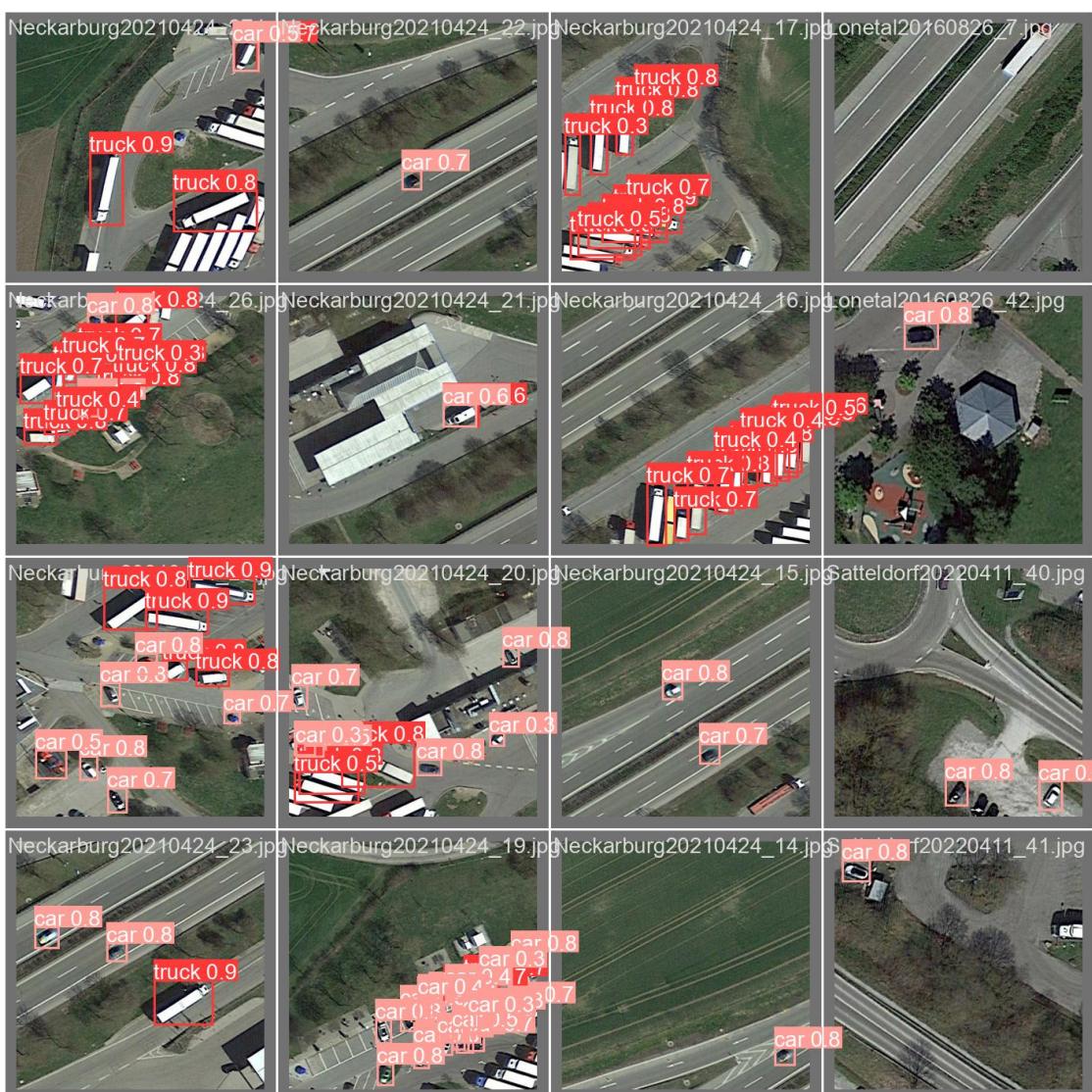


Abbildung 4.12.: Auszug aus der automatischen Validierung des YOLOv5-Netzes

Darauf sind neben den erkannten LKW ebenso die identifizierten PKW zu sehen, was durch die *Bounding Boxen* ersichtlich wird.

4.4.3. Test des Modells

Der Test des Modells dient zur Überprüfung der Generalisierungsfähigkeiten des trainierten Modells. Hierfür lässt sich der Testdatensatz einsetzen,

der wiederum Bilder enthält, die nicht während des Trainings zum Einsatz kamen. Ein beispielhaftes Testergebnis ist auf der folgenden Abbildung 4.13 abgebildet. Dabei handelt es sich um eine Satellitenaufnahme der Raststätte *Fläming*.



Abbildung 4.13.: Testergebnis der Raststätte *Fläming* mit YOLOv5

Zusätzlich ist hierauf zu erkennen, dass LKW sowie PKW durchaus mit einer hohen Genauigkeit zuverlässig erkannt werden. Somit lässt sich bestätigen, dass das trainierte YOLOv5-Modell für den allgemeinen Gebrauch einsetzbar ist und verlässliche Ergebnisse liefert.

Abschließend wird das Modell stichprobenartig visuell geprüft. Aus dieser Betrachtung lässt sich erkennen, an welchen Stellen das Modell nicht optimal arbeitet. Visuell lässt sich häufig bestmöglich erkennen, welche Parameter am Modell geändert werden sollten. Gegebenenfalls lässt sich anschließend das Modell mit geänderten Parametern erneut trainieren.

4.5. Berechnung eines Korrekturfaktors

Abbildung 4.13 zeigt zusätzlich, dass nicht sämtliche LKW sowie PKW erkannt und dass teilweise Objekte fälschlicherweise identifiziert werden. Um die Anzahl der erkannten Objekte möglichst nah am echten Wert zu halten, wird ein Korrekturfaktor ermittelt. Dieser wird anschließend mit den erkannten Objekten multipliziert. Im Folgenden wird die Ermittlung des Korrekturfaktors für die erkannten LKW erläutert, da der Fokus dieser Arbeit auf der Erkennung von LKW liegt.

Die vom Modell ermittelte Anzahl an LKW wird mit einer händischen Zählung der LKW verglichen. Für jedes ausgewertete Satellitenbild existiert eine Anzahl von LKW, wobei insgesamt 18 Bildern zur Verfügung stehen. In Tabelle 4.1 sind die Daten dargestellt, an welchen die Bilder aufgenommen wurden. Dazu kommen auf der einen Seite die ermittelten Werte für die Anzahl an LKW vom Algorithmus und auf der anderen Seite die Werte für die Anzahl der LKW, welche aus der händischen Zählung stammen. In der letzten Zeile sind die Werte für die Summe aller einzelnen Bilder abgetragen. Es ergibt sich dabei eine Gesamtzahl an LKW für beide Methoden.

Datum	YOLOv5 (LKW)	Händisch (LKW)
22.05.2022	62	63
18.09.2021	67	83
13.09.2021_01	46	52
13.09.2021_02	46	50
12.09.2021	112	117
11.09.2021	81	94
08.09.2021	61	63
04.09.2021	75	89
03.07.2021	76	85
11.08.2020_01	13	13
11.08.2020_02	21	24
11.08.2020_03	51	58
11.08.2020_04	62	68
11.08.2020_05	40	44
10.08.2020	38	44
31.07.2020	44	48
30.07.2020_01	52	63
30.07.2020_02	27	34
Gesamt	974	1092

Tabelle 4.1.: Die Parkplatzbelegung der Raststätte *Denkendorf* von YOLOv5 und von Hand bestimmt

Um nun die Fehlerquote zu bestimmen, wird angenommen, dass die händische Zählung 100 % beträgt. Aus dieser Annahme ergibt sich für LKW eine Fehlerquote von 12,1 %. Dieses Ergebnis deckt sich mit Abbildung 4.11, in der ebenfalls zu wenig LKW erkannt werden. Der Fehler liegt somit bei ungefähr 12 % und wird in den folgenden Auswertungen berücksichtigt, sodass es nicht zu übermäßig großen Abweichungen kommt.

4.6. LSTM für die Vorhersage von Belegungsdaten

In diesem Kapitel wird auf die Vorhersage von künstlichen Parkplatzbelegungsdaten mittels *LSTM* eingegangen. Dazu wird zunächst in Kapitel 4.6.1 auf die Aufbereitung der Daten für das Training des *LSTM* thematisiert. Im anschließenden Kapitel 4.6.2 wird der Trainingsprozess des *LSTM* erläutert.

4.6.1. Aufbereitung der Daten

Im Folgenden wurde aufgrund mangelnder Satellitendaten in ausreichend hoher Auflösung untersucht, inwiefern sich Zeitreihen für BW mittels *LSTM* künstlich gewinnen lassen. Dazu wurde angenommen, dass sämtliche Autobahnraststätten in BW als eine einzige angesehen werden können. Daneben wurde untersucht, ob sich authentische Daten bayrischer Raststätten gewinnbringend einsetzen lassen. Diese Daten wurden bereits in vorangegangen Arbeiten analysiert [40], stammen von der Webseite Bayerninfo.de und lassen sich ebenfalls für die Validierung des entwickelten Netzes nutzen.

Analog zum Objekterkennungsmodell *YOLO* (siehe Kapitel 4.3 auf Seite 27), besteht auch hier die Notwendigkeit, den Datensatz adäquat aufzubereiten. Dazu wurden zu Beginn die Aufzeichnungen der bayrischen Raststätten nach Raststätte getrennt und separat als CSV-Datei gesichert. Mithilfe der Messwerte der freien LKW-Parkplätze sowie der Anzahl der gesamten Parkplätze für LKW lässt sich die relative Auslastung der Raststätte errechnen. Dies ist notwendig, um die unterschiedliche Anzahl an Gesamtparkplätzen der Raststätten vergleichbar zu gestalten. Das Ziel ist somit die Vorhersage einer relativen Auslastung mithilfe des neuronalen Netzes. Bei Regressionsaufgaben stellt der reale Wert das Label der Daten dar, deshalb ist im Gegensatz zur Objekterkennung kein Labeling notwendig.

Dies wurde ebenfalls für die Raststätten in Baden-Württemberg durchgeführt, wobei zusätzlich der Korrekturfaktor (siehe Kapitel 4.5 auf Seite 42) berücksichtigt wurde. Der Datensatz besteht aus den erkannten LKW-Belegungen mittels des YOLOv5-Modells (siehe Kapitel 4.4 auf Seite 33) und den veröffentlichten Daten der SVZ BW (siehe Kapitel 4.2 auf Seite 25).

4.6.2. Training des LSTM

Als Datensatz wurden die aufgezeichneten Daten der Raststätte *Echinger Gfild* in Bayern der Webseite bayern-info.de genutzt. Hierfür wurden die belegten LKW-Parkplätze auf die Kapazität normiert, um eine Auslastung in Prozent zu erhalten. Die normierten Daten werden nun mit einem *bidirekionalen LSTM* trainiert. Dazu wird stets über einen Abschnitt der Daten (sogenanntes *Window*) trainiert. Die Größe des Abschnitts gibt im Anschluss den Vorhersagezeitraum an. Während des Trainings wird das trainierte Netz über das *Window* getestet, indem die Ausgabe des Netzes mit der realen Zeitreihe verglichen wird.

