

Todo list

sämtliche Abkürzungen mit glocaries einbinden und verwalten	1
gefühlt ein bisschen zu viel wiederholung	18
warum kann yolov8 obb? was wurde dort eingefügt?	21
weniger Funktionsnamen	25
Bilder in Originalgröße in Appendix packen	35
nicht nur erläutern was man sieht sondern auch was man daraus folgert (analyse) im Bildunterschrift	35
nächster Satz ist eigentlich analyse (auskommentiert)	36
ist auch eigentlihc analyse	41
ist eigenltihc analyse	41
Beschreibung Figures länger machen, mehr Analyse weniger "was sieht man"	45
mehr vielleicht	58
drinlassen? oder löschen? ist eigentlich redundant!	59

Deep Learning for Small Vehicle Detection and Classification on High-Resolution Multispectral Remote Sensing Imagery

MASTER'S THESIS
in partial fulfilment of the requirements for the degree of
MASTER OF SCIENCE

University of Münster
Institute for Geoinformatics

First Supervisor:
Prof. Dr. Benjamin Risse

Second Supervisor:
Dr. Christian Knoth

Mentor:
Sebastian Thiele

Submitted by:
Timo Lietmeyer

Münster, September 2025

Deep Learning for Small Vehicle Detection and Classification on High-Resolution Multispectral Remote Sensing Imagery

Deep Learning zur Detektion und Klassifikation kleiner
Fahrzeuge auf hochauflösenden multispektralen
Fernerkundungsbildern

Abstract

Die präzise Detektion und Klassifikation kleiner Fahrzeuge in hochauflösten Fernerkundungsdaten ist für Anwendungen wie Verkehrsüberwachung und Stadtplanung essenziell. Ziel dieser Arbeit war es, den Einfluss zusätzlicher spektraler Kanäle auf die Leistung bei der Objekterkennung zu untersuchen. Experimente auf multispektralen Luftbildern zeigen, dass Infrarotkanäle die Detektionsgenauigkeit kleiner Fahrzeuge verbessern, während Vegetationsindizes wie der NDVI nicht konsistent vorteilhaft sind. RGB-Daten allein liefern bereits solide Ergebnisse, wobei die Verfügbarkeit umfangreicher Trainingsdaten den größten Einfluss auf die Modellperformance hat. YOLOv9 erweist sich dabei als besonders geeignet für kleine Objekte. Die Ergebnisse bestätigen das Potenzial tiefenlernender Verfahren für die Fahrzeugdetektion in multispektralen Fernerkundungsdaten und eröffnen Perspektiven für weiterführende Untersuchungen, etwa zur Nutzung multitemporaler Daten oder Skalierungseffekten.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Structure of the thesis	3
2. Fundamentals	5
2.1. Computer Vision	5
2.2. Machine and Deep Learning	5
2.2.1. Backpropagation	8
2.2.2. Vanishing-Gradient-Problem	8
2.2.3. Exploding-Gradient-Problem	8
2.2.4. Overfitting	9
2.3. One Stage and Two Stage Detectors	10
2.4. Milestones and Historical Development of YOLO Architectures	10
2.5. Evaluation Metrics	12
2.5.1. K Fold Cross Validation	12
2.5.2. Intersection over Union (IoU)	12
2.5.3. Mean Average Precision (mAP)	13
3. State of research	15
4. Methodology	19
4.1. Database	20
4.1.1. Vehicle Detection on Aerial Images (VEDAI) Dataset	20
4.1.2. Dataset for Objectdetection in Aerial Images (DOTA) 1.5	21
4.2. YOLOv9	21
4.2.1. YOLOv9u by Wong Kin Yiu	22
4.2.2. YOLOv9e by Ultralytics	23
4.3. Programming Environment	23
4.3.1. Python Packages	24
4.3.2. Additional Python-Packages for data analysis	25

4.4. Implementation Preprocessing	25
4.4.1. DOTA Dataset	25
4.4.2. VEDAI Dataset	26
4.4.3. Fold Creation	29
4.5. Challenges Preprocessing	31
4.6. 6-Fold-Cross-Validation	31
4.7. High-Performance-Cluster PALMA	32
4.8. Bash Scripts	33
5. Results	35
5.1. Oriented Bounding Boxes and Axis Aligned Bounding Boxes	35
5.2. Permutation Experiments	41
5.2.1. Comparison with Confusion Matrices	44
5.3. Ablation Studies	50
6. Discussion	57
7. Conclusion and Outlook	61
7.1. Conclusion	61
7.2. Outlook	61
A. Appendix	63
A.1. Full-Sized Images	63
A.2. Results	66
A.2.1. Calculation Bounding Box Area in Percent	66
A.2.2. Additional Tables	66
List of Abbreviations	69
List of Figures	73
List of Tables	75

1 | Introduction

1.1. Motivation

sämtliche
Abkürzungen
mit glocaries
einbinden und
verwalten

Durch die zunehmende Verfügbarkeit hochauflösender Satelliten- und Luftbilddaten ist die weltweite Detektion von Objekten technisch möglich geworden. Gleichzeitig sinken die Kosten für die erforderliche Rechenleistung, sodass der Einsatz von Methoden der Künstlichen Intelligenz (KI) zur vollautomatischen Analyse solcher Daten kontinuierlich erleichtert wird. Damit eröffnen sich neue Möglichkeiten zur effizienten Auswertung großer Mengen an Bilddaten, die sowohl im zivilen als auch im militärischen Kontext von erheblicher Relevanz sind.

Ein bedeutsamer Anwendungsbereich liegt im Monitoring von zerstörten militärischen Anlagen oder Gebäudestrukturen in Konfliktregionen, wie beispielsweise in Darfur [1] oder der Ukraine. Der Einsatz von KI-gestützter Bildanalyse kann hier wertvolle Unterstützung bieten, da eine direkte Dokumentation vor Ort aufgrund der unsicheren Sicherheitslage oftmals nicht möglich oder mit erheblichen Risiken verbunden ist. Luft- und Satellitenbilder eignen sich daher besonders gut zur Informationsgewinnung.

Darüber hinaus könnte die kontinuierliche Überwachung von militärischem Gerät an Ländergrenzen zur Verbesserung der Lageeinschätzung beitragen, indem potenzielle Eskalationen oder bevorstehende Konflikte frühzeitig erkannt werden. Ein entscheidender Vorteil der KI-gestützten Analyse besteht darin, menschliche Analysten zu entlasten, indem Modelle Vorschläge für Detektionen und Klassifizierungen liefern, die anschließend überprüft und validiert werden. Auf diese Weise können Arbeitsabläufe effizienter gestaltet und Entscheidungsträger schneller mit relevanten Informationen versorgt werden.

Auch im zivilen Bereich eröffnen sich vielfältige Anwendungsfelder. So kann die Analyse des Mobilitätsverhaltens der Bevölkerung durch die Erfassung von Fahrzeugen auf großen öffentlichen Parkplätzen unterstützt werden. Mithilfe automatisierter Detektions- und Klassifikationsmethoden

lassen sich Fahrzeuge auf Satelliten- oder Luftbildern identifizieren und zählen, was Rückschlüsse auf die Auslastung von Parkflächen erlaubt. Diese Informationen können beispielsweise genutzt werden, um den Bedarf an zusätzlichen Parkflächen zu bestimmen oder die Effizienz der bestehenden Infrastruktur zu evaluieren. Zudem eröffnet sich die Möglichkeit, das Verkehrsaufkommen auf regionaler oder nationaler Ebene zu quantifizieren, sofern eine kontinuierliche Fahrzeugdetektion implementiert wird. Eine weiterführende Fragestellung ist die Differenzierung zwischen fahrenden und stehenden Fahrzeugen, die zusätzliche Einblicke in Mobilitätsmuster und Verkehrsströme liefern kann.

Eine zentrale Herausforderung bei der Anwendung maschinellen Lernens auf hochauflösende Fernerkundungsbilder liegt in der begrenzten Verfügbarkeit geeigneter Trainingsdaten sowie darin, dass Fahrzeuge aufgrund des Maßstabs relativ klein dargestellt werden. Die präzise Erkennung und Klassifizierung dieser kleinen Objekte erfordert angepasste Deep-Learning-Methoden, die auch feine Strukturen zuverlässig erfassen können.

Vorarbeiten in diesem Themenfeld, wie sie in einer vorherigen Bachelorarbeit durchgeführt wurden, zeigen, dass Deep-Learning-Methoden wie You Only Look Once (YOLO) (v3) kleine Objekte in einfachen Szenarien grundsätzlich erkennen können. Dabei konnte durch die Anpassung von Hyperparametern die Mean Average Precision (mAP) für sehr kleine Objekte verbessert werden. Allerdings wird im Anwendungsfall der Schiffsdetektion deutlich, dass die erzielte mAP kritisch zu bewerten ist und die Eignung von YOLOv3 für eine zuverlässige Detektion von kleinen Objekten nicht uneingeschränkt gegeben ist [2].

Die vorliegende Masterarbeit knüpft an diese Ergebnisse an und überträgt die Fragestellung auf den Bereich der Detektion und Klassifikation von kleinen Fahrzeugen in hochauflösenden multispektralen Luftbilddaten. Dabei wird eine neuere Version von You Only Look Once (YOLO) (v9) eingesetzt, die Verbesserungen in der Netzwerkarchitektur aufweist. Zusätzlich wird untersucht, inwieweit die Integration elektro-optische und multispektrale Luftbildaufnahmen die Präzision bei der Detektion und Klassifikation sehr kleiner Objekte steigern kann. Auf diese Weise erweitert die Arbeit bestehende Forschung sowohl inhaltlich (Fokus auf Fahrzeuge) als auch methodisch (Verwendung moderner YOLO-Architektur und multispektraler Daten) und adressiert die Forschungslücke der Objekterkennung in komplexen Szenarien.

1.2. Structure of the thesis

Diese Thesis bietet einen umfassenden Überblick über die Methoden, Experimente und Ergebnisse der Forschung. Sie beginnt mit einer Einleitung, die die Motivation und die Ziele der Arbeit darlegt, gefolgt von den Grundlagen, in denen der theoretische Hintergrund und relevante Metriken erläutert werden. Der Stand der Forschung fasst die aktuelle Literatur zusammen, bevor die Methodik die verwendeten Datensätze, Werkzeuge und die Vorgehensweise im Detail beschreibt. Anschließend werden die Ergebnisse der Experimente präsentiert und in der Diskussion interpretiert und in den Kontext der bestehenden Forschung eingeordnet. Die Arbeit schließt mit einer Schlussfolgerung und einem Ausblick auf mögliche zukünftige Forschungsrichtungen. Ergänzendes Material wie Bilder und Tabellen findet sich im Anhang.

2 | Fundamentals

2.1. Computer Vision

Computer Vision ist ein Teilbereich der künstlichen Intelligenz, der eng mit der Bildverarbeitung und dem maschinellen Lernen verknüpft ist. Dabei wird die Rohdatenerfassung durch Verfahren erweitert, die digitale Bildverarbeitung, Mustererkennung, maschinelles Lernen und Computergrafik kombinieren. Ziel ist es, Maschinen die menschliche Fähigkeit zu verleihen, aus Bildern Informationen zu extrahieren und zu interpretieren [3]. Zur Informationsgewinnung und zur Simulation menschlicher visueller Wahrnehmung werden Algorithmen sowie optische Sensoren eingesetzt [4]. Es lässt sich zwischen Bildbeschaffung und Bildanalyse unterscheiden. Wichtige Komponenten der Bildanalyse sind unter anderem:

- **Bilderzeugung:** Das Speichern eines Objekts als digitales Bild.
- **Bildverarbeitung:** Verbesserung der Bildqualität zur Erhöhung des Informationsgehalts.
- **Bildsegmentierung:** Trennung des Objekts vom Hintergrund.
- **Bildvermessung:** Ermittlung signifikanter Bildpunkte (sogenannter Features).
- **Bildinterpretation:** Ableitung semantischer Informationen aus den Bilddaten [5].

2.2. Machine and Deep Learning

Maschinelles Lernen beschreibt einen Ansatz in der Informatik, bei dem Algorithmen auf Basis von vorhandenen Daten selbstständig Muster erkennen, um ein definiertes Ziel zu erreichen – etwa die Klassifikation von Informationen. Im Gegensatz zu traditionellen Algorithmen, die feste, vom Menschen vorgegebene Regeln befolgen, entwickelt ein lernfähiger Algorithmus eigene Verarbeitungsstrategien. Die Entscheidungslogik entsteht dabei nicht durch manuelle Programmierung, sondern durch die Analyse von Daten und das Erkennen wiederkehrender Strukturen [6]. Der Begriff „Lernen“ bezieht sich allgemein auf den Prozess, bei dem Wissen oder Fähigkeiten

durch Erfahrung, Versuch und Irrtum, Anleitung oder Beobachtung erworben werden. Übertragen auf den maschinellen Kontext bedeutet dies, dass Maschinen durch wiederholte Analyse und Rückmeldung in der Lage sind, aus Beispielen zu generalisieren und neue, unbekannte Daten zu verarbeiten. Maschinelles Lernen kann daher als rechnergestützte Nachbildung menschlicher Lernprozesse verstanden werden, bei der Algorithmen in der Lage sind, abstrahierte Regeln aus Rohdaten zu entwickeln und auf neue Situationen anzuwenden [7], [8]. Diese Fähigkeit eröffnet neue Möglichkeiten bei der Lösung komplexer Probleme, die sich mit konventionellen, regelbasierten Methoden nur schwer oder gar nicht bearbeiten lassen [9].

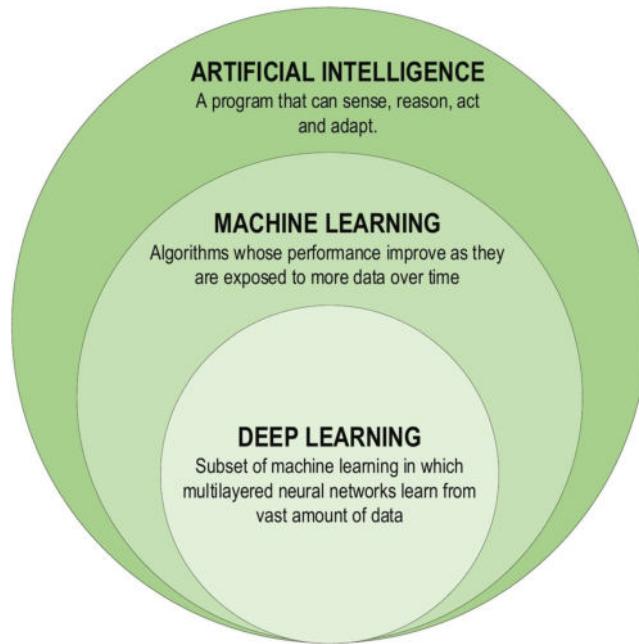


Figure 2.1.: Overview over Artificial Intelligence, Machine Learning and Deep Learning. [10]

Deep Learning (DL) ist ein Teilgebiet des maschinellen Lernens (s. Fig. 2.1), dessen Funktionsweise von den neuronalen Prozessen im menschlichen Gehirn inspiriert ist. Es simuliert den Informationsverarbeitungsprozess, wie er in zentralen sensorischen Regionen des Gehirns abläuft. Im Gegensatz zu klassischen regelbasierten Verfahren arbeitet DL nicht mit strikt von Menschen entworfenen Regeln, sondern nutzt große Datenmengen, um Muster zu erkennen und Eingaben auf spezifische Eigenschaften hin zu analysieren. Charakteristisch ist dabei die Verwendung zahlreicher Schichten Artificial Neural Networks (ANNs), in denen jede Schicht die zugelieferten Informationen in einer einzigartigen Form interpretiert. Während herkömmliche Techniken des maschinellen Lernens meist auf eine Trennung von Pre-Processing, Feature Extraction, Feature Selection, Learning und Classification angewiesen sind, kann DL diese Schritte teilweise automatisieren und in einem einzigen End-to-End-Prozess zusammenführen. Dadurch werden sowohl das Lernen von Feature Sets für mehrere Aufgaben als auch Klassifikation in einem Schritt ermöglicht [10].