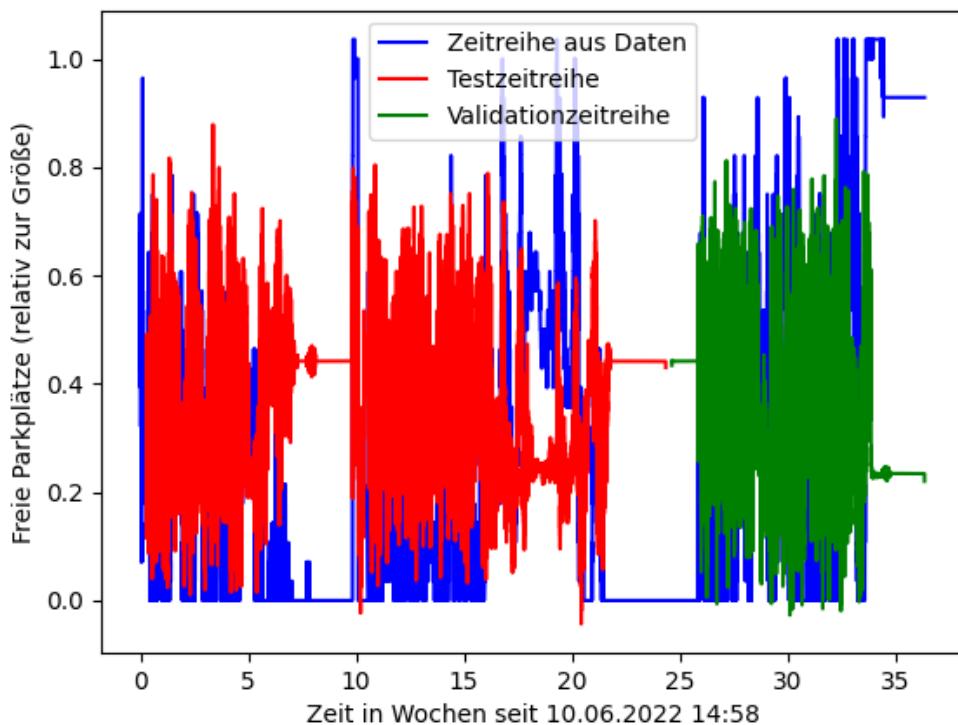


Abbildung 4.14.: Resultierende Zeitreihe des Trainings im Vergleich zur originalen Zeitreihe

Auf Abbildung 4.14 ist die während des Trainings vorhergesagte Zeitreihe abgebildet.

Zur Vorhersage wird nun das Ende der bekannten Zeitreihe als Eingabe in das Netz verwendet. Dabei wird der Abschnitt so gewählt, dass er exakt so lang ist wie das *Window*, das zum Training verwendet wurde. Um längere Abschnitte vorherzusagen wird der Vorgang mit den vorhergesagten Daten wiederholt. Durch diese Art der Vorhersage pflanzt sich allerdings der Fehler der Vorhersage fort. Jener Fehler lässt sich durch den Einsatz eines größeren Windows reduzieren. Damit wird der Grafikspeicher der Grafikkarte einer der limitierenden Faktoren der Genauigkeiten des Modells. In 4.15 ist eine durch das Modell auf Basis der bekannten Daten vorhergesagte Zeitreihe dargestellt. Hier wird deutlich, wie stark sich eine

fehlerhafte Vorhersage fortpflanzt.

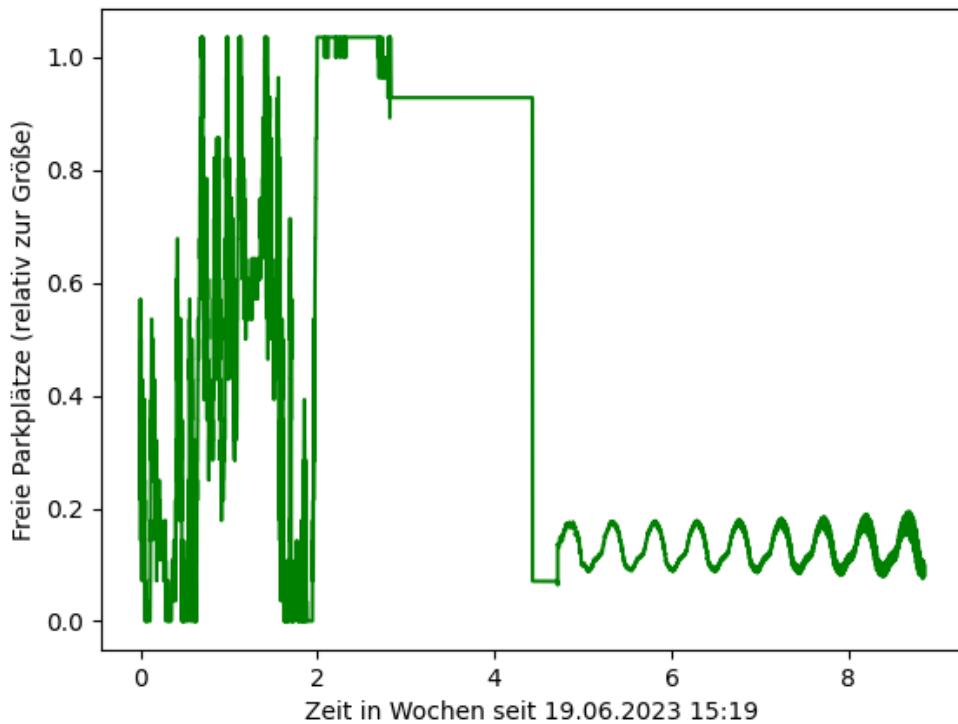


Abbildung 4.15.: Mit dem Modell vorhergesagte Zeitreihe

In der Abbildung 4.14 ist zu sehen, dass das *LSTM* einzig den ungefähren Trend erkennen kann. Die exakten Werte sind jedoch auch durch Skalierung nicht zu erreichen. Bei Vorhersagen über lange Zeiträume wie beispielsweise eine Woche ist das Modell nicht ausreichend präzise. Wie in 4.15 zu erkennen, spiegeln die Werte nicht im Mindesten den zu erwarteten Verlauf einer Parkplatzbelegung wider.

Um eine Verfälschung der Ergebnisse durch die langen Zeiträume mit konstantem Wert zu vermeiden, wurde das Training mit gefilterten Daten, die in 4.16 dargestellt sind, wiederholt.

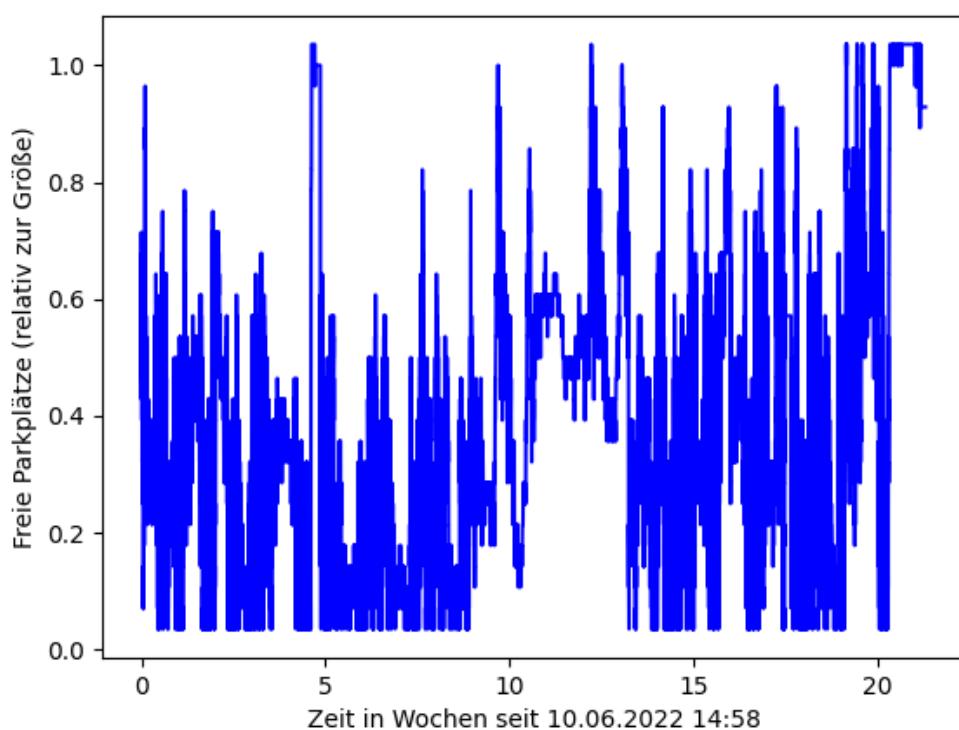


Abbildung 4.16.: Modifizierte Zeitreihe

Die Trainingsergebnisse in 4.17 zeigen, dass das Modell ähnlich schlechte Trainingsergebnisse zeigt.

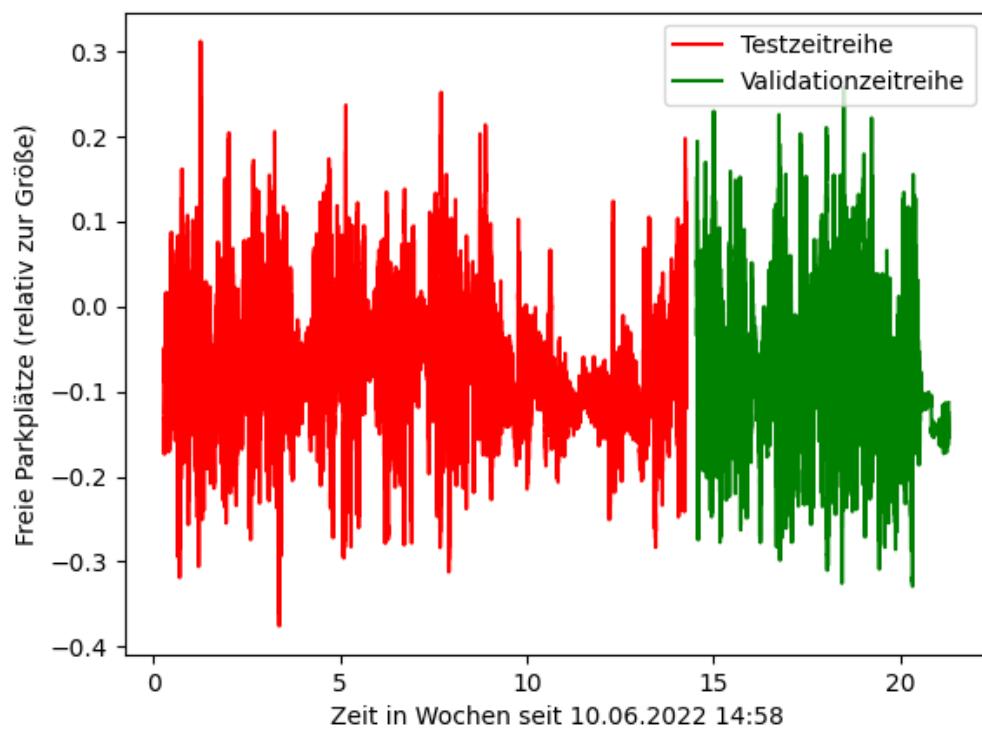


Abbildung 4.17.: Trainingsergebnisse mit der modifizierten Zeitreihe

Auch die Vorhersage wird wie in 4.18 dargestellt trotzdem nicht präziser.