Zu den zentralen Vorteilen von DL zählen seine universelle Anwendbarkeit in verschiedenen Bereichen, die Robustheit durch die automatisierte Extraktion optimierter Merkmale anstelle handgefertigter Features, die Fähigkeit zur Verallgemeinerung über verschiedene Datentypen hinweg – beispielsweise durch Transferlernen, was besonders in Szenarien mit geringen Datenmengen relevant ist – sowie eine hohe Skalierbarkeit, da Netzwerke unkompliziert durch zusätzliche Knoten erweitert werden können. Grundsätzlich lassen sich DL-Ansätze in drei Kategorien unterteilen: überwachtes, semi-überwachtes und unüberwachtes Lernen[10].

Beim überwachten Lernen (Supervised Learning) erhält ein Algorithmus eine Sammlung von Eingaben und den zugehörigen Ausgaben. Ein intelligenter Agent schätzt eine Ausgabe $\hat{y}_t = f(x_t)$ für eine Eingabe x_t und berechnet den Verlust $tz(\hat{y}_t, y_t)$ als Abweichung zur bekannten Zielausgabe y_t . Durch wiederholte Anpassung der Netzwerkparameter wird das Modell sukzessive optimiert, bis die Schätzungen mit den tatsächlichen Ausgaben übereinstimmen. Typische Architekturen in diesem Bereich sind Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) und Deep Neural Networks (DNNs). Im semi-überwachten Lernen werden Datensätze verwendet, die nur teilweise gelabelt sind. Dies reduziert die benötigte Menge an Trainingsdaten mit Annotationen, birgt jedoch die Gefahr, dass irrelevante Eingangsmerkmale zu falschen Klassifikationen führen können. Ein klassisches Anwendungsbeispiel ist die Textklassifikation. Im unüberwachten Lernen schließlich werden gänzlich ungelabelte Daten genutzt, um signifikante Merkmale oder innere Strukturen in den Daten zu identifizieren. Hierbei kann das Modell Cluster oder Zusammenhänge entdecken, jedoch liefert dieser Ansatz keine präzisen Informationen über die Sortierung der Daten und ist zudem rechenintensiv [10].

Innerhalb der Vielzahl an Netzarchitekturen haben sich CNNs als die wohl bekannteste und am häufigsten genutzte Variante etabliert. Inspiriert von biologischen Neuronenstrukturen identifizierte [9] drei zentrale Vorteile von CNNs: gleichwertige Darstellungen, spärliche Interaktionen sowie eine effiziente Parameternutzung durch Weight Sharing. Eine Eingabe x wird in den Schichten eines CNN durch drei Dimensionen repräsentiert: Höhe, Breite und Tiefe, wobei die Tiefe der Anzahl der Bildkanäle entspricht. In den Faltungsschichten (Convolutional Layers) werden Filter bzw. Kernel k mit Dimensionen $n \times n \times q$ angewandt, die lokale Verbindungen und Feature Maps h^k erzeugen [10]. Die Berechnung erfolgt dabei über das Punktprodukt zwischen Eingabe und Gewichtsmatrix:

$$h^k = f(W^k * x + b^k). \quad (2.1)$$

Darauf folgt ein Downsampling der Feature Maps in sogenannten Pooling-Schichten, das die Anzahl der Netzparameter reduziert, den Trainingsprozess beschleunigt und Überanpassung (Overfitting) entgegenwirkt. Pooling-Operationen wie Max- oder Average-Pooling aggregieren lokale Regionen einer Feature Map, bevor die abstrahierten Merkmale in Fully Connected Layers (FCLs) zusammengeführt werden. Die finale Klassifikation wird in der Ausgabeschicht häufig mit einem

Klassifikator wie der Support Vector Machine (SVM) realisiert. CNNs zeichnen sich nicht nur die bereits genannte bessere Generalisierung, der Verhinderung von Overfitting durch Weight Sharing sowie der flexible Nutzung extrahierter Merkmale sondern auch durch einfache Skalierbarkeit aus. Zu den zentralen Komponenten einer CNN-Architektur gehören Convolutional Layers mit trainierbaren Filtern, Pooling Layers zur Dimensionsreduktion, Aktivierungsfunktionen wie Rectified Linear Unit (ReLU) zur Einführung von Nichtlinearität, FCLs für die Merkmalsaggregation sowie Loss-Funktionen, die den Unterschied zwischen vorhergesagten und tatsächlichen Werten messen und für die Parameteroptimierung genutzt werden [10].

2.2.1. Backpropagation

Der Lernprozess in Deep-Learning-Netzen basiert auf dem Prinzip der Backpropagation [9]. Dabei wird zunächst in einer Vorwärtsphase (Forward Propagation) die Eingabe x durch das Netzwerk propagiert, sodass am Ende die Ausgabe y resultiert. Anschließend werden die Fehler durch Rückwärtspropagation (Backpropagation) schrittweise durch das Netz zurückgeführt, um Gradienten zu berechnen, die zur Aktualisierung der Gewichte genutzt werden. Dieses Verfahren ermöglicht eine effiziente Optimierung tiefer Netzwerke, stößt jedoch bei sehr tiefen Architekturen auf zwei zentrale Probleme: das Vanishing-Gradient-Problem und das Exploding-Gradient-Problem [10].

2.2.2. Vanishing-Gradient-Problem

Das Vanishing-Gradient-Problem tritt auf, wenn die Gradienten in tiefen Netzen während der Backpropagation exponentiell kleiner werden und somit keine signifikanten Gewichtsaktualisierungen mehr möglich sind. Dies kann im Extremfall dazu führen, dass ein Netz das Lernen vollständig einstellt. Typische Ursachen sind die Verwendung von Aktivierungsfunktionen mit kleinen Ableitungen (wie Sigmoid oder Tanh) oder sehr tiefe Netze mit vielen Schichten. Mögliche Gegenmaßnahmen sind die Verwendung von Aktivierungsfunktionen wie ReLU, Batch-Normalisierung zur Stabilisierung der Eingaben oder schnellere Hardware, die ein Training mit komplexeren Architekturen ermöglicht [10].

2.2.3. Exploding-Gradient-Problem

Das Exploding-Gradient-Problem stellt den gegenteiligen Fall dar: hier wachsen die Gradienten während der Rückpropagation exponentiell an, was zu extrem hohen Gewichtsaktualisierungen, Instabilität und im schlimmsten Fall zu Not A Number Values (NaN-Values) (Werte, die undefiniert oder nicht darstellbar sind) führt. Gelöst werden kann dies etwa durch Regularisierungstechniken, angepasste Netzwerkarchitekturen oder Gradient Clipping [10].

2.2.4. Overfitting

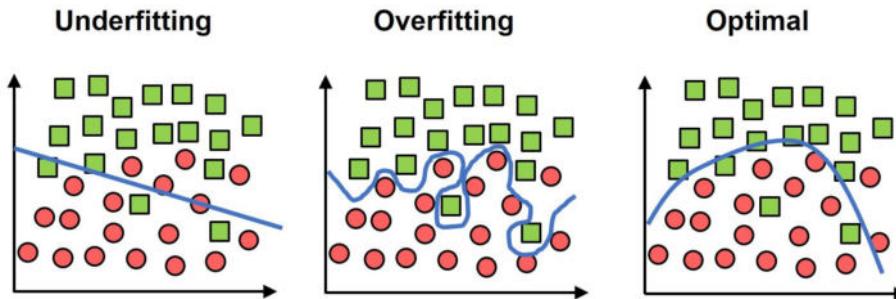


Figure 2.2.: Underfitting, Overfitting and Optimal at Deep Learning Trainings. [11]

Ein zentrales Problem beim Training tiefer neuronaler Netze stellt das Overfitting dar (vgl. Abb. 2.2). In diesem Fall passt sich das Modell zu stark an die Trainingsdaten an: Es erzielt dort eine hohe Genauigkeit, während die Leistung auf unabhängigen Testdaten deutlich abnimmt. Das Gegenphänomen ist das Underfitting, bei dem die zugrunde liegende Datenstruktur nicht hinreichend erfasst wird und das Modell folglich sowohl im Trainings- als auch im Testdatensatz unzureichende Ergebnisse liefert. Ziel ist daher die Entwicklung eines Modells, das die relevanten Muster der Trainingsdaten adäquat erlernt und gleichzeitig eine hohe Generalisierungsfähigkeit gegenüber unbekannten Daten aufweist [11].

Die hohe Anzahl an Parametern in tiefen Netzen erhöht die Wahrscheinlichkeit, dass ein Modell die Trainingsdaten übermäßig stark „auswendig lernt“ und dadurch seine Fähigkeit verliert, auf neuen Testdaten verlässliche Vorhersagen zu treffen. Overfitting kann durch verschiedene Regularisierungstechniken verringert werden, darunter Gewichtsverfall (Weight Decay), Batch-Normalisierung und Dropout. Eine weitere Möglichkeit stellt die Datenaugmentation dar, bei der der Trainingsdatensatz künstlich vergrößert wird, um die Generalisierungsfähigkeit zu verbessern. Ebenso kann Data Corruption – die gezielte Einbeziehung verrauschter oder manipulierter Daten – die Robustheit des Modells erhöhen. Schließlich existieren Verfahren, die übermäßig zuversichtliche Vorhersagen gezielt bestrafen und so ebenfalls als Regularisierung wirken [10].

Zusammenfassend bietet Deep Learning einen leistungsstarken Ansatz zur automatisierten Merkmalsextraktion und Klassifikation, der sich durch Flexibilität, Robustheit und hohe Skalierbarkeit auszeichnet. Gleichzeitig stellen Herausforderungen wie das Vanishing-Gradient-Problem, Exploding Gradients und Overfitting zentrale Forschungsfelder dar, für die kontinuierlich neue Lösungen entwickelt werden [10].

2.3. One Stage and Two Stage Detectors

Die Lokalisierung und Erkennung von Objekten in Bildern kann grundsätzlich durch zwei Ansätze realisiert werden: One-Stage-Detektoren und Two-Stage-Detektoren [12].

Two-Stage-Detektoren, wie beispielsweise Region-Based Convolutional Neural Network (R-CNN) [13] oder Mask R-CNN [14], bestehen aus zwei aufeinanderfolgenden Verarbeitungsschritten. In der ersten Stufe wird ein Region Proposal Network (RPN) eingesetzt, um potenziell relevante Bildbereiche zu identifizieren. Diese Vorschläge werden anschließend in der zweiten Stufe für die Objektklassifizierung und die Bounding-Box-Regression weiterverarbeitet. Two-Stage-Ansätze erreichen in der Regel sehr hohe Genauigkeitswerte, sind jedoch aufwendiger in der Berechnung und zeichnen sich durch eine geringere Verarbeitungsgeschwindigkeit aus [12].

Im Gegensatz dazu behandeln One-Stage-Detektoren, wie You Only Look Once (YOLO) [15] oder der Single Shot Multibox Detector (SSD) [16], die Objekterkennung als direktes Regressionsproblem. Dabei wird das Eingabebild in einem einzigen Schritt verarbeitet, wobei sowohl die Klassenzugehörigkeiten als auch die Koordinaten der Bounding Boxes vorhergesagt werden. Dieser Ansatz ermöglicht eine deutlich höhere Verarbeitungsgeschwindigkeit, erreicht jedoch in der Regel geringere Genauigkeitsraten im Vergleich zu Two-Stage-Detektoren [12].

2.4. Milestones and Historical Development of YOLO Architectures

You Only Look Once (YOLO) ist ein One-Stage-Detektor, der erstmals in der Version YOLOv1 von Redmon et al. (2016) veröffentlicht wurde [15]. Dieser Ansatz stellte einen neuartigen Weg zur Objekterkennung dar, da er Genauigkeit und Geschwindigkeit durch die Verarbeitung von Bildern in einer einstufigen Netzwerkarchitektur kombinierte. YOLOv1 bildete somit den Grundstein für Echtzeitanwendungen in der Bildverarbeitung und wurde zum Standard für nachfolgende Entwicklungen [17].

Auf YOLOv1 aufbauend, verbesserte YOLOv2, bzw. YOLO9000 [18], [19], die Auflösung, mit der das Modell arbeitete, und erweiterte die Objekterkennung auf mehr als 9000 Kategorien. Mit YOLOv3 wurden Multi-Scale-Vorhersagen und eine tiefere Netzwerkarchitektur eingeführt, um insbesondere die Erkennung kleiner Objekte zu optimieren [20].

YOLOv4 brachte verschiedene Hauptvarianten hervor. Die Standardversion YOLOv4-CSP integrierte Cross-Stage-Partial-Networks zur Leistungsverbesserung, während YOLOv4x-mish die Mish-Aktivierungsfunktion nutzte, um die Genauigkeit bei gleichbleibender Effizienz zu steigern [21], [22], [23].

Im Jahr 2020 entwickelte Ultralytics YOLOv5, das durch verbesserte Benutzerfreundlichkeit und Leistung überzeugte. Es wurden fünf Hauptvarianten eingeführt, die unterschiedliche Anwendungsbereiche abdecken – von ressourcenschonender Geschwindigkeit bis hin zu maximaler Genauigkeit auf leistungsstarker Hardware [17], [24]. Auf dieser Basis bauten die nachfolgenden Versionen YOLOv5 bis YOLOv11 auf, wobei der Fokus auf besserer Skalierbarkeit, reduzierten Rechenanforderungen und optimierten Echtzeit-Leistungsmetriken lag.

YOLOv6, vorgestellt 2022 von einem Team eines chinesischen E-Commerce-Plattformbetreibers [25], verfügte über eine neuartige Backbone- und Neck-Architektur. Zudem wurden fortschrittliche Trainingstechniken wie Anchor-Aided Training (AAT) und Self-Distillation implementiert [17], [25]. YOLOv7 [26], [27] führte weitere Innovationen ein, darunter trainable bag of freebies, also Optimierungen zur Steigerung der Genauigkeit ohne zusätzliche Inferenzkosten, sowie dynamische Label-Zuweisungen [17], [26], [27].

YOLOv8, veröffentlicht 2023 von Ultralytics [28], zeichnete sich durch eine effizientere Architektur, verbesserte Trainingstechniken und erweiterte Unterstützung für große Datensätze aus [17], [28]. YOLOv9 [29] integrierte Programmable Gradient Information (PGI) für eine leistungsfähigere Nutzung tieferer Netzwerke und führte die leichtgewichtige Netzwerkarchitektur Generalized Efficient Layer Aggregation Network (GELAN) ein, die auf Gradientenpfadplanung basiert und in Kombination mit PGI besonders gute Ergebnisse auf ressourcenschonenden Modellen erzielt [17], [29].

YOLOv10, entwickelt 2023 von chinesischen Wissenschaftlern [30], präsentierte einen neuartigen Ansatz für die Echtzeit-Objekterkennung. Durch den Verzicht auf Non-Maximum-Supression (NMS) und die Optimierung zentraler Modellkomponenten wurden Einschränkungen früherer YOLO-Versionen überwunden, was zu erheblichen Effizienz- und Leistungssteigerungen führte [30]. YOLOv11 verbesserte die Backbone- und Neck-Architektur weiter, wodurch die Merkmalsextraktion optimiert und ein ausgewogenes Verhältnis zwischen Präzision und Recheneffizienz erreicht wurde [17], [31].