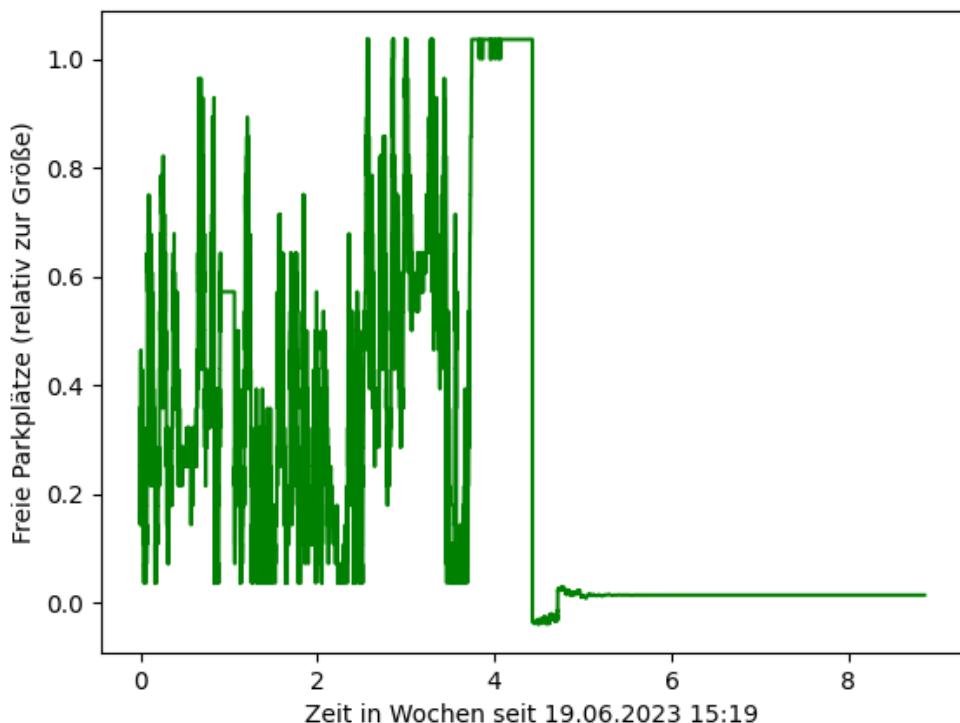


Abbildung 4.18.: Vorhersageergebnisse mit der modifizierten Zeitreihe

Da das Training des Netzes bereits bei der Vorhersage der Parkplatzbelegungsdaten für Bayern scheitert, wurde der Ansatz für Baden-Württemberg nicht weiterverfolgt. Dabei muss außerdem beachtet werden, dass der primäre Anwendungsbereich dieser Art von neuronalen Netzen die Vorhersage einzelner Datenpunkte in Abhängigkeit der vorangegangen Werte darstellt. Jedoch existieren in der Literatur vereinzelt Ansätze, um eine mehrere Datenpunkte mittels *LSTM* vorherzusagen (exemplarisch [41]). Allerdings bietet diese Methode keinen Mehrwert bei der Gewinnung einer Zeitreihe für Baden-Württemberg.

5. Ergebnisse der Arbeit

In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst. Zunächst wird in Kapitel 5.1 der erwartete Verlauf der Parkplatzauslastung diskutiert. Darauf folgt in Kapitel 5.2 die energetische Betrachtung der ermittelten Datenpunkte. Kapitel 5.3 umfasst das Fazit und bildet somit den Abschluss des Kapitels.

5.1. Erwarteter zeitlicher Verlauf der Parkplatzauslastung

Aus den Daten der bayrischen Raststätten geht hervor, dass die Parkplätze von den Morgen- bis zu den Abendstunden schwach ausgelastet sind, während sie nachts stark ausgelastet sind [40]. Dieser zeitliche Verlauf ist auf Abbildung 5.1 auf Seite 52 abgebildet. Deshalb wird ein solcher Verlauf ebenfalls für die Raststätten in Baden-Württemberg erwartet.

Nun soll dies auch für die Parkplätze in Baden-Württemberg geprüft werden. Jedoch steht eine geringe Menge an Datenpunkten für Raststätten in Baden-Württemberg zur Verfügung, die zudem zu ähnlichen Zeitpunkten aufgenommen wurden. Sämtliche Satellitenaufnahmen wurden zwischen 06:00 und 14:00 Uhr und in den Jahren 2018 bis 2023 erstellt, was nicht repräsentativ für die Auslastung der Parkplätze ist. Jedoch lässt sich überprüfen, ob die Parkplätze erwartungsgemäß zu den Mittagsstunden schwach ausgelastet sind. Dies geht sowohl aus den Daten des Objekterkennungsalgorithmus, als auch aus den Daten der SVZ BW hervor. Allerdings lässt der Datenbestand keine verlässliche Aussage über die nächtliche Auslastung baden-württembergischer Raststätten zu.

Darüber hinaus lässt sich der mittlere Wochenverlauf der 20 bayrischen Raststätten sowie die normierte Auslastung der Parkplätze betrachten. Dies auf Abbildung 5.1 dargestellt. Zusätzlich wurden die Datenpunkte, welche aus den zur Verfügung stehen Satellitenbildern gewonnen werden konnten, eingezeichnet.

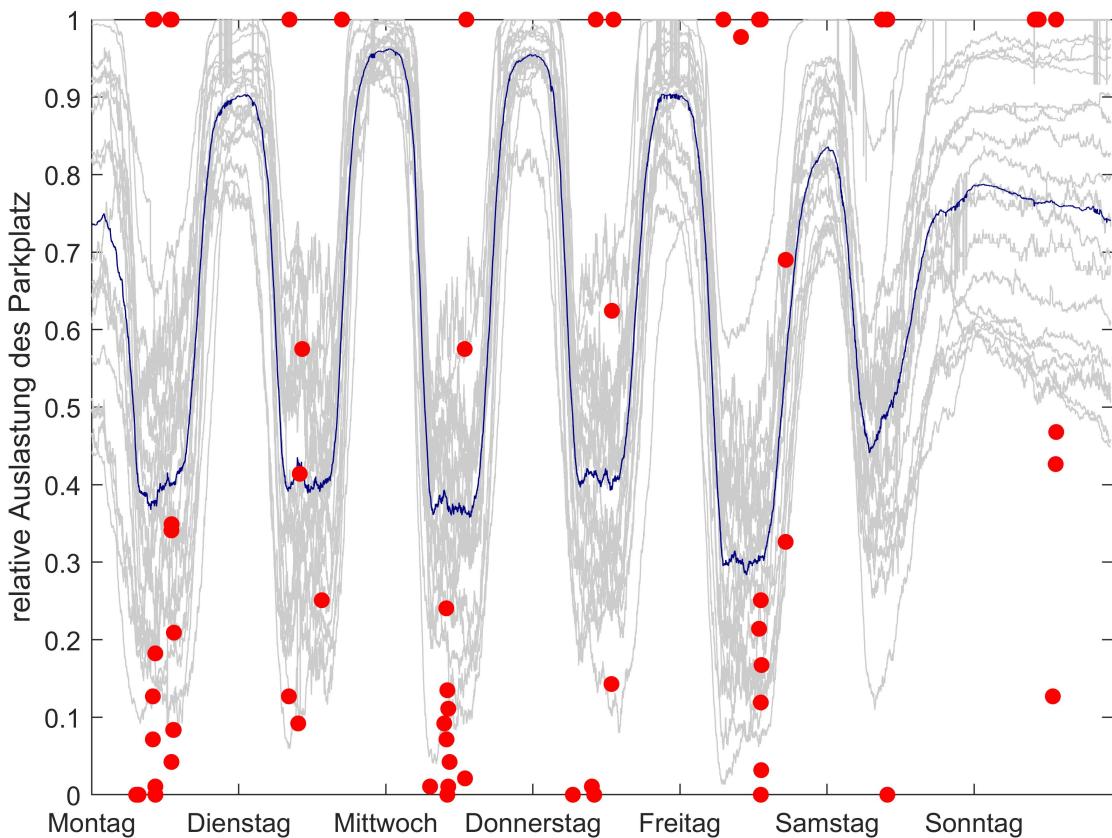


Abbildung 5.1.: Durchschnittlicher Verlauf der 20 bayrischen Raststätten (blau), auf der x-Achse: Wochentag, auf der y-Achse: relative Auslastung des Parkplatzes, zusätzlich baden-württembergische Punkte (rot), in Anlehnung an [40]

Es ist zu beobachten, dass der Verlauf durch die Satellitendaten zum Großteil bestätigt wird. Die Auslastung ist in den Nächten überaus hoch, während die Auslastung tagsüber gering ist. Jedoch lassen sich ebenso gegenläufige Werte erkennen, die auch tagsüber eine außergewöhnlich

hohe Auslastung aufweisen. Die hohen Werte am Sonntag lassen sich durch das Sonntagsfahrverbot erklären. Diese Werte finden sich auch in ähnlicher Dimension im bayrischen Datensatz. Bei den Daten mittwochs handelt es sich um Werte Anfang August und Mitte September. In diesen Zeitraum fällt der Beginn, beziehungsweise das Ende der Schulferien in Baden-Württemberg. In dieser Zeit sind die Straßen traditionell durch viele Urlauber sehr voll. Vermutlich wird dieser Tag dann oft als Ruhetag benutzt, da Staus und ähnliche Verkehrsbehinderungen in dieser Zeit überproportional oft auftreten. Die Häufung der Daten montags lässt sich durch ein leichtes Datenbias erklären. Durch die wenigen Daten bedingt, sind in unseren Daten überproportional Daten von Montagen, die gleichzeitig Feiertage sind, vorhanden. Andere Extrempunkte kommen unter anderem durch die Nähe zu anderen Feiertagen zustande. Weitere Gründe stellen eventuelle Fehler im Prozess der Datenaufbereitung oder ein Mangel an Parkplätzen für LKW dar.

5.2. Bestimmung des Ladebedarfs einer durchschnittlichen Raststätte

Die gewonnenen Datenpunkte baden-württembergischer Autobahnraststätten lassen sich nun für eine energetische Betrachtung nutzen. Da es aufgrund der Abwesenheit von Satellitenaufnahmen nicht möglich ist eine aussagekräftige Zeitreihe zu erstellen, wird nun die Auswertung zur Bestimmung des Ladebedarfs punktuell vorgenommen.

Zur Verfügung stehen circa 70 Datenpunkte aus den Satellitenbildern. Von diesen 70 Datenpunkten sind jedoch ungefähr 20 mit einer Auslastung von >100 % ausgewiesen. Zur Bestimmung des Ladebedarfs, und aufgrund der wenigen Daten, die zur Verfügung stehen, wird bei diesen Datenpunkten angenommen, dass es sich hierbei um Zeitpunkte handelt, an denen mehr LKW auf den Parkplätzen stehen als offiziell Parkplätze zur Verfügung stehen. Dies kann auf einen massiven Mangel von LKW Parkplätzen an Raststätten hinweisen oder es handelt sich beispielsweise um Raststätten

in Grenznähe, die zu erhöhten Wartezeiten führen können. Da die Parkplätze aber nur so viele Ladestationen zur Verfügung stellen, wie auch Parkplätze vorhanden sind, wird in den Fällen von >100 % die Auslastung auf 100 % reduziert. Um eine Vergleichbarkeit zwischen den Raststätten herzustellen, wurde die Auslastung auf die Anzahl an Parkplätzen bezogen. Anschließend wurde der Mittelwert für die Parkplatzanzahl aller Raststätten ermittelt, welcher 59 Parkplätze beträgt. Die Auswertung wird nun mit der Auslastung bezogen auf die mittleren LKW-Parkplätze durchgeführt. Dabei lassen sich zwei Szenarien betrachten: Einerseits, dass in einer Lenk- und Ruhepause geladen wird und andererseits, dass innerhalb der Standzeit geladen wird.

5.2.1. Szenario 1: Laden in Lenk- und Ruhepause

Ein LKW darf gemäß den Vorschriften zu Lenk- und Ruhezieten maximal 4,5 Stunden am Stück fahren, bevor die fahrende Person eine Pause von 45 Minuten einlegen muss [42]. Im ersten Szenario wird ein Worst Case Szenario angenommen. Für jenen Worst Case wird angenommen, dass der LKW seine maximale Reichweite in den 4,5 Stunden gefahren ist und voll aufgeladen werden muss. Basierend auf vorangegangen Arbeiten ergibt sich für die 18 t Klasse (Führerschein Klasse C) im Durchschnitt eine Kapazität von 587,1 kWh. Aus der Pausenzeit von 45 Minuten und der ermittelten durchschnittlichen Kapazität von 587,1 kWh lässt sich die Ladeleistung von 782,8 kW bestimmen. Aus der Auslastung pro Datenpunkt und der mittleren Anzahl an Parkplätzen wird die Anzahl an LKW pro Datenpunkt bestimmt. Mit der Ladeleistung wird nun eine Gesamtlast pro Datenpunkt berechnet. Da eine zeitliche Auflösung bei so großen Zeitsprüngen keinen Mehrwert bietet, wird die Last in einem Histogramm dargestellt, welches auf Abbildung 5.2 zu sehen ist.

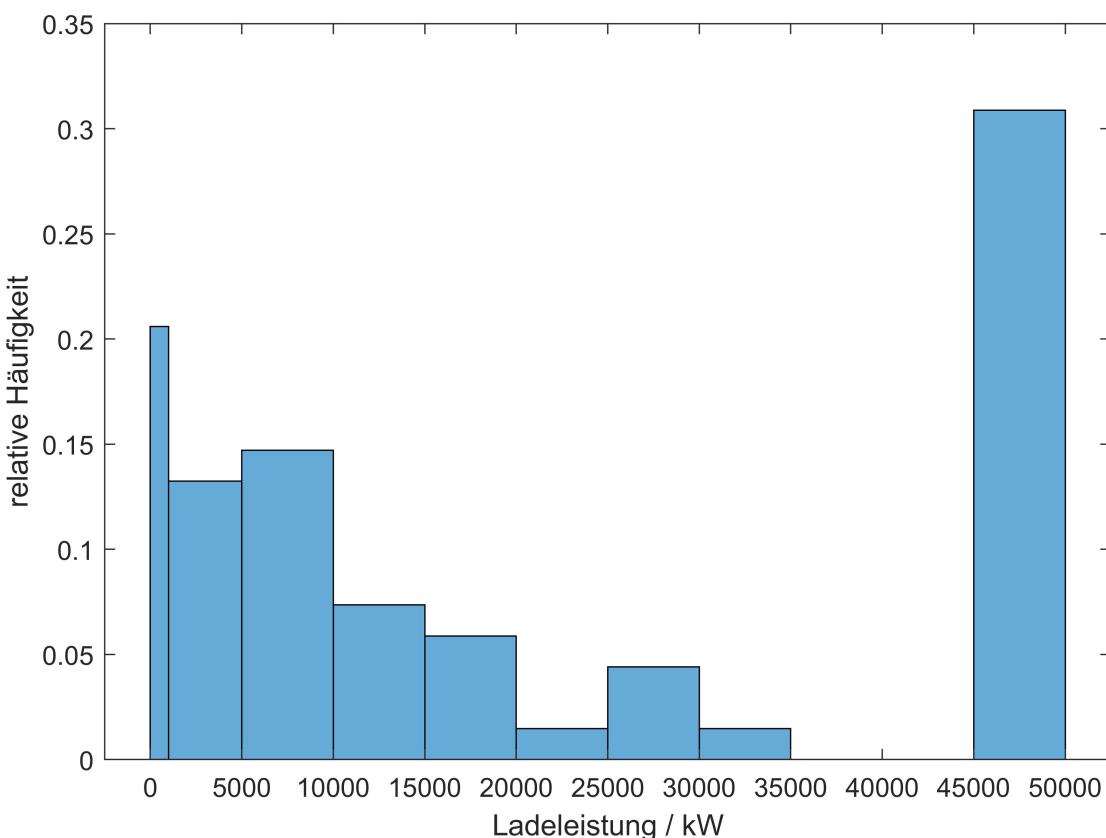


Abbildung 5.2.: Histogramm, auf der x-Achse: Ladeleistung in kW, auf der y-Achse: relative Häufigkeit

Die minimale Last beträgt 0 kW, da es Zeitpunkte gibt, zu denen keine LKW auf dem Parkplatz laden. Die maximale Last tritt auf, wenn die Auslastung 100 % beträgt. Hierbei beläuft sich die Last auf 46185,2 kW. Daneben lässt sich erkennen, dass die Last entweder ihren Maximalwert erreicht oder sich in niedrigeren Bereichen abspielt. Zusätzlich wurde die Last pro Parkplatz ausgewertet, was auf Abbildung 5.3 in Form eines *Boxplots* zu sehen ist.

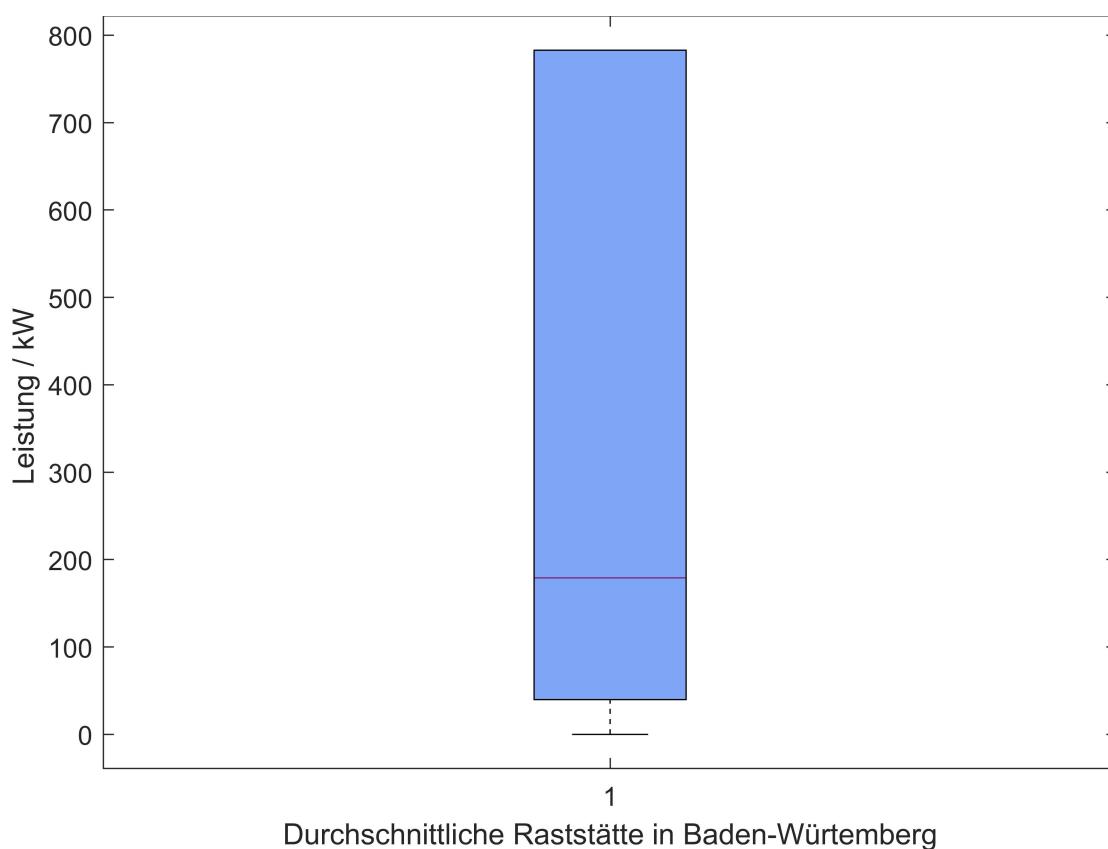


Abbildung 5.3.: Boxplot für Szenario 1, auf der x-Achse: Durchschnittliche Raststätte in Baden-Württemberg, auf der y-Achse: Ladeleistung in kW

Insgesamt liegt die Last pro Parkplatz zwischen 0 und 782,8 kW. Der Median liegt hier bei 179 kW und der Mittelwert bei 332 kW. Das *25th Percentile* liegt bei 39,8 kW, während das *75th Percentile* auf dem Maximalwert von 782,8 kW liegt. Überdies existieren keine Ausreißer.

5.2.2. Szenario 2: Laden in Standzeit

Im zweiten Szenario wird betrachtet, wie sich die Last an der Raststätte verhält, wenn die fahrende Person neun Stunden gefahren ist und eine Standzeit von elf Stunden einhalten muss. Analog zum ersten Szenario wird hier die Ladeleistung über die Kapazität und die Standzeit bestimmt.

Diese Ladeleistung wird ebenfalls mit der Auslastung und der mittleren Anzahl an Parkplätzen verrechnet. Für das zweite Szenario wird erneut ein Histogramm für die Last sowie ein *Boxplot* für die Last pro Parkplatz angefertigt. Abbildung 5.4 zeigt das Histogramm.