Schließlich nutzt YOLOv12 einen aufmerksamkeitszentrierten Ansatz und implementiert Area Attention (AA2)-Modul (AA2) sowie Residual Efficient Layer Aggregation Networks (R-ELAN), um die Merkmalsverarbeitung weiter zu verbessern [17], [32].

2.5. Evaluation Metrics

2.5.1. K Fold Cross Validation

Die K-fold Cross Validation (KCV) ist eine der am häufigsten verwendeten Methoden zur Modellauswahl und zur Fehlerabschätzung bei Klassifikationsaufgaben. Bei dieser Technik wird der Datensatz in k gleich große Teilmengen aufgeteilt. Iterativ werden $k - 1$ dieser Teilmengen zum Trainieren des Modells verwendet, während die verbleibende Teilmenge zur Bewertung der Modelleistung dienen [33]. In der Praxis wird k häufig auf 5 oder 10 gesetzt, da Schätzungen bei diesen Werten weder durch eine hohe Verzerrung noch durch eine sehr hohe Varianz gekennzeichnet sind [34].

Darüber hinaus ermöglicht die K-Fold Cross Validation eine strukturierte Aufteilung des Datensatzes in Trainings-, Validierungs- und Testmengen. Dabei ist sicherzustellen, dass die Verteilung der Klassenobjekte sowie die Anzahl der Bilder in den einzelnen Folds weitgehend homogen ist. Überschneidungen von Bildern zwischen verschiedenen Folds müssen vermieden werden, da das Modell sonst Gefahr läuft, zu overfitten, wenn identische Bilder sowohl im Trainings-, Validierungs- als auch im Testdatensatz enthalten sind.

2.5.2. Intersection over Union (IoU)

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Figure 2.3.: Graphical Visualiziation for IoU. [35]

Die Intersection-over-Union (IoU) (für eine graphische Erklärung s. Abb.2.3), auch als Jaccard-Index bekannt, ist die am häufigsten verwendete Metrik zum Vergleich der Ähnlichkeit zweier beliebiger Formen. Sie kodiert die Formmerkmale der zu vergleichenden Objekte, beispielsweise Höhe, Breite und Position zweier Bounding Box (BB), in eine Bereichseigenschaft und berechnet daraus ein normalisiertes Maß, das sich auf die verglichenen Flächen oder Volumina bezieht (siehe Abb. 2.3). Diese Eigenschaft macht die IoU unabhängig von der Größe des betrachteten Objekts [36].

Aufgrund dieser Unabhängigkeit basiert eine Vielzahl von Leistungsmaßen in den Bereichen Segmentierung [37], [38], [39], [40], Objekterkennung [40], [41] und Objektverfolgung [42], [43] auf der IoU. Dabei entspricht eine der beiden BB der Ground Truth (GT), also dem korrekt gelabelten und lokalisierten Objekt, während die zweite BB die vom Modell vorhergesagte Lokalisierung des Objektes darstellt. Die auf dieser Grundlage berechnete Übereinstimmung (IoU), bei der die Überlappungsfläche der beiden Bounding Boxen durch die Gesamtfläche, die beide abdecken, geteilt wird, ermöglicht eine präzise Bewertung der Genauigkeit von Deep-Learning-Modellen.

Definition 2.5.1 (Intersection over Union)

Let B_t be the set of image pixels covered by a Ground Truth Bounding Box for a class k , and B_p be the set of image pixels covered by a predicted Bounding Box for the same class. Then the IoU is calculated as follows:

$$\text{IoU}_k = \begin{cases} \frac{|B_t \cap B_p|}{|B_t \cup B_p|}, & \text{if } |B_t \cup B_p| > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2.2)$$

2.5.3. Mean Average Precision (mAP)

Die Intersection-over-Union steht in engem Zusammenhang mit der Mean Average Precision (mAP), ist jedoch nicht identisch. Während die IoU die Genauigkeit eines einzelnen vorhergesagten BB misst, stellt die mAP eine umfassendere Metrik dar, die die Leistung eines Modells über alle Objekte und Klassen eines Datensatzes bewertet [44].

Zur Berechnung der mAP werden die Konzepte Precision und Recall herangezogen. Precision beschreibt den Anteil der vom Modell als positiv klassifizierten Erkennungen, die tatsächlich korrekt sind, während Recall den Anteil der tatsächlich vorhandenen Objekte angibt, die korrekt erkannt wurden. Auf Basis dieser Werte kann eine Precision-Recall-Curve (PRC) (s. Def. 2.5.2) erstellt werden, wobei Precision auf der Y-Achse und Recall auf der X-Achse abgetragen wird [9]. Objektdetektoren geben für jede Bounding Box eine Konfidenz an, die die Sicherheit des Modells hinsichtlich der Vorhersage widerspiegelt. Vorhersagen, deren Konfidenz unter einer festgelegten Schwelle liegen, können verworfen werden.

Definition 2.5.2 (Interpolated Precision-Recall Curve)

Let $P(r)$ be the value of precision at recall value r . The interpolation of the precision-recall curve is then given by P_{int} .

$$P_{int}(r) = \max_{r \leq y} P(y) \quad (2.3)$$

Die PRC kann für alle möglichen Konfidenzschwellen erstellt werden, wodurch sich die durchschnittliche Präzision als Fläche unter der Kurve berechnen lässt. Die Modellbewertung erfolgt anschließend anhand der Mean Average Precision (mAP) (s. Form. 2.5), die als Mittelwert der Average Precision (AP) (s. Form. 2.4) über alle Klassen ermittelt wird (s. Def. 2.5.3)[45]. Dabei können unterschiedliche Genauigkeitsanforderungen an die Übereinstimmung der Bounding Boxes gestellt werden, beispielsweise durch IoU-Schwellenwerte von 0,5 oder einem Bereich von 0,5 bis 0,95. Letzteres entspricht dem Mittelwert von mAP@0.5, mAP@0.55 bis mAP@0.95. Diese strengere Metrik erfordert eine größere Überlappung zwischen GT und Vorhersage und ist besonders geeignet, wenn die vorhergesagten Positionen der Bounding Boxes sehr präzise sein müssen [45].

Definition 2.5.3 (Mean Average Precision)

Let C be the set of classes with $c \in C$, and $P_{int}(r)$ be the interpolated precision-recall curve. The Average Precision (AP) and the mean Average Precision (mAP) are then defined as follows:

$$AP_c = \int_0^1 P_{int}(r) dr \quad (2.4)$$

$$mAP = \frac{1}{|C|} \sum_c AP_c \quad (2.5)$$

3 | State of research

Deep Learning kann zusammen mit Fernerkundungsdaten für eine Vielzahl von Anwendungszwecken eingesetzt werden, beispielsweise für Landnutzungs- und Landbedeckungsklassifikation, Szeneklassifikation oder Objekterkennung [46]. Darüber hinaus finden sich Einsatzgebiete in der Hyperspektralanalyse, in der Interpretation von SAR-Bildern, in der Interpretation hochaufgelöster Satellitenbilder, in der multimodalen Datenfusion sowie in der 3D-Rekonstruktion. Besonders hervorzuheben sind hierbei die Hyperspektralanalyse und die Interpretation hochaufgelöster Satellitenbilder. In der Hyperspektralanalyse wurde beispielsweise ein Stacked Auto Encoder genutzt, um hierarchische Merkmale in der spektralen Domäne zu extrahieren [47]. Ebenso konnten überwachte Convolutional Neural Networks erfolgreich für die Klassifizierung von Anbauarten eingesetzt werden [48].

Bevor auf die methodischen Entwicklungen eingegangen wird, seien die für diese Arbeit relevanten Begriffe kurz definiert. Zu *High Resolution* ist zu bemerken, dass bestimmte multispektrale Satelliten-systeme wie Sentinel-1, Sentinel-2 oder Landsat vergleichsweise geringe räumliche Auflösungen (typischerweise im Bereich von 10–30 m) aufweisen [49]. Demgegenüber besitzen luftgestützte Aufnahmen – wie sie etwa im VEDAI-Datensatz bereitgestellt werden – eine sehr viel höhere Auflösung; dort entsprechen die Pixel beispielsweise einer Fläche von 12.5×12.5 cm und sind somit deutlich besser zur Erkennung kleiner Objekte geeignet [50]. Unter *Multispectral Imagery* wird die spektrale Auflösung eines Sensors verstanden, also die Anzahl der spektralen Bänder und der Bereich des elektromagnetischen Spektrums, den ein Sensor erfasst [51]. Sensoren können mehrere Bänder simultan aufzeichnen; ein Echtfarbbild besteht typischerweise aus den Bändern R (Rot), G (Grün) und B (Blau), das durch weitere Bänder wie Near-Infrared (NIR) oder andere Infrarotbänder erweitert werden kann. Aus solchen Kanälen lassen sich zusätzlich Indizes (z. B. Normalized Difference Vegetation Index (NDVI)) für die Analyse ableiten [3]. Eine eindeutige, universelle Definition für *Small Object Detection* existiert nicht, da die Klassifikation von „klein“ von der Auflösung des Bildes abhängt; gängige Operationalisierungen messen die relative Größe eines Objekts bezogen auf das Bild und beschreiben kleine Objekte beispielsweise durch mittlere relative Überlappungswerte (Intersection over Image Area) im Bereich von etwa 0.08% bis 0.58% [52].

Für die Interpretation hochauflöster Satellitenbilder spielt Deep Learning eine zentrale Rolle. Ein wichtiges Beispiel ist die Szenenklassifikation, bei der jedem Bild automatisch ein semantisches Label zugeordnet wird. Weit verbreitete Anwendungsbereiche sind Objekterkennung [53], [54], Change Detection [55], Urban Planning und Land Resource Management. Typischerweise lassen sich diese Aufgaben in die Schritte Feature Extraction und Classification unterteilen. Da Deep Learning mit zunehmender Tiefe abstraktere und unterschiedungskräftigere semantische Merkmale erlernen kann, wird im Vergleich zu klassischen Ansätzen eine deutlich bessere Klassifizierungsleistung erzielt; aus diesem Grund gilt Deep Learning als Stand der Technik für Szenenklassifikationsprobleme bei hochauflösten Satellitenbildern [46].

Die Objekterkennung stellt in diesem Kontext eine besondere Herausforderung dar, da ein oder mehrere spezifizierte Ground-Objekte wie Gebäude, Fahrzeuge oder Flugzeuge nicht nur lokalisiert, sondern auch einer Kategorie zugeordnet werden müssen. Deep Convolutional Neural Networks werden hierfür bevorzugt, da sie in der Lage sind, hochgradig abstrakte und semantisch aussagekräftige Merkmalsrepräsentationen zu lernen [46]. Ein besonders relevantes Anwendungsfeld ist die Verkehrsplanung: Satellitenbilder können genutzt werden, um Informationen zur Fahrzeugverteilung in einer gesamten Region zu einem bestimmten Zeitpunkt zu gewinnen. Momentaufnahmen des gesamten Straßennetzes liefern dabei wertvolle Einblicke in die Verteilung der Fahrzeuge und bieten Informationen für Bereiche, die von herkömmlichen Zählsystemen – meist induktive Schleifendetektoren, die in die Fahrbahn eingebettet sind – nicht erfasst werden. Seit der Einführung ziviler optischer Hochauflösungssatelliten wie Quickbird und Ikonos sind Bilder mit einer Auflösung von etwa einem Meter verfügbar. Damit ist es möglich geworden, relativ kleine Objekte wie Fahrzeuge auf hochauflösten Satellitenbildern sicher zu detektieren und korrekt zu klassifizieren [56]. Moderne Satellitenmissionen wie die von Planet (50 cm/Pixel) [57] oder die Airbus Pléiades Neo Satelliten (30 cm/Pixel) [58] bieten mittlerweile noch höhere Auflösungen, was die Möglichkeiten für die präzise Erkennung kleiner Fahrzeuge erheblich erweitert.

Vor dem Durchbruch tiefer neuronaler Netze dominierten klassische Verfahren wie Support Vector Machine (SVM) oder Stacked Auto Encoder (SAE). Bei hyperspektralen Fernerkundungsbildern zeigte sich jedoch, dass SAE gegenüber SVM keine offensichtliche Leistungssteigerung erzielte; dies wird häufig damit erklärt, dass SAE durch die große Anzahl an Parametern deutlich mehr Trainingsdaten benötigt und der zu erwartende Vorteil durch den Aufwand der Datensammlung relativiert wird [59].

Mit dem Aufkommen von Deep Learning etablierten sich CNNs sowohl für spezifische als auch für generische Objekterkennungsaufgaben. Zur Verbesserung der Trainingsverfahren wurde ein weakly supervised learning Framework vorgeschlagen, das ein vortrainiertes CNN und ein negatives Bootstrap-Schema nutzt, um eine schnelle Konvergenz des Detektors zu erreichen [60]. Weitere Arbeiten kombinierten tiefen, aus vortrainierten CNNs extrahierte Umgebungsmerkmale mit

klassischen lokalen Merkmalen wie Histogrammen orientierter Gradienten [61] zur zuverlässigen Detektion von Öltanks [62]. Zwei-Stufen-Ansätze wurden ebenfalls erfolgreich angewendet: So wurde GoogLeNet mit unterschiedlichen Fine-Tuning-Parametern trainiert und anschließend im Sliding-Window-Verfahren zur Objekterkennung eingesetzt [63]. Das Problem variierender Objektorientierungen wurde durch die Nutzung kombinierter CNN-Layer-Merkmale adressiert, die orientierungsrobuste Erkennungen in einem groben Lokalisierungsrahmen ermöglichen [64].

Parallel dazu gewann die Nutzung multispektraler und hyperspektraler Daten an Bedeutung. Multispektrale Kanäle wie RGB, NIR, Middle-Infrared (MIR) oder Far-Infrared (FIR) werden beispielsweise im Kontext des autonomen Fahrens genutzt, da insbesondere Infrarotkanäle auch bei schlechten Witterungsbedingungen zuverlässig Detektionen erlauben [65]. Im Bereich der Fahrzeugdetektion wird das panchromatische Band für die Fahrzeugklassifikation eingesetzt, während multispektrale Informationen zur Maskierung von Vegetation und Schatten genutzt werden [56].

Für die Detektion von Fahrzeugen in hochauflösten Satellitenbildern wurden hybride CNN-Modelle entwickelt, welche Feature-Maps aus Convolutional- und Pooling-Layern in Blöcke variabler Größe unterteilen, um mehrskalige Merkmale zu extrahieren [66]. Ein alternativer Ansatz nutzt graphbasierte Superpixel-Segmentierung, um Bildausschnitte zu extrahieren, die anschließend von einem CNN daraufhin klassifiziert werden, ob ein Fahrzeug vorhanden ist [67].

Vorangegangene Untersuchungen zeigen, dass bei YOLOv3 die Optimierung von Hyperparametern – insbesondere der Rastergröße, Trainingsdauer und Lernrate – die Mean Average Precision (mAP) bei der Detektion kleiner Objekte verbessern kann. So führte beim Airbus Ship Detection Datensatz [68] die Anpassung der Rastergröße sowie der Einsatz eines Learning Rate Schedulers zu besseren Ergebnissen in den Klassen kleinerer Objekte (0-5 % bzw. 5-20 % der Bildfläche). Für Datensätze mit größeren Bildern und einer Vielzahl von Objekten, wie z. B. DOTA, konnte hingegen keine signifikante Verbesserung der mAP erzielt werden. Diese Ergebnisse deuten darauf hin, dass YOLOv3 unter bestimmten Bedingungen für die Detektion kleiner Objekte effektiv eingesetzt werden kann, für komplexere Multi-Klassen-Szenarien jedoch möglicherweise nicht optimal ist [2].