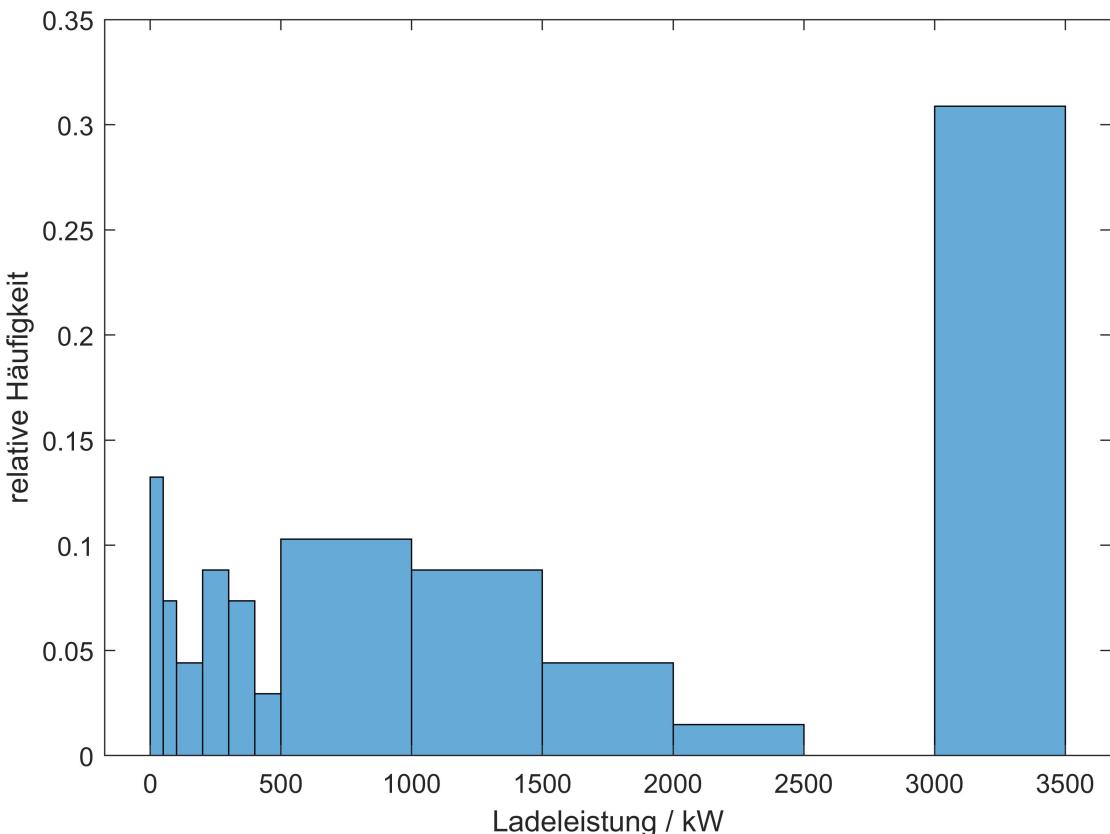


Abbildung 5.4.: Histogramm für Szenario 2, auf der x-Achse: Ladeleistung in kW, auf der y-Achse: relative Häufigkeit

Auch hier ergibt sich ein ähnliches Bild wie schon im ersten Szenario. Entweder sind es Zeiten mit einer Auslastung von circa 100 % oder es sind Zeitpunkte mit einer geringeren Auslastung und daher auch einer geringeren Last. Die minimale Last beträgt ebenfalls 0 kW. Daneben existieren erneut Zeitpunkte, an denen sich keine LKW auf dem Parkplatz befinden. Die maximale Last beträgt in diesem Szenario 3149 kW.

Abbildung 5.5 zeigt den *Boxplot* für die Ladeleistung pro Parkplatz.

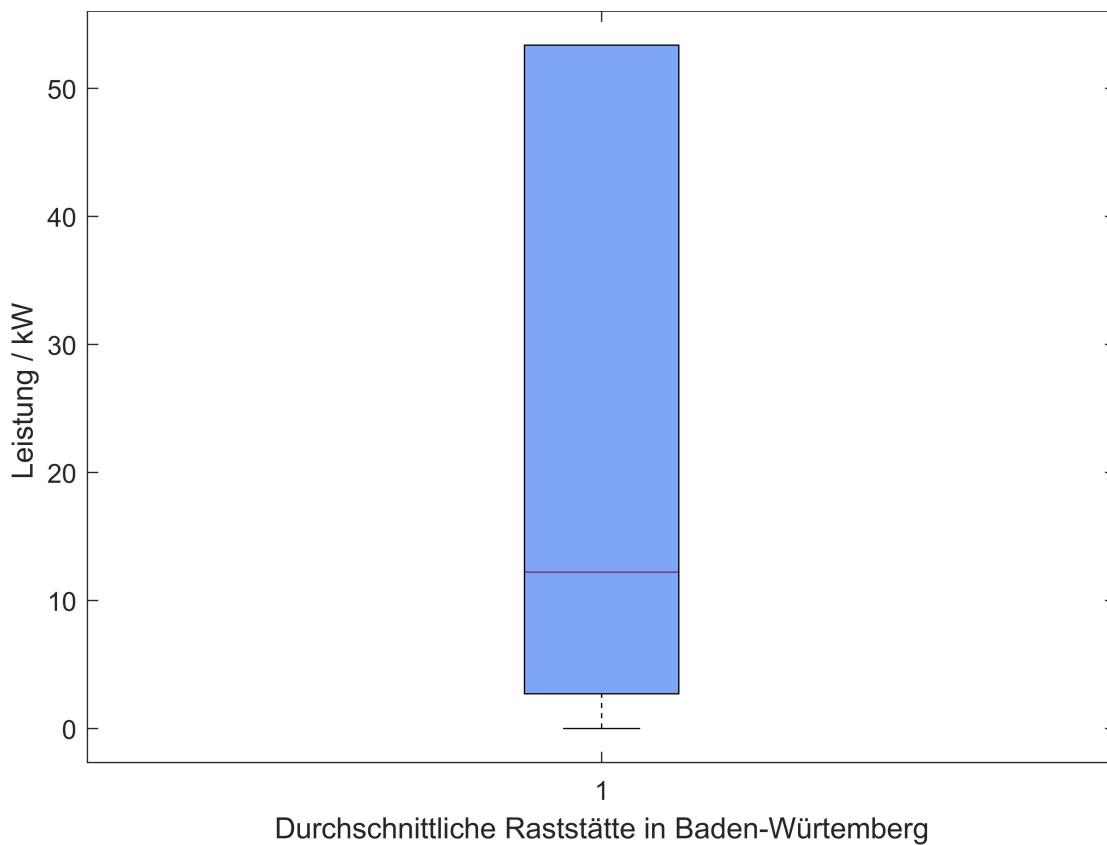


Abbildung 5.5.: *Boxplot* für Szenario 2, auf der x-Achse: Durchschnittliche Raststätte in Baden-Württemberg, auf der y-Achse: Ladeleistung in kW

Die Last liegt in diesem Szenario zwischen 0 und 53,4 kW, wobei der Median diesmal 12,2 kW beträgt, und der Mittelwert beläuft sich auf 22,6 kW. Das *25th Percentile* liegt bei 2,7 kW und das *75th Percentile* fällt nochmals mit dem maximalen Wert zusammen auf 53,4 kW. In Szenario 2 treten ein weiteres Mal keine Ausreißer auf.

5.3. Fazit

Es wurde ein Bilderkennungsmodell auf Basis eines YOLOv5-Modells trainiert, welches LKW auf Satellitenaufnahmen hinreichender Auflösung mit einer Genauigkeit von 12 % erkennt.

Allerdings existieren im Moment noch zu wenig hochauflösende Satellitenaufnahmen von Raststätten in Baden-Württemberg für die Auswertung und Voraussage von deren Belegung.

Daher wurden sämtliche Raststätten in BW als eine betrachtet. Die Parkplätze wurden mithilfe der Anzahl der gesamten LKW-Parkplätze normiert, um eine prozentuale Auslastung zu erhalten.

Ein Modell nach dem Prinzip eines LSTM-Modells konnte aufgrund der Datenlage aber keine Voraussage der Belegung von Raststätten in Baden-Württemberg treffen.

Aus den Daten der Raststätten aus Bayern lässt sich ableiten, dass die LKW-Parkplätze tagsüber kaum ausgelastet sind. Die erkannten LKW auf den baden-württembergischen Raststätten bestätigen diesen Verlauf für Baden-Württemberg.

Der diskutierte Ansatz kann auch keine Aussage über die Belegung der Raststätten bei Nacht treffen, welche mutmaßlich deutlich höher sein wird als tagsüber, wie aus den Daten aus Bayern zu erkennen.

Im nächsten Schritt wurde trotzdem die Ladeleistung von LKW auf Raststätten mit den zur Verfügung stehenden Punkten berechnet. Hierzu wurden die zwei Szenarien *Laden in Lenk- und Ruhepause* und *Laden in Standzeit* betrachtet. Die maximale Ladelast wurde im ersten Szenario zu 728,8 kW und im zweiten zu 53,4 kW pro Parkplatz bestimmt.

6. Zusammenfassung und Ausblick

Die Kombination der Elektrifizierung des Straßenverkehrs und einer adäquaten Ladeinfrastruktur stellt einen elementaren Schlüssel bei der Erreichung der Klimaschutzziele dar. Dabei spielen Autobahnraststätten eine entscheidende Rolle, da dort in kurzen Zeitabständen starke Belastungen durch LKW sowie PKW zu erwarten sind. Folglich sollte aufgrund fehlender Daten im Rahmen dieser Arbeit der Tagesverlauf der Auslastung einer Autobahnraststätte in Baden-Württemberg untersucht werden. Zu diesem Zweck sollten Satellitenbilder für die Gewinnung der Datenbasis eingesetzt werden. Schließlich sollten jene Daten für die Ableitung von Netzplanungsansätzen für die Auslegung des Netzanschlusses einer Autobahnraststätte dienen.

Zu Beginn der Arbeit wurden relevante Konzepte des *Deep Learnings* sowie dessen Teilgebiets, der Objekterkennung, thematisiert. Um einen Überblick über die eingesetzten Modelle zu erhalten, wurden *YOLO* sowie *LSTM* eingeführt. In diesem Kontext wurden außerdem einige relevante Metriken für die Bewertung von Objekterkennungsalgorithmen erläutert. Im Anschluss wurde eine umfassende Recherche über die Verfügbarkeit von Satellitenbildern von süddeutschen Autobahnraststätten durchgeführt. Diese ergab, dass lediglich Aufnahmen der *SkySat*-Mission sowie *Google Earth Pro* eine adäquate Auflösung bei gleichzeitigem kostenfreiem Zugang bieten. Aus diesem Grund wurden jene Satellitenbilder als Quelle für die Erstellung eines Datensatzes mithilfe des Objekterkennungsalgorithmus eingesetzt. Eine weitere Quelle stellt die "VerkehrsInfo BW" dar, die von der Straßenverkehrszentrale Baden-Württemberg betrieben wird. Allerdings

werden diese Werte unregelmäßig in großen Zeitabständen aktualisiert, weshalb sie ebenfalls nicht für die Erstellung einer Zeitreihe nutzbringend einsetzbar ist.

Im Hinblick auf die Erkennung kleiner Objekte, lässt sich festhalten, dass das Modell *YOLO* geeignet ist. Daher wurden unterschiedliche Modellgrößen untersucht und verglichen, wobei das Modell der Größe *Large* bemerkenswerte Ergebnisse lieferte. Schließlich wurde diese Modellgröße für die Erkennung von LKW auf Satellitenbildern trainiert und eingesetzt.

LSTMs eignen sich für die Vorhersage von zeitlich zusammenhängenden Daten wie Zeitreihen. Aus diesem Grund sowie niedriger Verfügbarkeit von Daten aus Baden-Württemberg wurde untersucht, inwiefern sich künstliche Zeitreihen mittels *LSTM* generieren lassen. Dies gelang jedoch nicht, da nicht genügend Stichproben vorhanden waren, um realistische Vorhersagen treffen zu können. Deshalb wurde die energetische Betrachtung zur Auslegung eines Netzanschlusses einer baden-württembergischen Autobahnraststätte in dieser Arbeit punktuell durchgeführt.

Für die Belastung an der Raststätte kann zwischen Zeiten, an denen sehr große Ladeleistungen erwartet werden, und Zeiten, an denen sich die Ladeleistung in einem überschaubaren Rahmen bewegt, unterschieden werden. Im ersten Szenario besteht die Möglichkeit, dass Ladeleistungen von über 45 MW erreicht werden, wenn eine erhebliche Anzahl an LKW laden müssen. Da diese Zeitpunkte vermutlich tagsüber vorkommen, wäre es beispielsweise möglich mit Photovoltaik über den Parkplätzen die Spitzen zu reduzieren.

Aus dem zweiten Szenario wird ersichtlich, dass die Belastung deutlich niedriger ist. Selbst bei voller Auslastung wird die Grenze von 3,5 MW nicht überschritten. Zu vielen Zeitpunkten beläuft sich die Belastung lediglich auf 1 – 2 MW. Unter Berücksichtigung des Vorhandenseins von Dachphotovoltaik eines Speichers oder eines Windrads/Windparks, wird die Belastung zudem merklich reduziert. Ein gesamt gedachter Ansatz lässt sich bereits zum Zeitpunkt dieser Arbeit für einige Zeitpunkte einen möglichen Lösungsansatz bereitstellen. Für die Stoßzeiten mit außerordentlich hohem Bedarf, muss ein Konzept entwickelt werden, in welchem

unter anderem der Anschluss bedacht wird.

Zum Zeitpunkt dieser Arbeit besteht keine Möglichkeit, Zeitreihen von baden-württembergischen Autobahnraststätten mithilfe von Satellitenbildern zu erstellen. Überdies bietet das Detektierungssystem des Landes Baden-Württemberg an den Autobahnraststätten der A5 keinen Mehrwert, da nur äußerst selten aktualisierte Daten zur Verfügung gestellt werden. Daneben existieren weitere Möglichkeiten, um Datenpunkte mithilfe von *Deep Learning* artifiziell zu erhalten, wie beispielsweise *WaveNet* sowie *Transformer*. Diese Ansätze lassen sich künftig verfolgen, um künstliche Zeitreihen zu generieren, insofern sich das Vorhandensein von Satellitenaufnahmen nicht erhöht. Unter der Annahme, dass sich die Verfügbarkeit von hochauflösten Satellitenbildern steigert, lässt sich festhalten, dass das trainierte YOLO-Modell für eine zukünftige Erstellung von Zeitreihen einsetzbar ist.

Literaturverzeichnis

- [1] *BMDV - Klimaschutzziele und Beschlüsse*, 4/25/2023.
- [2] J. Xu, „Deep Learning for Object Detection: A Comprehensive Review“, *Towards Data Science*, 11/9/2017.
- [3] W. Liu, D. Anguelov, D. Erhan u. a., *SSD: Single Shot MultiBox Detector*, 2015.
- [4] S. Ren, K. He, R. Girshick und J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*.
- [5] T.-Y. Lin, P. Goyal, R. Girshick, K. He und P. Dollár, *Focal Loss for Dense Object Detection*.
- [6] P. Hu und D. Ramanan, *Finding Tiny Faces*.
- [7] *Papers with Code - Small Object Detection*, 6/12/2023.
- [8] *Deep Learning* (Adaptive Computation and Machine Learning Series). Cambridge, Mass.: MIT Press Ltd, 2017, isbn: 9780262035613.
- [9] P. Jaccard, „Distribution de la flore alpine dans le Bassin des Dranses et dans quelques régions voisines“, 1901.
- [10] L. F. Da Costa, *Further Generalizations of the Jaccard Index*, 2021.
- [11] J. Redmon, S. Divvala, R. Girshick und A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection“, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2016, S. 779–788, isbn: 978-1-4673-8851-1.
- [12] Glenn Jocher, Alex Stoken, Jirka Borovec u. a., *ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements*, 2020.

- [13] S. Hochreiter und J. Schmidhuber, „Long short-term memory“, *Neural computation*, Jg. 9, Nr. 8, S. 1735–1780, 1997, issn: 0899-7667.
- [14] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, Bd. 385, isbn: 978-3-642-24796-5.
- [15] J. Elman, „Finding structure in time“, *Cognitive Science*, Jg. 14, Nr. 2, S. 179–211, 1990, issn: 03640213.
- [16] N. Gruber und A. Jockisch, „Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text?“, *Frontiers in artificial intelligence*, Jg. 3, S. 40, 2020.
- [17] B. Lindemann, T. Müller, H. Vietz, N. Jazdi und M. Weyrich, „A survey on long short-term memory networks for time series prediction“, *Procedia CIRP*, Jg. 99, S. 650–655, 2021, issn: 22128271.
- [18] C. Fjellström, *Long Short-Term Memory Neural Network for Financial Time Series*.
- [19] TensorFlow, *API Documentation* | *TensorFlow v2.12.0*, 5/26/2023.
- [20] *PlanetScope - Earth Online*, 7/6/2023.
- [21] A. Paszke, S. Gross, F. Massa u. a., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*.
- [22] M. Abadi, P. Barham, J. Chen u. a., *TensorFlow: A system for large-scale machine learning*.
- [23] B. Pang, E. Nijkamp und Y. N. Wu, „Deep Learning With TensorFlow: A Review“, *Journal of Educational and Behavioral Statistics*, Jg. 45, Nr. 2, S. 227–248, 2020, issn: 1076-9986.
- [24] K. Team, *Keras: Deep Learning for humans*, 29.08.2023.
- [25] H. J. Kramer, „CubeSat Concept“, *European Space Agency*, 30.5.2012.
- [26] Tanya N. Harrison, „Introduction to the Planet Constellations of Satellites and Imagery“,
- [27] „Combined Imagery Product Spec FINAL | May 2022“,

- [28] Wikipedia, Hrsg., *RapidEye*, 2023.
- [29] E. S. Agency, „RapidEye“,
- [30] E. S. Agency, „SkySat“,
- [31] *SkySat Full Archive and New Tasking - Earth Online*.
- [32] Google Earth Studio, *FAQ – Google Earth Studio*, 4/15/2022.
- [33] Baden-Württemberg.de, *Dynamische Lkw-Stellplatzanzeigen entlang der A 5 in Betrieb*, 9/19/2023.
- [34] *VerkehrsInfo BW: Karte*, 4/17/2023.
- [35] P. Viola, M. Jones u. a., „Robust real-time object detection“, *International journal of computer vision*, Jg. 4, Nr. 34-47, S. 4, 2001.
- [36] GitHub, *GitHub - SkalskiP/make-sense: Free to use online tool for labelling photos*. <https://makesense.ai>, 24.09.2023.
- [37] D. Wahyudi, I. Soesanti und H. A. Nugroho, „Toward Detection of Small Objects Using Deep Learning Methods: A Review“, in *2022 14th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, 2022, S. 314–319, isbn: 978-1-6654-6077-4.
- [38] A. G. Kravets, A. A. Bolshakov und M. Shcherbakov, *Cyber-Physical Systems: Modelling and Intelligent Control*. Cham: Springer International Publishing, 2021, Bd. 338, isbn: 978-3-030-66076-5.
- [39] A. Froidevaux, A. Julier, A. Lifschitz u. a., „Vehicle Detection and Counting from VHR Satellite Images: Efforts and Open Issues“, in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, 2020, S. 256–259, isbn: 978-1-7281-6374-1.
- [40] Saša Škorić, *Untersuchung der Netzintegration von Ladeinfrastruktur für elektrische Lkw an Autobahnraststätten*, Bachelorarbeit, Stuttgart, 2023.
- [41] R. Chandra, S. Goyal und R. Gupta, „Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction“, *IEEE Access*, Jg. 9, S. 83 105–83 123, 2021.

- [42] EUR-Lex - 32006R0561 - EN - EUR-Lex, 24.09.2023.

Anhang

A. Unterstützende Grafik

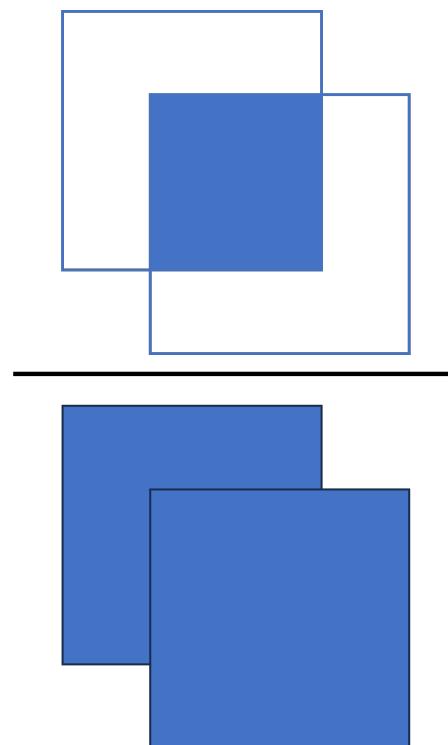


Abbildung A.1.: Grafische Darstellung der *Intersection over Union*

B. Trainingsergebnisse

Auf Abbildung B.1 sind die Trainingsmetriken des YOLOv5-Modells mit der Eingabeauflösung 320x320 Pixel sichtbar.

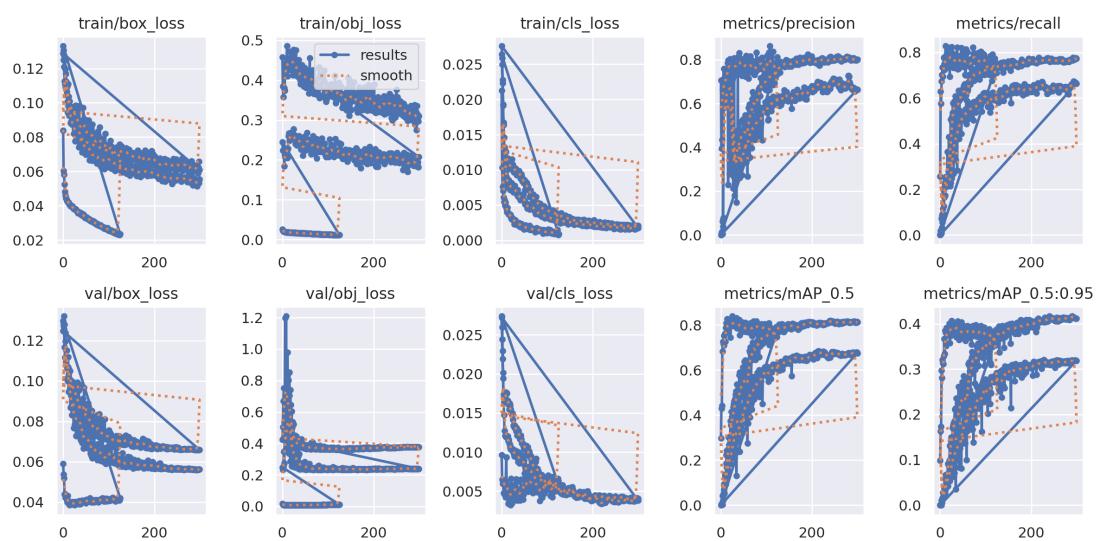


Abbildung B.1.: Trainingsmetriken des YOLOv5-Modells mit 320x320 Pixel Eingabeauflösung

Auf Abbildung B.2 sind die Trainingsmetriken des YOLOv5-Modells mit der Eingabeauflösung 640x640 Pixel sichtbar.