Die Befunde ergänzen die bisherige Literatur und unterstreichen, dass neben der Architektur auch die Wahl der Hyperparameter und die Charakteristika des Datensatzes entscheidend für die Detektionsleistung sind. Sie unterstützen die Notwendigkeit, die Effekte zusätzlicher spektraler Kanäle sowie unterschiedlicher Bounding Box-Typen (axis-aligned vs. oriented) im Kontext hochauflöster multispektraler Satellitenbilder gezielt zu untersuchen.

Trotz dieser Fortschritte bleiben Forschungsfragen offen. Insbesondere ist zu klären, inwieweit moderne One-Stage-Detektoren wie YOLO durch die Eingabe zusätzlicher spektraler Kanäle (z.,B. Infrared (IR), NDVI) gegenüber reinen RGB-Eingaben profitieren. Theoretisch sollte sich die

3. STATE OF RESEARCH

Leistung durch zusätzliche Kanäle verbessern, da Vegetation (z. B. durch Hinzufügen des NDVI) besser separierbar ist und IR zusätzliche informationsreiche Signale liefert. Ebenfalls zu untersuchen sind der Einfluss von axis-aligned versus oriented bounding boxes (ABB vs. OBB) auf die Erkennungsleistung sowie die Frage, welche einzelnen Kanäle die Leistung verbessern oder verschlechtern.

gefühlt ein
bisschen zu
viel wiederhol-
ung

In der vorliegenden Arbeit wird eine moderne YOLO-Architektur (Version 9) eingesetzt, wodurch sich möglicherweise gegenüber älteren Versionen wie YOLOv3 verbesserte Detektionsergebnisse und Trainingsmöglichkeiten ergeben. Dies erlaubt, die Effekte zusätzlicher spektraler Kanäle und optimierter Hyperparameter im Kontext hochauflöster multispektraler Luftbildern gezielt zu untersuchen.

4 | Methodology

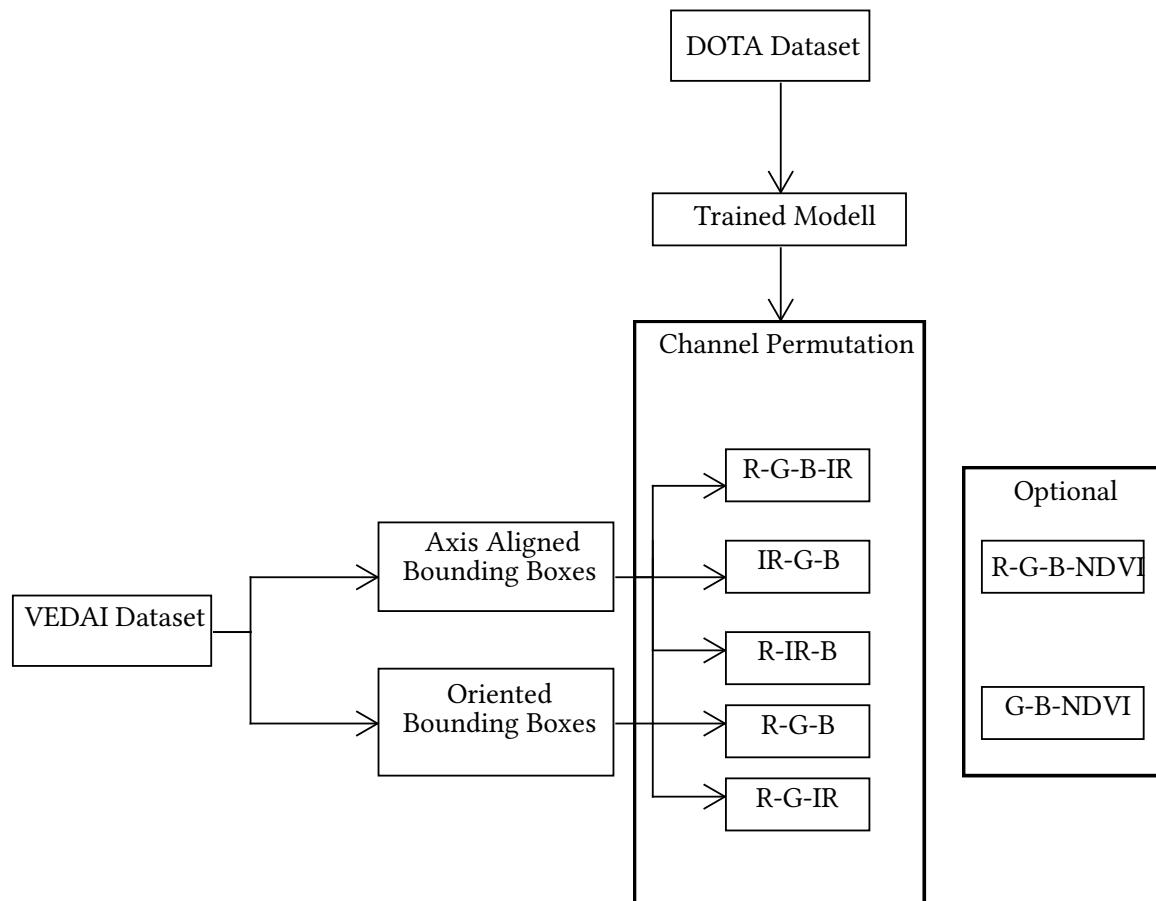


Figure 4.1.: Flowchart for workflow. The VEDAI dataset is first used to investigate the differences between ABB and OBB, with the channel permutation experiments based on the respective better bounding box format. In the next step, a self-trained model based on the DOTA dataset is used as a pre-trained model. The permutation experiments based on this model comprise five different channel combinations and, optionally, the inclusion of NDVI in combination with several true colour channels (Own representation).

Die methodische Vorgehensweise dieser Arbeit ist im zugehörigen Flowchart (s. Fig. 4.1) dargestellt. Das VEDAI-Dataset wird zuerst zur Untersuchung der Unterschiede zwischen ABB und OBB verwendet, wobei die Experimente zu den Kanalpermutationen auf dem jeweils besseren Bounding-

Box-Format basieren. Im nächsten Schritt kommt als vortrainiertes Modell ein selbst trainiertes Modell auf Basis des DOTA-Datensatzes zum Einsatz. Die darauf aufbauenden Permutationsexperimente umfassen fünf verschiedene Kanalkombinationen sowie optional die Einbeziehung von NDVI in Kombination mit mehreren Echtfarbkanälen.

Im folgenden Abschnitt wird die Methodik dieser Arbeit beschrieben. Zunächst wird die Datengrundlage anhand der verwendeten Datensätze erläutert. Darauf aufbauend werden die eingesetzten YOLO-Versionen vorgestellt. Im Anschluss folgt die Implementierung, bestehend aus den verwendeten Python-Packages sowie einer Beschreibung des Programmcodes, welcher auf Github [69] zur Verfügung gestellt wird. Danach werden die zentralen Herausforderungen im Preprocessing dargestellt. Anschließend werden die Ergebnisse der 6-Fold-Cross-Validation präsentiert, wobei eigene Folds erstellt wurden, die für alle Experimente mit ABB, OBB sowie den Permutationen (s. Tab. 4.1) genutzt wurden. Abschließend wird die Hardware beschrieben, auf der die Experimente durchgeführt wurden, nämlich der Hyper-Performance-Cluster PALMA, sowie die dort ausgeführten Bash-Skripte mit den für YOLO verwendeten Hyperparametern.

Permutation	Red (R)	Green (G)	Blue (B)	Infrared (IR)	NDVI
RGBIR	x	x	x	x	
IRGB		x	x	x	
RIRB	x		x	x	
RGB	x	x	x		
RGIR	x	x		x	
RGBNDVI	x	x	x		x
GBNDVI		x	x		x

Table 4.1.: Channel permutations represented by included channels (x = channel present).

4.1. Database

Die Datenbasis umfasst 2 Datensätze, die annotierte Luft- und Satellitenbildern enthalten. Das DOTA Dataset wurde genutzt, um ein vortrainiertes Modell zu erzeugen, das als Ausgangspunkt für das Training mit Kanalpermutationen dient, während das VEDAI Dataset mit der in Sec. 4.6 beschriebenen Verteilung für die Permutationsexperimente genutzt wurde.

4.1.1. Vehicle Detection on Aerial Images (VEDAI) Dataset

Das VEDAI Dataset [70] aus dem Jahr 2015 bietet sich als Datengrundlage an, da es hochauflösende Luftbilder enthält, die speziell für die Fahrzeugerkennung geeignet sind [50]. Es umfasst annotierte

Daten, die Fahrzeuge in unterschiedlichen Szenarien, Größen und Orientierungen zeigen. Außerdem ist es ein Benchmark für die Detektion von sehr kleinen Objekten. Das Dataset enthält sowohl RGB-Bilder als auch multispektrale Daten, was es ideal für die Untersuchung der Auswirkungen zusätzlicher Kanäle wie IR auf die Objekterkennung macht. Es sind mehr als 3700 Objekte in ungefähr 1200 Bildern annotiert. Diese Objekte sind in 9 Klassen (Ship, Camping Car, Car, Pickup, Plane, Tractor, Truck, Van und Vehicles) unterteilt und der Hintergrund der Objekte ist abwechslungsreich, was die Robustheit des trainierten Modells erhöht [50].

Die Auflösung der Bilder liegt bei $12.5 \text{ cm} \times 12.5 \text{ cm}$ pro Pixel, was ausreichend ist um einzelne Fahrzeuge zu erkennen. Die Bilder mit einer Größe von 1024×1024 Pixeln sind bei einer Befliegung im Jahr 2012 in US Amerikanischen Bundesstaat Utah aufgenommen worden [50]. Beispielbilder für jede Klasse sind in den Abbildungen 5.4, 5.11 und 5.15 zu sehen.

4.1.2. Dataset for Objectdetection in Aerial Images (DOTA) 1.5

Das Dataset for Object Detection in Aerial Images (DOTA) eignet sich aufgrund seiner Vielfalt an Klassen und seiner domänenspezifischen Ausrichtung besonders gut als Grundlage für das Pretraining in den Permutationsexperimenten. Der Datensatz enthält eine große Anzahl annotierter Objekte aus verschiedenen Kategorien, die in hochauflösenden Satellitenbildern enthalten sind. Die Bildauflösungen im DOTA-Datensatz variieren stark und reichen von 800×800 bis hin zu $20\,000 \times 20\,000$ Pixeln. Dabei umfasst der Datensatz sowohl RGB-Bilder als auch Graustufenbilder, die den panchromatischen Kanal der GF2- und JL1-Satelliten darstellen. Die Bilder stammen aus verschiedenen Quellen, unter anderem von Google Earth sowie weiteren kommerziellen und institutionellen Anbietern.

Für das Training in dieser Arbeit wurden alle 16 verfügbaren Klassen des DOTA-Datensatzes berücksichtigt. Besonders relevant für die anschließenden Experimente sind die Klassen *Large Vehicle*, *Small Vehicle*, *Plane* und *Ship*, da sie eine hohe Bedeutung für die Objekterkennung im urbanen und infrastrukturellen Kontext besitzen.

Das Training wurde mit dem Modell YOLOv9u über 1500 Epochen durchgeführt, wobei eine *Patience* von 150 verwendet wurde. Alle weiteren Hyperparameter wurden analog zu den in Tabelle 4.5 dargestellten Einstellungen gewählt. Die erzielten Ergebnisse zeigen einen mittleren mAP50-95 von ca. 0,42, was angesichts der großen Anzahl an Bildern und Klassen als zufriedenstellender Wert zu bewerten ist. Es ist kein Overfitting erkennbar, und die Precision-Werte sind insgesamt gut. Die Metriken des Trainingsverlauf sind in Abbildung A.1 und Beispielklassen in Fig. A.2 dargestellt.

4.2. YOLOv9

warum kann
yolov8 obb?
was wurde
dort einge-
fügt?

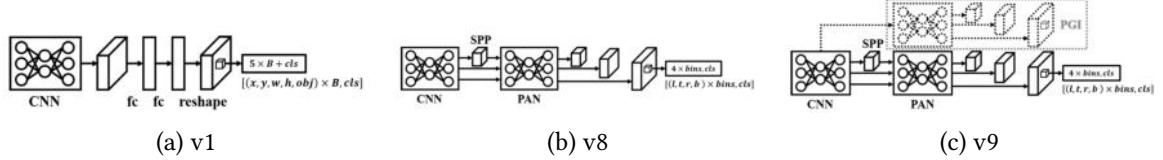


Figure 4.2.: Different YOLO Architectures

Die Entwicklung von YOLO über die verschiedenen Versionen hinweg zeigt deutliche Fortschritte. Während YOLOv1 (s. Fig. 4.2a) durch die One-Stage-Methode eine große Anzahl an Parametern und Rechenaufwand einsparen konnte, indem Vorhersagen durch die FCL in der zweiten Schicht erzeugt wurden, baut YOLOv8 (s. Fig. 4.2a) auf YOLOv5 auf und integriert zahlreiche Optimierungen [71]. Dazu gehört die Efficient Layer Aggregation Network (ELAN)-Integration aus YOLOv7, die um eine zusätzliche Residual Connection erweitert wurde, während der Decoder auf YOLOv6 basiert und unverändert bleibt. Diese Version zeigt zudem eine Verbesserung der Trainingsleistung um etwa 30% und ermöglicht verschiedene nachgelagerte Aufgaben wie Segmentierung, Instanzsegmentierung, Posenabschätzung und Multiple-Object-Tracking. YOLOv9 (s. Fig. 4.2c) integriert schließlich PGI, um zu verhindern, dass wichtige Informationen auf oberflächlicher Netzebene verloren gehen. PGI sorgen für eine robustere Leistung, indem die Originalinformationen maximal tief beibehalten werden [71].

Die zentralen Stärken von YOLOv9 liegen in seiner hohen Verarbeitungsgeschwindigkeit sowie der präzisen Objekterkennung, wodurch eine effiziente Analyse großer Datensätze ermöglicht wird.

Die Varianten YOLOv9u und YOLOv9e unterscheiden sich primär hinsichtlich des verwendeten Labelformats. YOLOv9e verwendet ein normiertes Koordinatensystem:

$$x_{\text{norm}} = \frac{x_{\text{pixel}}}{w_{\text{image}}}, \quad y_{\text{norm}} = \frac{y_{\text{pixel}}}{h_{\text{image}}}, \quad w_{\text{norm}} = \frac{w_{\text{box}}}{w_{\text{image}}}, \quad h_{\text{norm}} = \frac{h_{\text{box}}}{h_{\text{image}}} \quad (4.1)$$

YOLOv9u hingegen verwendet ein Polygonformat:

$$(\text{class_index}, x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4). \quad (4.2)$$

4.2.1. YOLOv9u by Wong Kin Yiu

Die Implementierung von YOLOv9u basiert auf der Version YOLOv8.1.23 [72], um die ursprüngliche YOLOv9 Version um diverse nachgelagerte Aufgaben zu erweitern. Hierzu zählen Instanzsegmentierung, orientierte Objekterkennung, Posenabschätzung, Bildklassifikation sowie transformer-basierte Objekterkennung [30], wobei mit YOLOv8 als Basis ein *Detect-Layer* integriert wird, der

die Vorhersage orientierter BB ermöglicht. In der vorliegenden Arbeit wird YOLOv9u eingesetzt, da diese als Open-Source-Variante verfügbar ist und die zugrundeliegende Methodik durch das veröffentlichte Paper dokumentiert wurde [29]. Das Modell YOLOv9e wird von Ultralytics bereitgestellt, jedoch ohne begleitende wissenschaftliche Veröffentlichung, sodass dessen Aufbau und methodische Grundlagen nicht nachvollziehbar dokumentiert sind [28].