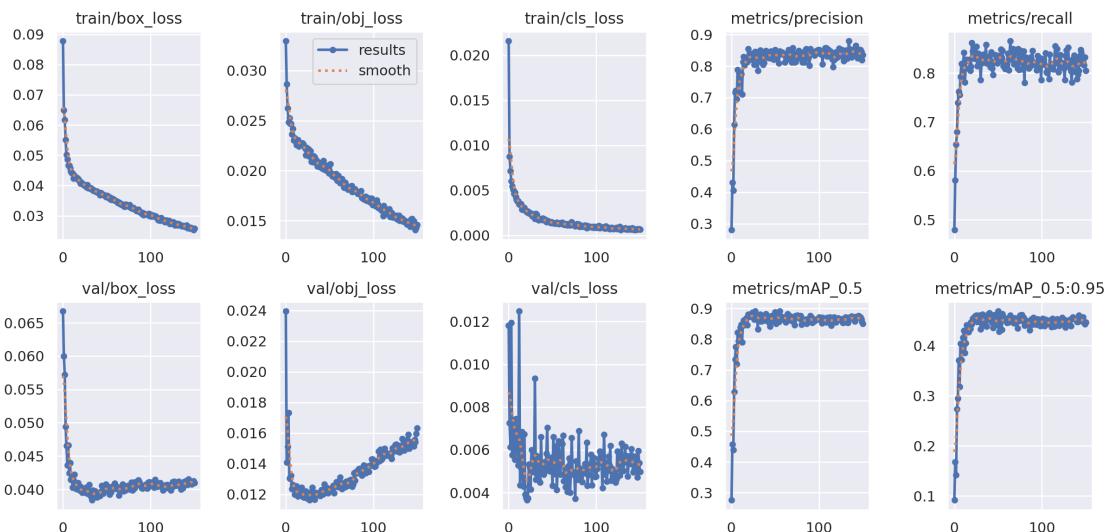


Abbildung B.2.: Trainingsmetriken des YOLOv5 Modells mit 640x640 Pixel Eingabeauflösung

Auf Abbildung B.3 sind die Trainingsmetriken des YOLOv5-Modells mit der Eingabeauflösung 960x960 Pixel sichtbar.

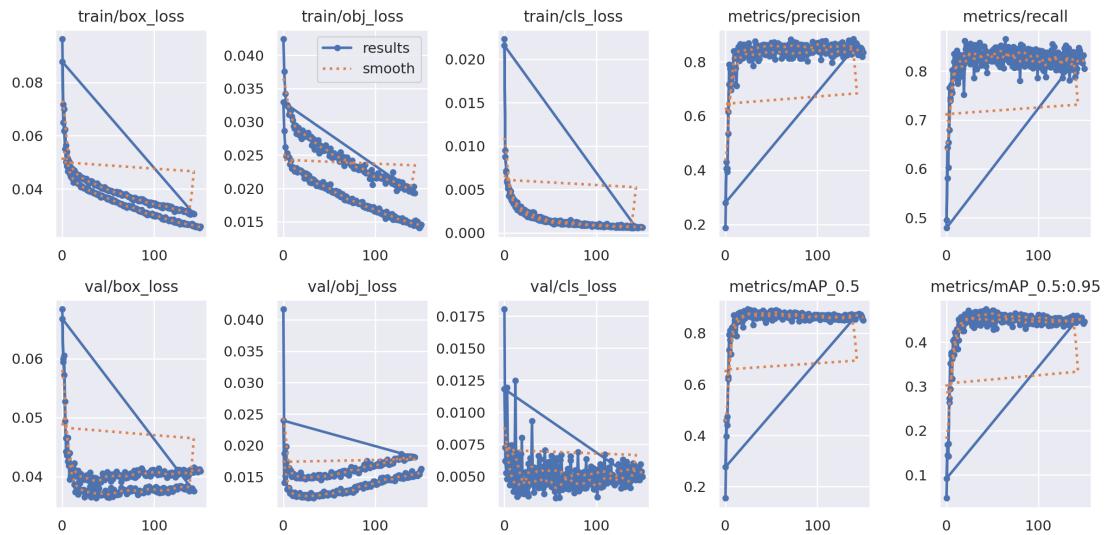


Abbildung B.3.: Trainingsmetriken des YOLOv5 Modells mit 960x960 Pixel Eingabeauflösung

Auf Abbildung B.4 sind die Trainingsmetriken des YOLOv5-Modells mit der Eingabeauflösung 1080x1080 Pixel sichtbar.

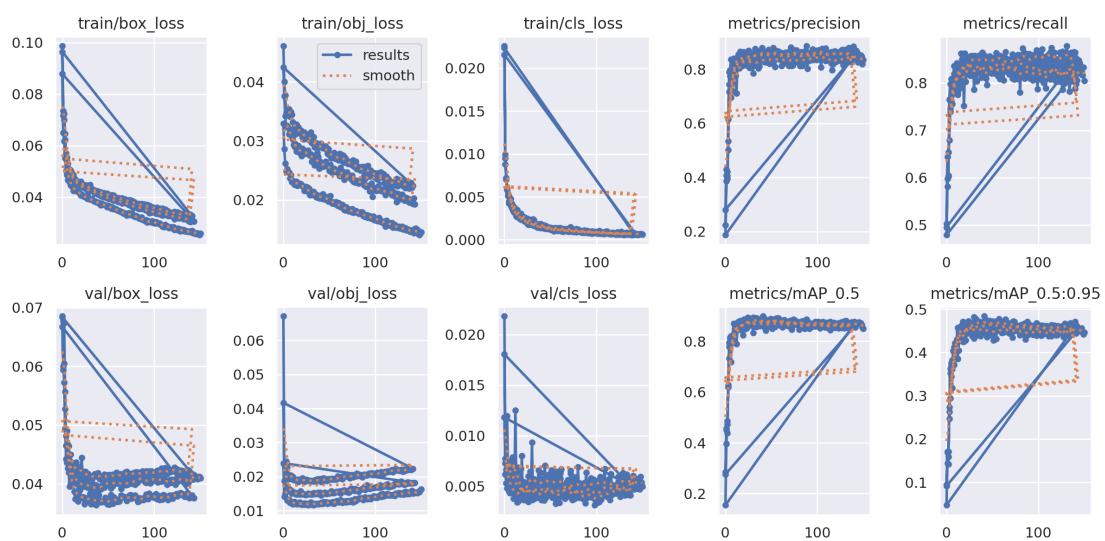


Abbildung B.4.: Trainingsmetriken des YOLOv5 Modells mit 1080x1080 Pixel Eingabeauflösung

Auf Abbildung B.5 sind die Trainingsmetriken des YOLOv5-Modells der Größe XLarge mit 960x960 Pixel Eingabeauflösung abgebildet.

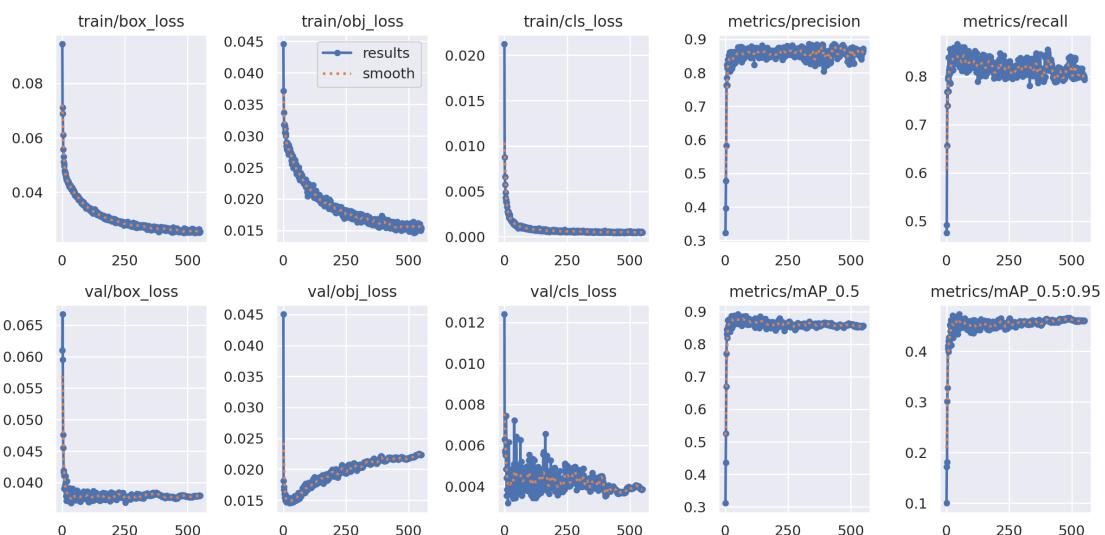


Abbildung B.5.: Trainingsmetriken des YOLOv5-Modells in der Größe XLarge mit 960x960 Pixel Eingabeauflösung

Daneben sind auf der folgenden Abbildung B.6 die Labelmetriken des YOLOv5-Modells der Größe *XLarge* mit 960x960 Pixel Eingabeauflösung zu sehen.