4.2.2. YOLOv9e by Ultralytics

Die YOLOv9-Modellreihe von Ultralytics umfasst mehrere Varianten, die sich in Bezug auf Modellgröße, Genauigkeit und Rechenkomplexität unterscheiden. Alle Varianten integrieren die von Wang et al. entwickelten PGI- und GELAN-Architekturen und unterstützen zudem die Verarbeitung von ABB. Die Bandbreite reicht von der kompakten YOLOv9t-Variante mit einer Eingabebildgröße von 640×640 Pixeln, 2 Millionen Parametern und 7,7 GFLOPs bis hin zur leistungsstärksten YOLOv9e-Variante mit 58,1 Millionen Parametern und 192,5 GFLOPs (siehe Tabelle 4.2).

Aufgrund der höchsten erzielten Genauigkeit ($mAP_{50-95} = 55,6$) sowie der höchsten Rechenkapazität wurde das YOLOv9e-Modell für alle Experimente mit axis-aligned Bounding Boxes verwendet. Die Wahl dieser Variante erfolgte mit dem Ziel, die maximale Detektionsleistung im Rahmen der Evaluierung sicherzustellen. Zudem wird diese Version genutzt, da sie die Verarbeitung von Axis-aligned Bounding Box unterstützt.

Model	Image Size	mAP_{val} 50–95	mAP_{val} 50	Parameters (M)	FLOPs (B)
YOLOv9t	640	38.3	53.1	2.0	7.7
YOLOv9s	640	46.8	63.4	7.2	26.7
YOLOv9m	640	51.4	68.1	20.1	76.8
YOLOv9c	640	53.0	70.2	25.5	102.8
YOLOv9e	640	55.6	72.8	58.1	192.5

Table 4.2.: Comparison of YOLOv9 model variants in terms of accuracy and computational complexity at an input resolution of 640×640 pixels. [29], [73]

4.3. Programming Environment

Die genutzte Programmiersprache ist Python. Die Scripte für den HPC PALMA wurden in Bash geschrieben.

4.3.1. Python Packages

Es wurden diverse Python Packages zur Implementierung genutzt, die im folgenden kurz beschrieben werden.

Computer Vision 2

Ein Teil der "Open Source Computer Vision" Bibliothek ist Computer Vision 2 (CV2) [74]. Die aktuelle Version 4.12.0 wurde am 09.07.2025 veröffentlicht [75].

Die Bibliothek bietet Algorithmen und Funktionen zur Bild- und Videobearbeitung an, die in dieser Arbeit hauptsächlich dazu genutzt werden, die verschiedenen Kanäle zu neuen Bildern zu kombinieren. Hierbei ist zu beachten, dass CV2 Bilder als BGR (Blue, Green, Red) Farbkombination abspeichert, was zu Falschfarbdarstellungen einiger Abbildungen, die in RGB abgebildet werden, (wie Fig. 5.11) führen kann.

Numpy

Ein weiteres Open-Source Projekt ist NumPy, welches 2005 gegründet wurde, um numerische Operationen in Python zu implementieren [76]. Die aktuelle Version ist 2.3.0 und wurde am 07.06.2025 veröffentlicht [77].

Mithilfe dieser Bibliothek kann eine zufällige Verteilung der Bilddaten auf die Folds realisiert werden, indem die eingelesene Bildliste einer zufälligen Permutation unterzogen wird. Darüber hinaus stellt die Bibliothek Funktionen zur Verfügung, mit denen sowohl die Berechnung und Reskalierung des NDVI-Kanals als auch die Transformation der Bounding-Box-Koordinaten in das für das YOLO-Framework erforderliche Format durchgeführt werden können. Zusätzlich ermöglicht sie die Umrechnung axis-aligned Bounding Boxes in oriented Bounding Boxes sowie die Flächenberechnung beider Bounding-Box-Typen.

Scipy

SciPy ist in der Version 1.16.0 am 22.06.25 veröffentlicht worden [78]. Diese Bibliothek bietet Algorithmen für wissenschaftliche Berechnungen in Python an [78]. Die Unterfunktion "ConvexHull" wird zur Flächenberechnung der oriented Bounding Boxen genutzt.

4.3.2. Additional Python-Packages for data analysis

Zur Analyse der Modelle wurden die folgenden Bibliotheken verwendet.

Matplotlib

Die Bibliothek "Matplotlib" wurde in der Version 3.10.0 am 23.12.2024 veröffentlicht[79]. Im Rahmen dieser Arbeit werden durch diese Erweiterung die Datenanalysen und die Ausgabe der Plots ermöglicht.

Seaborn

"Seaborn" ist in der Version 0.13.2 im Januar 2024 veröffentlicht worden [80]. Die Bibliothek ermöglicht das Erstellen von Grafiken, wie Boxplots, zur vereinfachten Datenanalyse.

pandas

Die Bibliothek "pandas" ist ebenfalls ein Open Source Datenanalyse und -manipulierungstool. Die aktuelle Version 2.3.1 wurde am 07.07.2025 veröffentlicht. Die Bibliothek wird auch zur Datenanalyse in dieser Arbeit verwendet [81].

4.4. Implementation Preprocessing

Damit die Trainingsdaten vom Deep-Learning-Modell YOLOv9 verarbeitet werden können, ist eine Vorverarbeitung notwendig, die deren Format und Struktur an die Anforderungen des Modells anpasst. Der genaue Ablauf dieser Vorverarbeitung wird im Folgenden beschrieben.

weniger Funktionsnamen

4.4.1. DOTA Dataset

Im Rahmen der Datenvorverarbeitung wurden ausschließlich die originalen Rohbilder des DOTA-Datensatzes verwendet, ohne dass ein Zuschnitt oder eine anderweitige Modifikation der Bildinhalte vorgenommen wurde. Der Fokus der Verarbeitung lag auf der Umwandlung der vorhandenen Labeldaten in das YOLOv9-OBB-Format. (vgl. Kap. 4.2).

Für jedes Bild stellt DOTA eine separate Textdatei bereit, die alle zugehörigen Annotationen enthält. Zu Beginn wurde eine standardisierte Ordnerstruktur nach dem YOLO-Schema erstellt, bestehend

aus den Unterverzeichnissen `train/images`, `train/labels`, `val/images` und `val/labels`. Die Annotationsdateien wurden daraufhin zeilenweise eingelesen und verarbeitet.

Jede Zeile, die ein einzelnes Objekt beschreibt, wurde zunächst an die jeweilige Bildgröße angepasst, wobei die absoluten Koordinaten der Objektrahmen in relative Werte normalisiert wurden. Da die Auflösung der Bilder im DOTA-Datensatz stark variiert (zwischen 800×800 und $20,000 \times 20,000$ Pixeln), war eine dynamische Skalierung erforderlich. Die konvertierten Informationen wurden anschließend in das YOLOv9u-Format überführt, wobei die Klassen-ID jeweils am Anfang der Zeile stand. Die so erzeugten Labeldateien wurden unter Beibehaltung des ursprünglichen Dateinamens gespeichert.

Insgesamt wurden sämtliche 16 im DOTA-Datensatz enthaltenen Objektklassen berücksichtigt und entsprechend konvertiert. Die Bilddaten selbst blieben unangetastet und wurden in ihrer Originalform auf den HPC Cluster PALMA übertragen. Aufgrund der hohen Anzahl an verfügbaren Annotationen gestaltete sich der Konvertierungsprozess als auch der Trainingsprozess zeitintensiv.

4.4.2. VEDAI Dataset

Die Verarbeitung des VEDAI-Datensatzes erfolgt über eine zentrale Hauptmethode, in der mehrere konfigurierbare Parameter in Form von booleschen Variablen bereitgestellt werden. Besonders wichtig ist hierbei der Parameter `oriented`, welcher angibt, ob die Bounding Boxes (BBs) als achsenparallele Rechtecke oder als orientierte Rechtecke im OBB-Format generiert werden. Zusätzlich kann über `bool_create_yaml` gesteuert werden, ob für jedes erzeugte Trainingsset eine entsprechende YAML-Datei automatisch mitgeneriert werden soll.

Ein weiterer Parameter, `merge_ir_bool`, ermöglicht es, das Infrarotbild (IR) in die erzeugten Bilddaten zu integrieren. Mithilfe des Parameters `namestring` lässt sich jedem erzeugten Trainingsset ein individueller Name zuweisen. Entscheidend für die Zusammensetzung der Bildkanäle ist das Dictionary `perm_object`, in dem über boolesche Werte festgelegt wird, welche Kanäle (z. B. R, G, B, IR, NDVI) in das finale Bild aufgenommen werden sollen.

In der zentralen Hauptfunktion wird anschließend bestimmt, welche Teilmengen des Datensatzes erzeugt werden sollen. Dies betrifft u. a. Ablation-Datasets, Permuations-Datasets oder ABB- und OBB-Datasets. Für beide erstgenannten Varianten folgt der Ablauf einem konsistenten Schema: Die Parameter `oriented` und `merge_ir_bool` werden aktiviert, ein Speicherpfad für das erzeugte Dataset definiert, und das `perm_object` wird entsprechend der gewünschten Kanalzusammensetzung konfiguriert. Beispielweise wird für die Kombination RGIR lediglich R, G und IR im `perm_object` auf True gesetzt. Im Anschluss wird mittels einer for-Schleife über die Folds 0 bis 5

iteriert und für jeden Fold mit der Methode `create_fold_cross_validation` das entsprechende Datenset aus Trainings-, Validierungs- und Testdaten generiert.

Erzeugung von Folds (`create_fold_cross_validation`) Diese Methode übernimmt die vollständige Erstellung der Ordnerstruktur im für YOLO erforderlichen Format (`train/images`, `train/labels`, `val/images`, `val/labels`, `test/images`, `test/labels`). Zudem wird eine YAML-Datei erzeugt, die sowohl die Pfade zu den Bilddateien als auch die Anzahl der enthaltenen Kanäle enthält. Ein Trainingsdatensatz setzt sich stets aus drei Komponenten zusammen:

1. **Trainingsdaten:** Bestehen aus den Bildern der übrigen Folds (also alle außer dem aktuellen Fold und Fold 6).
2. **Validierungsdaten:** Entsprechen den Bildern des jeweils aktuellen Folds.
3. **Testdaten:** Bestehen stets aus den Bildern von Fold 6.

Der aktuell ausgewählte Fold wird aus der Liste mit allen Folds entfernt, um Überschneidungen mit Trainings-, Validierungs- und Testdaten auszuschließen. Anschließend wird die zentrale Datei mit allen Labelinformationen des Datensatzes eingelesen. Über eine `for`-Schleife werden alle Folds durchlaufen, die nicht dem aktuellen Fold entsprechen. Dabei wird jeweils die Datei mit den Bildnamen eingelesen und für jedes dieser Bilder die Funktion `create_image_and_label` aufgerufen.

Diese Funktion liest für jede Zeile das zugehörige RGB- und IR-Bild ein und filtert die Labels so, dass nur die zum aktuellen Bild gehörenden Annotationen weiterverarbeitet werden. Das neue Bild wird anschließend in den jeweiligen Zielpfad kopiert, wobei intern die Funktion `merge_RGB_IR_image` zur Anwendung kommt. Diese Funktion stellt sicher, dass nur die im `perm_object` spezifizierten Kanäle in die Zielfile übernommen werden (vgl. Funktion `merge_RGB_IR_image`). Im Anschluss wird die Funktion `create_label_file` aufgerufen, um das zugehörige Label-File im gewünschten Format zu erzeugen.

Nach dem Abschluss der Verarbeitung aller Trainingsfolds wird abschließend sowohl der Validierungsfold (aktueller Fold) als auch der Testfold (immer Fold 6) mithilfe der Funktion `create_image_and_label` separat verarbeitet. Nach deren erfolgreichem Durchlauf gilt das Datenset für die jeweilige Kanal-Kombination als vollständig erstellt.

Fusion von RGB- und IR-Bildern (`merge_RGB_IR_image`) Diese Methode lädt die entsprechenden RGB- und IR-Bilder und erzeugt basierend auf den im `perm_object` definierten Vorgaben eine kombinierte Bilddarstellung. Die Fusion der Kanäle erfolgt unter Verwendung von `cv2.merge`.

Dabei ist zu berücksichtigen, dass CV2 standardmäßig das BGR-Format (Blue, Green, Red) verwendet, wenn Bilder kanalweise zerlegt oder gespeichert werden. Da die aktiven Kanäle jedoch in der im `perm_object` vorgegebenen Reihenfolge zusammengeführt werden, ist dieser Aspekt für das Ergebnis von nachgeordneter Relevanz.

Optional kann innerhalb dieser Funktion zusätzlich ein NDVI-Kanal erzeugt werden. Zur numerischen Stabilisierung bei der Berechnung des NDVI (vgl. Definition 4.4.1) wird der Nenner ($IR + R$) auf einen Minimalwert von 0,01 gesetzt, um Divisionen durch Null zu verhindern. Die resultierenden NDVI-Werte werden anschließend skaliert und in einem 8-Bit-Format zurückgegeben.

Vor dem Zusammenführen werden die RGB-Kanäle stets separat extrahiert. Das Schreiben der resultierenden Bilddateien an den definierten Zielpfad erfolgt schließlich über die Funktion `copy_image`

Definition 4.4.1

Let ρ_{NIR} denote the reflectance in the near-infrared band and ρ_{RED} denote the reflectance in the red band. The *Normalized Difference Vegetation Index* (NDVI) is defined as

$$NDVI := \frac{\rho_{\text{NIR}} - \rho_{\text{RED}}}{\rho_{\text{NIR}} + \rho_{\text{RED}}} \quad (4.3)$$

Erzeugung von Label-Dateien (`create_label_file`) Zur Erstellung der Label-Datei wird zunächst der Pfad zur Ausgabedatei bestimmt und das zugehörige Bild eingelesen, um die Bilddimensionen für die spätere Normalisierung zu ermitteln. Anschließend werden alle übergebenen Label-Informationen durchlaufen. Für jedes Label wird mithilfe der Funktion `get_bounding_box_in_px` (basierend auf einem Skript nach Razakarivony et al.) die Bounding Box in Pixelkoordinaten berechnet.

Im nächsten Schritt wird die Zielzeile im YOLO-Format erstellt: Die Klasse wird dabei in eine numerische ID zwischen 0 und 8 überführt (mittels einer internen Mapping-Funktion), gefolgt von den acht Koordinatenpunkten der Bounding Box, jeweils durch Leerzeichen getrennt.

Im Fall von `oriented == True` werden die acht Punkte einer rotierbaren Box gespeichert. Ist `oriented == False`, können die Koordinaten entweder in klassischer YOLO-Darstellung (zentrumsbasiert mit Höhe und Breite) oder als achsenparalleles Rechteck im OBB-Format gespeichert werden. Die Umwandlung erfolgt hier durch unterschiedliche Konvertierungsfunktionen. Abschließend wird die Zeile in die Datei geschrieben und der Prozess für das nächste Label wiederholt.

Berechnung der Bounding Box in Pixeln (get_bounding_box_in_px) Diese Funktion dient der Extraktion und Transformation der Labelkoordinaten aus dem ursprünglichen VEDAI-Format. Zunächst wird der Label-String anhand von Leerzeichen in seine Bestandteile zerlegt. Die Position (x, y) sowie die Orientierung befinden sich an den Stellen 1 bis 3 des Arrays, während sich die Klassen-ID an Position 12 befindet. Die Eckpunkte der Bounding Boxen (x-Werte: Positionen 4–8, y-Werte: 8–12) werden extrahiert und daraus die Seitenlängen berechnet.

Die längsten Seiten des Objekts werden identifiziert und als Fahrzeuggänge, die kürzeren als Fahrzeugbreite angenommen. Aus den längsten Seiten wird ein Richtungsvektor berechnet, und der Winkel der Fahrzeugausrichtung über den Arctan bestimmt. Die vier Eckpunkte der orientierten Box werden schließlich unter Berücksichtigung von Fahrzeugmitte, Länge, Breite und Rotationswinkel berechnet.

Ist `oriented == True`, so werden die vier Eckpunkte als ganzzahlige Koordinaten-Tupel zurückgegeben. Ist `oriented == False`, wird aus den minimalen und maximalen x- und y-Werten der rotierten Box eine achsenparallele Bounding Box erzeugt und zurückgegeben.

4.4.3. Fold Creation

Die Erstellung der Folds erfolgt über die zentrale Hauptfunktion `main`, welche auf der im vorherigen Kapitel beschriebenen Vorverarbeitung basiert. Diese Funktion liest Textdateien ein, in denen jeder Bildname zeilenweise enthalten ist. Ziel des gesamten Skripts ist es, solche Textdateien zu generieren, welche später zur Trainings-, Validierungs- und Testdatenerstellung genutzt werden. Zunächst wird eine Liste aller im Quellverzeichnis vorhandenen Bilder erstellt. Für jedes Bild wird dabei zusätzlich die Anzahl der Objekte pro Klasse gespeichert. Anschließend wird die Methode `create_own_folds` aufgerufen, wobei die Anzahl der Folds flexibel gewählt werden kann – standardmäßig werden fünf Folds genutzt.

Die Methode `create_own_folds` erhöht die gewünschte Fold-Anzahl um eins, um einen separaten Testfold zu berücksichtigen. Danach wird ein leeres mehrdimensionales Array erzeugt, das der neuen Anzahl an Folds entspricht. Die übergebene Bildliste wird einmal zufällig durchmischt, bevor den einzelnen Folds initial je ein Bild zugewiesen wird. Im Anschluss wird über eine separate Funktion (vgl. Funktion `count_objects_in_fold_arr`) ermittelt, wie viele Objekte jeder Klasse in jedem Fold enthalten sind, um eine gleichmäßige Verteilung sicherzustellen.

Die eigentliche Verteilung erfolgt iterativ in einer `while`-Schleife, die so lange läuft, bis alle Bilder verteilt sind. Innerhalb dieser Schleife wird für jedes Bild analysiert, welche Klasse im Bild am

seltensten vorhanden ist (vgl. Funktion `get_smallest_class_in_image`). Wird genau eine seltene Klasse identifiziert, so wird das Bild jenem Fold zugewiesen, der aktuell die geringste Anzahl von Objekten dieser Klasse enthält. Falls mehrere seltene Klassen vorhanden sind, wird anhand einer vordefinierten Frequenzbewertung entschieden. Jede Klasse wird hierbei anhand ihrer Gesamtverteilung im VEDAI-Datensatz mit einem Gewicht bewertet, das auf einer Skala von 0 (häufig) bis 8 (selten) liegt (s. Tab. A.2). Es wird die Klasse mit dem höchsten Frequenzwert ausgewählt, und entsprechend der geringsten Objektanzahl dieser Klasse wird das Bild einem Fold zugewiesen. Bilder ohne Objekte werden separat in einem Array für leere Bilder gespeichert.

Nachdem alle Bilder mit Objekten verteilt wurden, erfolgt die Zuweisung der leeren Bilder. Dazu wird zunächst eine ganzzahlige Division der Anzahl leerer Bilder durch die Anzahl der Folds durchgeführt, um eine möglichst gleichmäßige Grundverteilung zu ermöglichen. Der verbleibende Rest wird anschließend über eine zusätzliche Schleife auf jene Folds verteilt, die aktuell die geringste Bildanzahl enthalten.

Am Ende der Verteilung erfolgt eine erneute Zählung der Objekte in jedem Fold. Die finalen Fold-Zuweisungen werden anschließend als Textdateien gespeichert – jeweils eine Datei pro Fold (insgesamt sechs Dateien, Fold 0 bis Fold 5), die ausschließlich die entsprechenden Bildnamen enthalten. Die Dateien sind im gleichen Format wie die originalen VEDAI-Folds gehalten, um die Anwendung der in Kapitel 4.4.2 beschriebenen Vorverarbeitung möglichst einfach und konsistent zu gestalten.

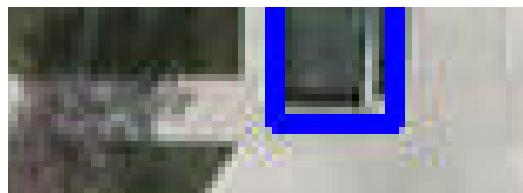
Zählung der Objekte in den Folds (`count_objects_in_fold_arr`) Zur Auswertung der Objektverteilung innerhalb der Folds wird die Methode `count_objects_in_fold_arr` verwendet. Diese Funktion erzeugt eine Datenstruktur, die für jeden Fold die Anzahl der enthaltenen Objekte pro Klasse speichert. Die Zählung erfolgt durch eine doppelt geschachtelte Schleife, in der alle Bilder jedes Folds durchlaufen und die Objekte gezählt werden. Am Ende liefert die Funktion eine vollständige Übersicht der Objektverteilung als Grundlage für weitere Entscheidungen.

Identifikation der kleinsten Klassenverteilung (`get_indices_of_folds_with_smallest_object_count`) Die Methode `get_indices_of_folds_with_smallest_object_count` dient dazu, für eine gegebene Objektklasse den oder die Folds mit der geringsten Anzahl dieser Klasse zu ermitteln. Basierend auf den Zähldaten aus `count_objects_in_fold_arr` wird über alle Folds iteriert und geprüft, welcher Fold die minimalen Objektzahlen für die entsprechende Klasse enthält. Dabei wird eine Liste mit allen zutreffenden Indizes zurückgegeben, sofern mehrere Folds denselben Minimalwert aufweisen.

Bestimmung der seltensten Klasse im Bild (get_smallest_class_in_image) Die Methode `get_smallest_class_in_image` analysiert ein gegebenes Bild und gibt jene Objektklasse zurück, die in diesem Bild am wenigsten vertreten ist. Diese Information dient als Ausgangspunkt für die optimale Zuweisung des Bildes zu einem Fold und trägt zur Balance der Objektverteilung bei.

4.5. Challenges Preprocessing

Bei der Verarbeitung des VEDAI-Datensatzes traten mehrere technische Herausforderungen auf. Zunächst mussten die vorhandenen Label in das YOLOv9-OBB-Format konvertiert werden, um mit dem gewählten Modell kompatibel zu sein. Dabei zeigte sich, dass ein kleiner Teil der Daten (7 von 3 757 Instanzen) Koordinatenwerte aufwies, die außerhalb des zulässigen Bereichs lagen (d. h. kleiner als 0 (s. Fig. 4.3a) oder größer als 1 (s. Fig. 4.3b)). Dieses Problem wurde durch den Einsatz von Exception Handling sowie durch Rundung der Werte auf den gültigen Wertebereich behoben.



(a) Two Coordinate out of range and smaller 0



(b) Two Coordinates out of range and higher 1

Figure 4.3.: Example for label coordinates outside of the image (for full-sized Image see A.3)

4.6. 6-Fold-Cross-Validation

Um die Robustheit der Ergebnisse sicherzustellen und eine fundierte Vergleichbarkeit verschiedener Modelle zu ermöglichen, wurde in dieser Arbeit eine 6-Fold-Cross-Validation angewendet. Insbesondere wurde Wert darauf gelegt, die Evaluation nicht nur auf einem Teildatensatz durchzuführen, sondern eine Mehrfachteilung der Daten vorzunehmen, sodass Schwankungen durch unterschiedliche Trainings- und Validierungsaufteilungen minimiert werden.

Im Gegensatz zu den von Razakarivony et al. [50] bereitgestellten Folds, deren Zusammenstellung identische Bilder gleichzeitig im Trainings- und Validierungssatz aufweist, wurden für diese Arbeit eigene, disjunkte Folds generiert. Letztere gewährleisten, dass keine Überlappungen zwischen Trainings- und Validierungsdaten existieren und somit die Gefahr von verfälschten Trainingsergebnissen durch Overfitting ausgeschlossen ist. Die genaue Methodik der Fold-Erstellung ist in Kapitel 4.4.3 beschrieben.

Die Datenaufteilung erfolgte in sechs Folds (0-5): Fünf Folds (Fold 0 bis 4) wurden jeweils für Training und Validierung genutzt, während Fold 5 ausschließlich für abschließende Testzwecke reserviert blieb. Dadurch wurde erreicht, dass die finale Evaluation auf bisher ungesehenen, neutralen Daten stattfand.

Ein besonderes Augenmerk galt einer homogenen Verteilung der Objekte und Objektklassen über alle Folds hinweg. Jeder Fold umfasst zwischen 207 und 221 Bilder, darin enthalten sind jeweils drei bis sieben reine Hintergrundbilder ohne Objekte. Dies ermöglichte eine ausgewogene und repräsentative Bewertung der Modelle hinsichtlich aller vorkommenden Klassen. Die resultierende Klassen- und Bildverteilung pro Fold ist in Tabelle 4.3 dargestellt.

Insgesamt trägt dieses Verfahren dazu bei, dass die experimentellen Resultate eine hohe Validität aufweisen und zuverlässig Rückschlüsse auf die Auswirkungen der zusätzlichen Kanäle bei den verwendeten Modellen erlauben.

Class/Fold	0	1	2	3	4	5
<i>Car</i>	229	239	226	225	240	225
<i>Truck</i>	51	57	50	49	51	50
<i>Ship</i>	30	28	29	30	27	27
<i>Tractor</i>	30	32	33	30	31	30
<i>Camping car</i>	65	72	64	69	64	63
<i>Van</i>	18	17	17	17	16	16
<i>Vehicle</i>	34	37	34	34	39	33
<i>Pick-Up</i>	164	161	157	160	159	157
<i>Plane</i>	4	11	4	18	4	7
<i>Quantity Images</i>	206	214	221	211	207	209
<i>Background Images</i>	7	3	3	3	3	3

Table 4.3.: Class distribution across folds.

4.7. High-Performance-Cluster PALMA

Für das Training der Modelle wurde das Hyper-Performance-Cluster (HPC) PALMA der Universität Münster genutzt. Als Arbeitsumgebung kam dabei eine isolierte Python-Umgebung (UV [82]) zum Einsatz. Die Ausführung der Trainingsprozesse erfolgte über selbst entwickelte Bash-Skripte, die auf unterschiedlichen GPU-Knoten des Clusters (u. a. HGX-Knoten) verteilt wurden, um eine effiziente Parallelisierung der Trainings zu ermöglichen.

PALMA (Abkürzung für "Paralleles Linux-System für Münsteraner Anwender") wird von dem Center for Information Technology (CIT) der Universität Münster, basierend auf Hardware der

Firma MEGWARE, bereitgestellt [83] und verfügt über die in Tab. 4.4 technische Spezifikationen [83]:

Palma Specifications	
Total number of CPU cores	16,272
Memory	77,568 GB
Number of compute nodes	444
Processor	Intel Xeon Gold 6140 (18 cores, 2.30 GHz, Skylake architecture)
Network interconnect	100 GHz/s Intel Omni-Path
Storage system	GPFS with a total capacity of 2.4 PB
Linpack performance	$R_{max} = 800 \text{ TFLOP/s}$; $R_{peak} = 1,277 \text{ TFLOP/s}$
Operating system	Rocky Linux 9

Table 4.4.: System overview PALMA

Die hohe Rechenleistung und Speicherkapazität des Clusters machten es möglich, auch komplexe Trainingsprozesse mit vielen hochauflösenden Bilddaten effizient und schnell zu verarbeiten.

4.8. Bash Scripts

Für das Training der Modelle wurden mehrere Bash-Skripte entwickelt, die für den Hochleistungsrechner PALMA ausgelegt waren. Die Trainingsläufe erfolgten primär über SLURM Job Arrays, wobei jeder Fold einer Permutation einem separaten Job zugeordnet wurde. Somit entsprach ein Modell stets einem Fold innerhalb einer bestimmten Permutation.

Die Entscheidung zwischen Axis-aligned Bounding Box und Oriented Bounding Box erfolgte unter denselben Trainingsparametern wie bei den Permutationsexperimenten (s. Tab. 4.5), jedoch mit unterschiedlichen Modellen: YOLOv9e für ABB und YOLOv9u für OBB, da jeweils nur diese Varianten die entsprechenden Bounding-Box-Formate unterstützen. In diesem Fall wurde kein vortrainiertes Modell genutzt, sondern ein Training *from scratch* durchgeführt.

Im Anschluss wurden die Permutationsexperimente durchgeführt. Dabei kamen einheitlich die in Tab. 4.5 aufgeführten Hyperparameter zum Einsatz. Die Bildauflösung betrug 1024×1024 Pixel, die Trainingsdauer umfasste 500 Epochen, und ein frühzeitiges Beenden war durch einen patience-Wert von 0 deaktiviert, um eine konsistente Vergleichbarkeit sicherzustellen. Grundlage war ein auf dem DOTA-Datensatz vortrainiertes Modell, wobei durchgängig die Architektur YOLOv9u (s. Kap. 4.2.1) verwendet wurde.

Für die Ablationsstudien wurde hingegen auf den Einsatz eines vortrainierten Modells verzichtet, während alle übrigen Trainingsparameter unverändert blieben.

Nach Abschluss des Trainings wurde jeweils das Modell mit der besten Validierungsleistung ausgewählt und anschließend auf den entsprechenden Testfold angewendet. Die Evaluation erfolgte in beiden Fällen in einem einmaligen Durchlauf, um konsistente und vergleichbare Ergebnisse zu gewährleisten.

Hyperparameter	Value / Description
Task	Oriented Bounding Box (OBB)
Mode	Training
Model	YOLOv9u, pretrained on DOTA
Dataset	VEDAI
Epochs	500
Batch size	4
Image size	1024 × 1024 px
Early stopping (patience)	0 (disabled)
Optimizer	Auto
Initial learning rate (lr_0)	0.01
Momentum	0.937
Weight decay	0.0005
Warmup epochs	3
Data augmentation	FlipLR 0.5, Mosaic 1.0, Erasing 0.4, AutoAugment randaugment
Device	GPUs 0,1,2,3
Workers	8
Save directory	/scratch/tmp/t_liet02/cross_validation/rgb/fold4

Table 4.5.: Key training hyperparameters at the RGB Permutation (Fold 4)

5 | Results

Das Kapitel orientiert sich an Abbildung 4.1 und behandelt zunächst den Vergleich von ABB- und OBB-Formaten, anschließend die Permutationsexperimente zur Überprüfung der Robustheit und abschließend die Ablationsstudie zur Bewertung einzelner Komponenten.

In den Bildcollagen (vgl. Abb. 5.4, 5.11 und 5.15) wird das in Tab. A.1 definierte Farbschema verwendet, um die dargestellten Modellvorhersagen klar unterscheidbar zu machen und eine eindeutige Zuordnung zu den jeweiligen Klassen zu gewährleisten.