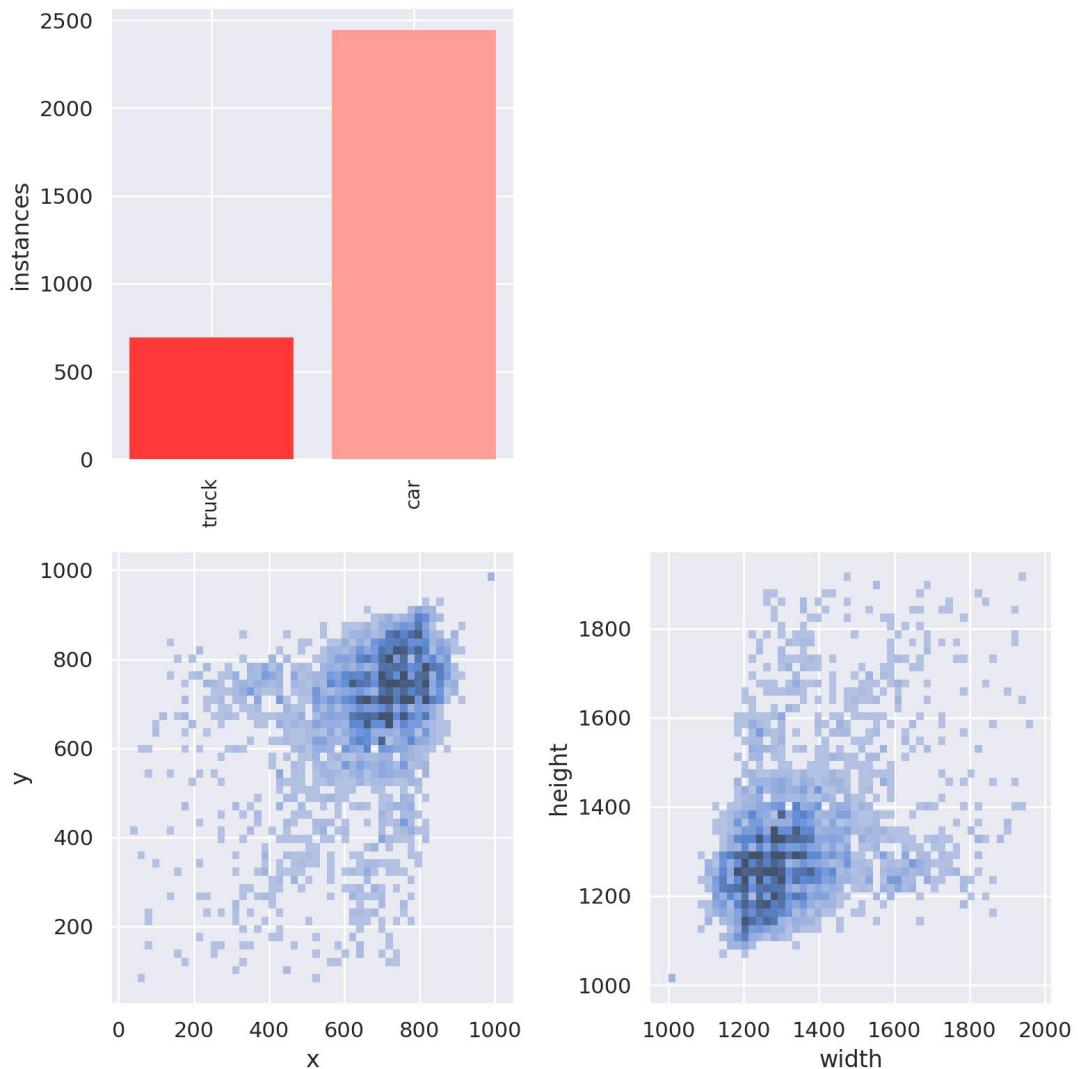


Abbildung B.6.: Labelmetriken des 960x960 Pixel Modells der Größe *XLarge*

Zusätzlich ist auf Abbildung B.7 das Korrelogramm der Labels des YOLOv5-Modells der Größe *XLarge* mit 960x960 Pixel Eingabeauflösung

abgebildet.

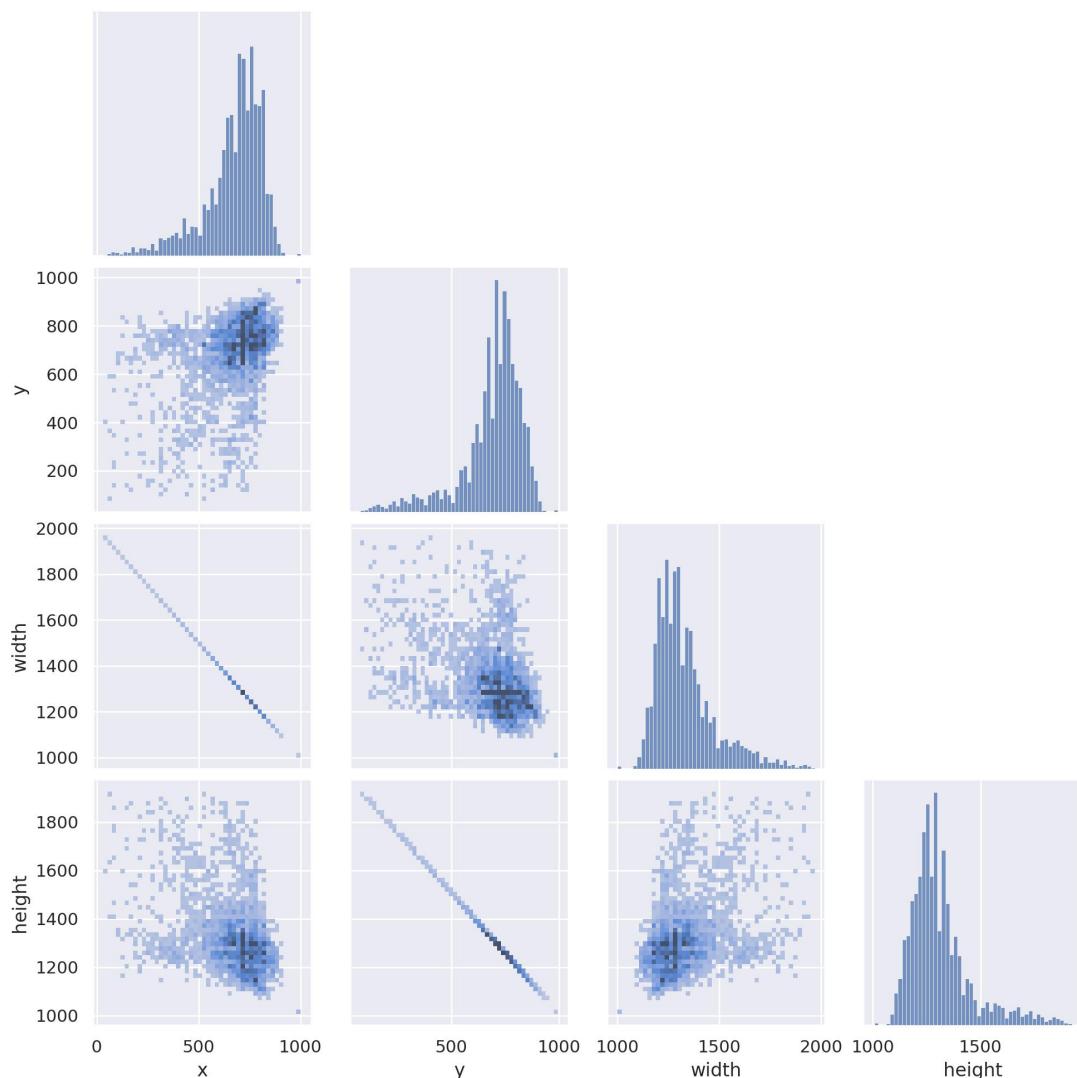


Abbildung B.7.: Korrelogramm der Labels des 960x960 Pixel Modells der Größe *XLarge*

Überdies folgen auf der Abbildung B.8 die Trainingsmetriken des YOLOv5-Modells der Größe Large mit 960x960 Pixel Eingabeauflösung, wobei der Datensatz ausschließlich aus *SkySat*-Aufnahmen besteht.

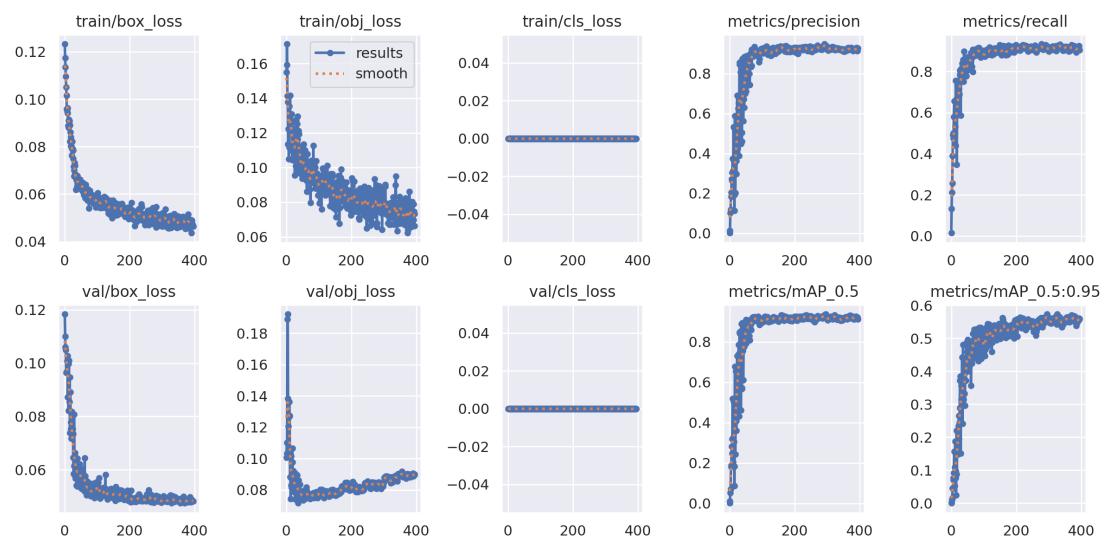


Abbildung B.8.: Trainingsmetriken des YOLOv5-Modells der Größe *Large* mit 960x960 Pixel Eingabeauflösung, Datensatz ausschließlich aus *SkySat*-Aufnahmen

Abschließend ist auf Abbildung B.9 der Verlauf der *mAP* über den Epochen ohne *Early Stopping* abgebildet. Bei *Early Stopping* handelt es sich um eine Methode, um das Auswendiglernen der Daten zu verhindern.

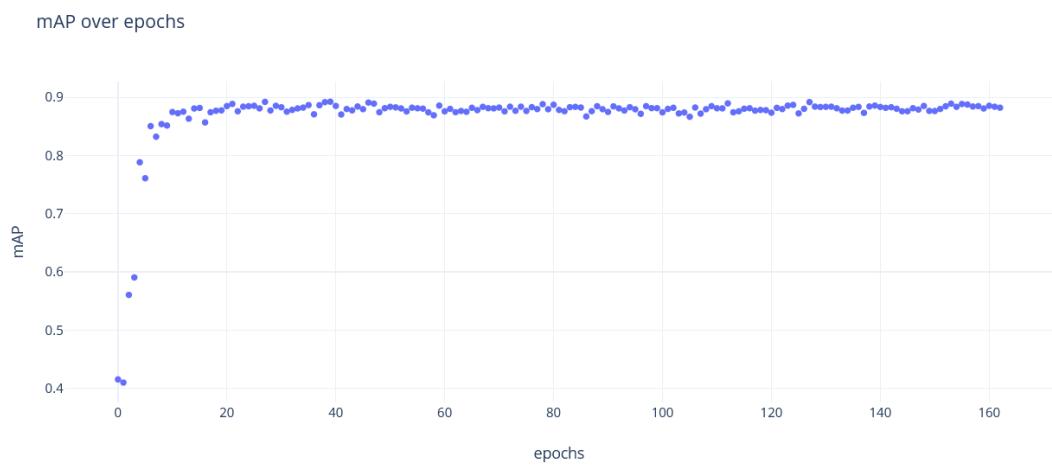


Abbildung B.9.: *mAP* über den Verlauf der Epochen eines Modells ohne *Early Stopping*