5.1. Oriented Bounding Boxes and Axis Aligned Bounding Boxes

In Abbildung 5.1 wird ein Vergleich zwischen OBB und ABB für verschiedene Objektklassen dargestellt. Besonders deutlich wird dieser Unterschied beim LKW-Beispiel: In Abbildung 5.1a nimmt die ABB mit ca. 1378 px eine deutlich größere Fläche ein als die OBB in Abbildung 5.1b mit ca. 967 px. Beim Vergleich stark rotierten Objekten, wie etwa bei den Schiffen in Abbildung 5.1c und 5.1d, zeigt sich, dass die OBB das Objekt selbst deutlich präziser umschließt und weniger umliegendes Areal einbezieht. Ein weiterer Vorteil der OBB ist das Fehlen von Überlappungen zwischen den Boxen. Für weitere Beispiele s. Abb. 5.4. Die Zeile "abb in obb" beschreibt, dass das OBB-Modell (YOLOv9u) mit ABB-Bounding Boxen trainiert wurde. Dies bedeutet, dass die im VEDAI-Datensatz vorhandenen OBBs in das für YOLOv9 vorgesehene Bounding Box Format (s. Equ. in Kap. 4.2) als ABB (also mit 8 Koordinaten, die eine achsenparallele Bounding Box aufspannen) umgewandelt wurden.

Bilder in Originalgröße in Appendix packen

nicht nur erläutern was man sieht sondern auch was man daraus folgert (analyse) im Bildunterschrift

Ein Vergleich aller Bounding Box-Flächen (vgl. Abbildung 5.2) verdeutlicht, dass OBB im Mittel kompakter ist (Median ABB: ca. 1000 px, Median OBB: >700 px). Die ABB-Flächen entsprechen nahezu exakt den "abb in obb", da die Bounding Boxes fast identisch sind; der einzige Unterschied liegt im trainierten Modell und den Berechnungen der Ecken der ABB. Die Bounding Boxen nehmen als OBB (ca. 700 px pro Box) etwa 0,066% der Gesamtfläche des Bildes ein, als ABB (ca. 1000 px pro Box) etwa 0,095%. Damit liegen OBB knapp unterhalb der Untergrenze für sehr kleine Objekte nach Chen et al. [52] (s. Kap. 3) und gelten als "very small objects", während ABB oberhalb der Untergrenze für kleine Objekte liegt und damit als "small object" gilt (Berechnung s. A.2.1).

Die Analyse der mAP50–95 des besten Validierungsmodells auf dem Validierungsdatensatz (Abbildung 5.3) zeigt, dass das OBB-Modell in allen Fällen eine bessere Leistung erzielt als das ABB-Modell. Alle Modelle für die die Einschätzung der Performance von OBB und ABB wurden ausschließlich auf Basis von RGB-Bildern trainiert. Interessanterweise erzielen ABBs auf dem OBB-Modell eine höhere Performance als die angepassten OBB-Boxen innerhalb desselben Modells (yolov9u). Das selbe Phänomen lässt sich auch auf dem Testdatensatz beobachten (Abbildung 5.5), sodass ABB in Verbindung mit dem OBB-Modell bessere Ergebnisse liefert. Auffällig ist, dass bei den ABB-basierten Modellen (sowohl ABB als auch "abb in obb") stets Fold 1 den besten Wert beim Validation Dataset auf dem Validation Dataset erreicht hat, während beim OBB-Modell Fold 0 dort die besten Ergebnisse lieferte (s. Tab. 5.1 und Fig. 5.3). Die Ergebnisse der jeweiligen Folds mit dem höchsten mAP50-95 Wert aus Abb. 5.3 sind durch den roten Punkt in Abbildung 5.5 gekennzeichnet. Also hat laut Tab. 5.1 und Fig. 5.3 der Fold 1 den höchsten mAP Wert im ABB Modell erzielt. Auf dem Testfold hat dieser Fold 1 jedoch lediglich das untere Quantil erreicht, wie in Abb. 5.5 am roten Punkt zu sehen.

nächster Satz
ist eigentlich
analyse
(auskommen-
tiert)

Model	Best mAP Fold (Validation)	Best mAP Fold (Test)
abb	1	1
obb	0	0
abb in obb	1	1

Table 5.1.: Best fold per model at Bounding Box format

Die Auswirkungen der unterschiedlichen Rotationen der erkannten Objekte lassen sich in Abbildung 5.4 exemplarisch an der Klasse plane beobachten. Die Bounding Boxes unterscheiden sich hinsichtlich ihrer Rotationswinkel deutlich zwischen den drei Modellen. Abweichungen in der Überlappung der erkannten Bounding Boxes mit den Ground-Truth-Annotationen treten hingegen wesentlich seltener auf, was die höhere mAP-Performance der ABB-basierten Modelle erklärt. Auf weitere Auffälligkeiten, wie die Schwierigkeiten bei der Trennung von Objekten und Hintergrund, wird in den folgenden Abschnitten eingegangen.

Aufgrund der geringeren Wahrscheinlichkeit für Überlappungsfehler sowie der präziseren Objektumrandung wurde das OBB-Modell für die Durchführung der Permutationstests ausgewählt.



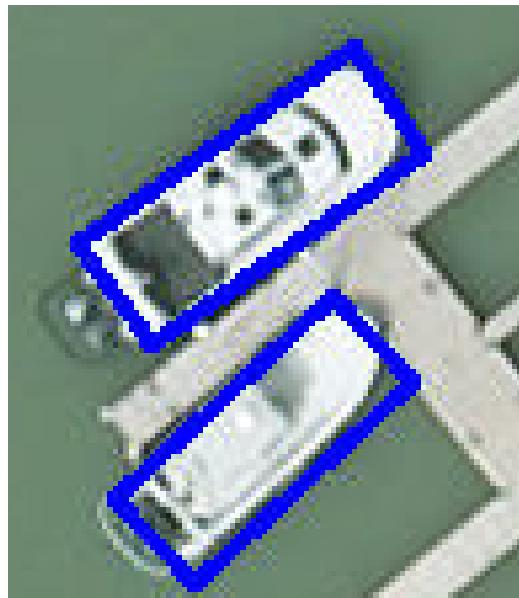
(a) Label: "Truck", abb format, area: $\approx 1378\text{px}$



(b) Label: "Truck", obb format, area: $\approx 967\text{px}$



(c) Labels: "Ship", abb format



(d) Labels: "Ship", obb format

Figure 5.1.: Comparison of the bounding box formats of two different object classes (for full-sized images see A.4). OBB enclose objects much more accurately than ABB, as they include fewer irrelevant background areas. The reduced risk of overlap with neighbouring structures allows the model to extract more relevant object features and thus learn a more accurate representation of the classes (Own representation).

5. RESULTS

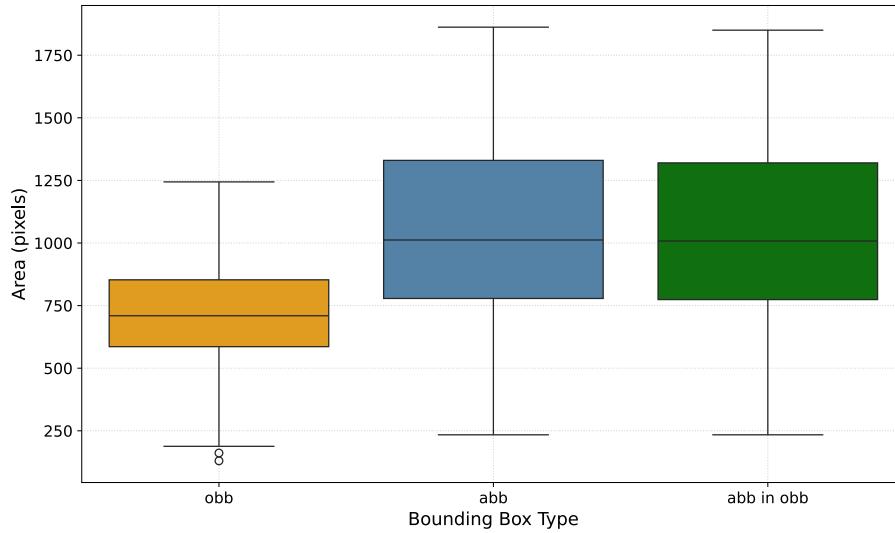


Figure 5.2.: Comparison of Bounding Box Area in pixels for ABB, OBB, abb in obb. "aab in obb" means, that Axis-aligned Bounding Box converted to obb was used in the YOLOv9u model with zero rotation. Due to their rotation-based adjustment, OBBs are more compact than ABBs and comprise less background. Since the calculation of 'ABB in OBB' areas yields almost identical values, it can be assumed that OBB models achieve higher performance due to more precise limitations (Own representation).

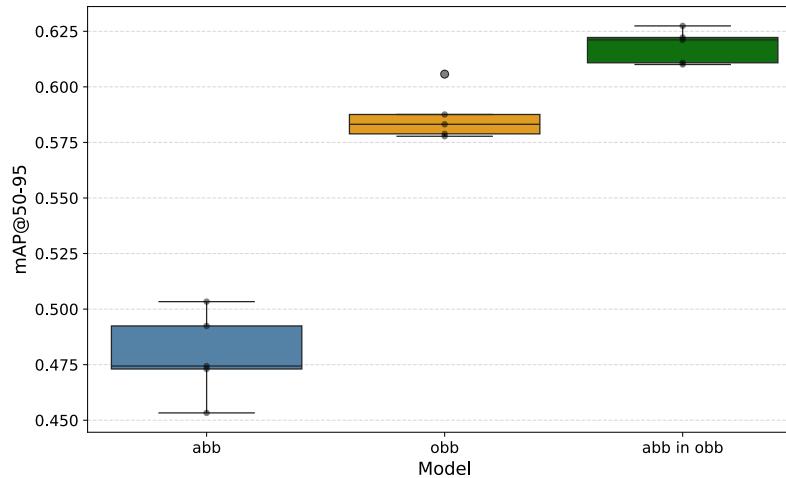


Figure 5.3.: Comparison of mAP50-95 values for ABB and OBB (Best validation model on validation dataset). "aab in obb" means, that Axis-aligned Bounding Box converted to obb was used in the YOLOv9u model with zero rotation. ABB show a lower mAP, presumably due to the model, while OBB are superior due to their better object adaptation. Notably, 'ABB in OBB' achieves higher values, which may indicate the strong rotation sensitivity of OBB (Own representation).

	Car	Truck	Camping Car	Tractor	Plane	Ship	Vehicle	Pick-Up	Van
GT (ABB)									
GT (OBB)									
ABB									
OBB									
ABB in OBB									

Figure 5.4.: Examples of different classes for the object recognition performance of different bounding box formats. The numbers displayed on the bounding boxes indicate the confidence of the model in the classification of the respective box. For Color Scheme explanation see Tab. A.1. The example of the class *Plane* clearly illustrates the influence of rotational capability on prediction quality. ABB proposes a BB that is very close to the GT, while OBB and ‘ABB in OBB’ apply different rotations and are therefore based on different GTs. This results in lower mAP values for OBB, whereas ‘ABB in OBB’ achieves higher performance due to its larger bounding boxes and better overlap with the GT. (Own representation)

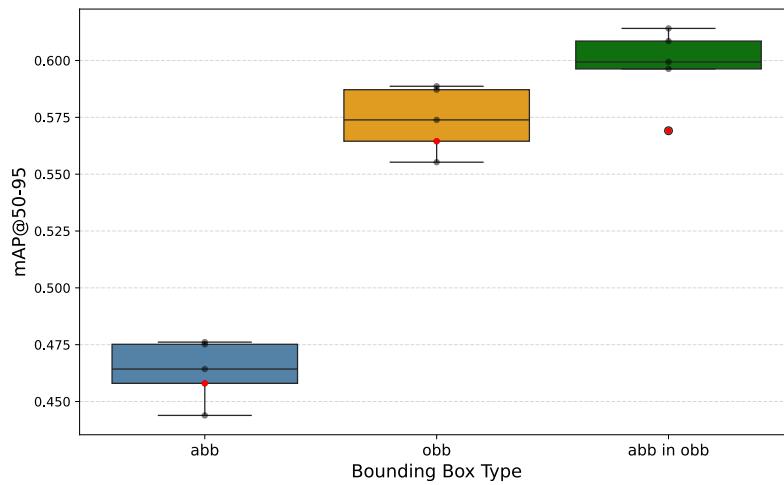


Figure 5.5.: Comparison of mAP₅₀₋₉₅ values for ABB and OBB (Best validation model on test dataset). "aab in obb" means, that Axis-aligned Bounding Box converted to obb was used in the YOLOv9u model with zero rotation. The red dot is the fold, that has the best mAP₅₀₋₉₅ at the validation dataset (see fig. 5.3). On the test dataset, ABB achieves poorer results, presumably due to model limitations, while OBB achieves significantly better performance due to more precise object boundaries. The 'ABB in OBB' configuration shows the best performance, as even small rotation deviations in the OBB model strongly influence the mAP and the model's GT BB are significantly larger (Own representation).

5.2. Permutation Experiments

Im Folgenden werden zunächst die allgemeinen Ergebnisse der Modelle hinsichtlich der Detektionsleistung beschrieben und mit den in Abbildung 5.6 dargestellten mAP50–95-Werten verglichen. Anschließend erfolgt eine detaillierte Analyse der Klassenerkennungsleistung anhand der Differenzmatrizen.

Abbildung 5.6 verdeutlicht, dass das RGIR-Modell die beste mAP50–95-Performance auf dem Testfold erreicht, gefolgt von den Modellen RIRB und RGB. Auffällig sind dabei die vergleichsweise hohen Unterschiede zwischen den Quantilen, was auf eine stärkere Streuung der Ergebnisse hindeutet. Das RGBIR-Modell liegt im Ranking an vierter Stelle, während das IRGB-Modell deutlich schwächer abschneidet als die zuvor genannten Modelle, jedoch immer noch bessere Ergebnisse erzielt als die beiden Modelle, die einen NDVI-Kanal beinhalten. Letztere zeigen insgesamt die schwächste Performance, was auf eine eingeschränkte Nützlichkeit des NDVI-Kanals für diese Aufgabe hindeutet.

Vergleicht man die Ergebnisse mit denen auf dem Validierungsdatensatz (vgl. Abbildung 5.7), so zeigen sich insgesamt größere Schwankungen zwischen den Quantilen. Während beim Validierungsdatensatz die Ergebnisse der meisten Modelle eng beieinanderliegen und lediglich die NDVI-basierten Modelle klare Schwächen zeigen, weisen die Resultate auf dem Testdatensatz eine breitere Spannweite auf. Insbesondere RGIR zeigt hier die größte Quantilspanne und erreicht im Vergleich zu den fünf Hauptmodellen die schwächste Robustheit.

ist auch
eigentlich anal-
yse

Tabelle 5.2 fasst die Folds pro höchsten mAP Wert und Modell zusammen. Auffällig ist, dass die meisten roten Punkte im Bereich des Medians der Modelle liegen und lediglich das RGIR-Modell eine Übereinstimmung zwischen dem besten Fold im Testdatensatz und dem besten Fold im Validierungsdatensatz aufweist. Während jeder Fold jeweils ein- bis zweimal als bestes Validierungsmode auf dem Validierungsdatensatz abschnitt, konnten nur Fold 2 und Fold 3 die Spitzenposition des besten Validierungsmode auf dem Testdatensatz erreichen. Dies deutet darauf hin, dass die Generalisierungsfähigkeit der Modelle stark variieren kann und insbesondere die Übertragbarkeit vom Validierungs- auf den Testdatensatz nicht in allen Fällen konsistent ist.

ist eigenltihc
analyse

Ein detaillierterer Blick auf die Klassenerkennungsleistung wird in den Differenzmatrizen (vgl. Abbildung 5.10) deutlich. Diese veranschaulichen die spezifischen Stärken und Schwächen der einzelnen Modelle im Vergleich zu RGBIR. Während einige Modelle bei bestimmten Klassen deutliche Vorteile aufweisen, zeigen sich bei anderen Klassen systematische Fehlklassifikationen. So können Unterschiede von über 20 % bei einzelnen Klassenbeispielen auftreten, die die Gesamtbewertung einzelner Modelle maßgeblich beeinflussen.

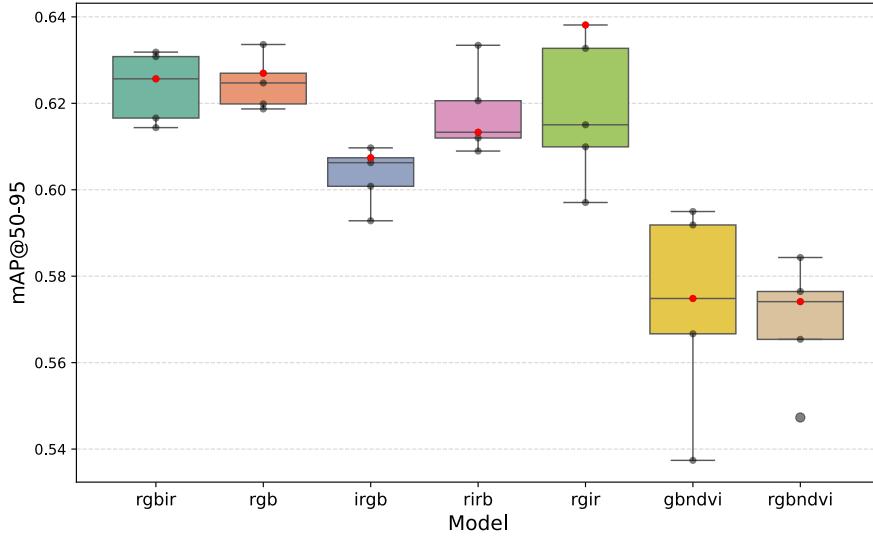


Figure 5.6.: Comparison of mAP50-95 values for different channel permutation models (Best validation model on test dataset). The red dot is the fold, that has the best mAP50-95 at the validation dataset (see fig. 5.7). RGIR performs best, with models using NDVI performing significantly worse than the five main permutations. NDVI may introduce noise. Fluctuations between the quantiles in the main permutations are due to the fact that the channel changes do have an influence. RGBIR has the third-best performance because it contains all bands. RGB and RIRB are better, possibly because too much information also causes noise. Blue does not appear to be important for vehicle detection. (Own representation)

Model	Best mAP Fold (Validation)	Best mAP Fold (Test)
RGBIR	0	2
RGB	0	2
IRGB	1	3
RIRB	4	2
RGIR	3	3
GBNDVI	1	2
RGBNDVI	4	2

Table 5.2.: Best mAP folds for validation and test sets

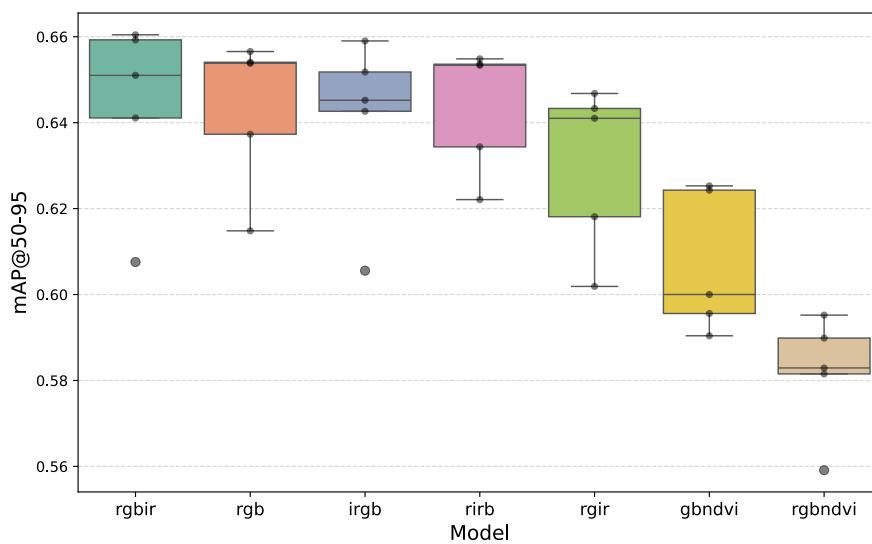


Figure 5.7.: Comparison of mAP50-95 values for different channel permutation models (Best validation model on validation dataset). The RGBIR channel permutation achieves the best performance, followed by the other main permutations, while RGIR performs worst among the five main permutations. Models with NDVI are again significantly worse due to additional noise. The small fluctuations between the five main quantiles indicate robust folds and stable training (Own representation).

5.2.1. Comparison with Confusion Matrices

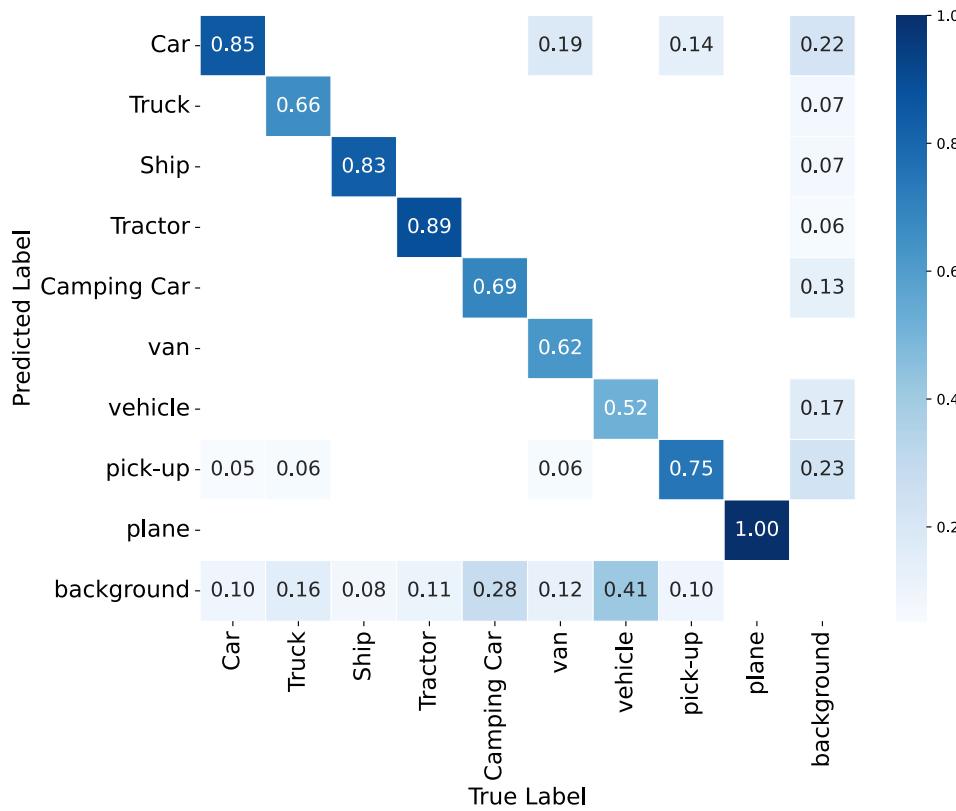


Figure 5.8.: Confusion Matrix for RGBIR Modell (Fold 2). All values falling within the interval $[-0.05, 0.05]$ are excluded from the matrix. The model performs very well on the training data. The class *Plane* is perfectly recognised and classified, while *Car*, *Ship* and *Tractor* achieve excellent results. The generic class *Vehicle* is often confused with the background, presumably due to the high variability of the object types it contains and because it is generic, meaning that there are no clear shapes and colours. The good recognition rate for *Car* illustrates the strong influence of a large amount of training data on classification accuracy (Own representation).

Die in Abbildung 5.8 dargestellte Confusion Matrix für das RGBIR-Modell (Fold 2) zeigt eine sehr starke Leistung. Insbesondere bei der Klasse *Plane* erreicht das Modell eine perfekte Erkennungsrate von 100 %. Auch die Klassen *Car*, *Ship*, *Tractor* weisen eine hervorragende Performance von über 80 % auf. Gute Ergebnisse im Bereich von 60–80 % werden für die Klassen *Truck*, *Camping Car*, *Van* und *Pick Up* erzielt. Die generische Klasse *Vehicle* zeigt mit über 52 % eine akzeptable Erkennungsrate. Problematisch ist hingegen, dass die Klasse *Vehicle* häufig als *Background* fehlklassifiziert wird (über 40 %). Zusätzlich tritt eine erhöhte Fehlklassifikationsrate bei *Camping Car* (28 %) und *Truck* (16 %) auf, während die restlichen Klassen zwischen 8–12 % liegen. Aufgrund dieser starken Gesamtleistung dient das RGBIR-Modell (Fold 2) als Basis für die Differenzmatrizen. Auss-

chlaggebend ist auch, dass in diesem Permutationsdurchlauf alle verfügbaren Bänder unverändert in Rohform enthalten sind.

Zur quantitativen Analyse der Unterschiede zwischen den Vorhersagen verschiedener Modelle werden Differenzmatrizen erzeugt. Dabei werden die Confusion-Matrizen zunächst zeilenweise normalisiert und anschließend elementweise voneinander subtrahiert. Geringfügige Abweichungen im Rahmen von -0.05-0.05% werden dabei maskiert und nicht dargestellt, um diese nicht überzubewerten. Aufgrund der teilweise minderwertigen Labeldarstellung, deren Ausmaß nicht quantifiziert werden kann und die im folgenden Absatz kurz erläutert wird, werden die Unterschiede in der letzten Spalte (Background/Object-Unterscheidung) in der folgenden Analyse nur eingeschränkt berücksichtigt.

Die Abb. 5.9 zeigt, dass die Qualität der Labelannotation teilweise unzureichend ist. So enthält beispielsweise Bild 34, das eine typische Vorstadtumgebung darstellt, keine annotierten Objekte, obwohl Fahrzeuge eindeutig sichtbar sind. Das trainierte Modell erkennt diese jedoch mit hoher Konfidenz. Infolgedessen wird die Trennschärfe zwischen GT-Hintergrund und Objekten beeinträchtigt, was sich insbesondere in der letzten Spalte der Confusion Matrix widerspiegelt.

Zur Interpretation der Differenzmatrizen gilt: Rot markiert eine bessere Performance des RGBIR (F2) Modells, Blau eine bessere Leistung des Vergleichsmodells. Je stärker die Farbintensität, desto größer der Unterschied. Die Unterschiede werden in vier Kategorien eingeteilt: gering (0–5 %), moderat (5–10 %), hoch (10–20 %) und sehr hoch (>20 %) (s. Tabelle 5.3). Die Vergleichsreihenfolge startet mit dem schwächsten Modell und arbeitet sich bis zum besten Modell gemäß den Ergebnissen aus Abbildung 5.6 vor.

Difference (in % from reference)	Category
0 – 5 %	low
5 – 10 %	moderate
10 – 20 %	high
> 20 %	very high

Table 5.3.: Classification of differences relative to the reference

Abbildung 5.10a verdeutlicht deutliche Unterschiede zugunsten von RGBIR. Besonders stark zeigt sich dies bei der korrekten Erkennung von *Ship* (+24 %), *Vehicle* (+16 %), *Plane* (+12 %) und *Tractor* (10 %). Auch bei Verwechslungen zwischen *Van* mit *Car* treten Unterschiede von mehr als 10 % auf. Dagegen weist RGBNDVI weniger hohe Fehlklassifikationen auf, beispielsweise bei *Plane* (-12 %) und *Van* (-23 %), die weniger häufig fälschlich als *Background* erkannt werden. Insgesamt zeigt RGBIR fünf Klassen (*Ship*, *Tractor*, *Vehicle*, *Van*, *Plane*), in denen es mindestens 5 % besser abschneidet, während RGBNDVI keine höhere Klassifikationsgenauigkeit verzeichnen kann.

Beschreibung
Figures länger
machen,
mehr Analyse
weniger "was
sieht man"

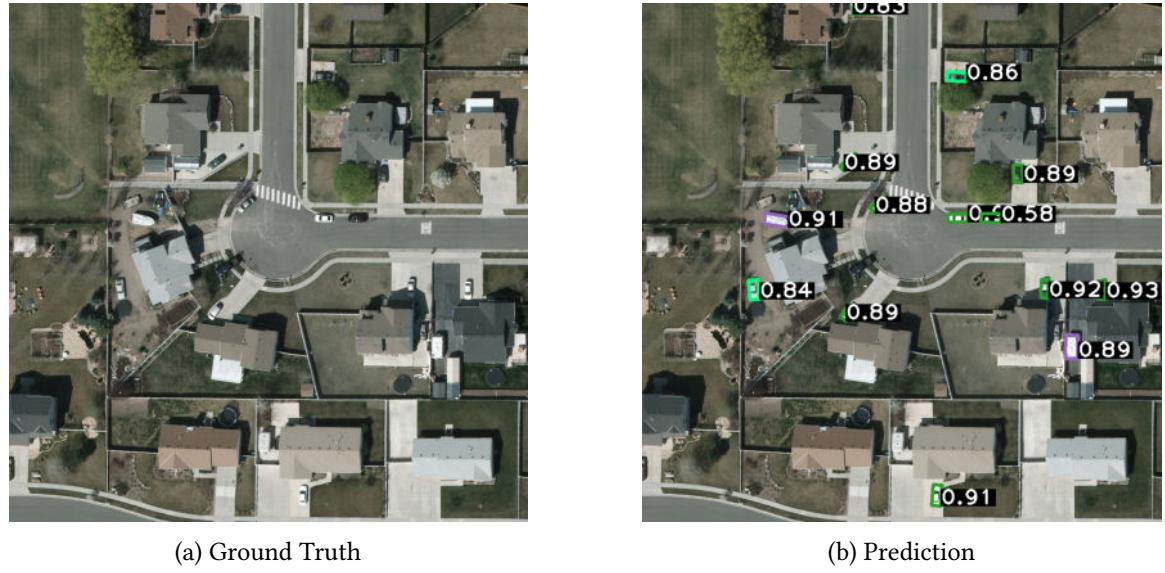


Figure 5.9.: Comparison of image 00000034_co.png shows a discrepancy between ground truth and model prediction: vehicles are recognisable in the GT but not annotated, while the model (RGBIR, F0, (Validation on Validation)) detects and classifies them correctly. This calls into question the validity of GT-based metrics for distinguishing between background and object (Own representation).

In Abbildung 5.10b zeigt sich ein sehr hoher Unterschied bei der Klasse *Tractor*, wo RGBIR deutlich überlegen ist (+22%). Die Verbesserung bei *Ship* beträgt dagegen lediglich +8 %. Auffällig ist, dass *Tractor* (-15%), *Ship* (-12%) und *Van* (-11%) im GBNDVI-Modell weniger häufig fälschlich als *Background* klassifiziert werden. Insgesamt ist RGBIR in vier Klassen (*Ship*, *Tractor*, *Van*, *Vehicle*) um mindestens 5 % besser, während GBNDVI keine höhere Genauigkeit bei einer Klasse aufweist.

Wie Abbildung 5.10c zeigt, bleibt *Ship* eine Klasse, bei der RGBIR besonders stark ist (+21 %). Dennoch treten in beiden Modellen Fehlklassifikationen auf, etwa die Verwechslung von *Ship* mit *Camping Car* (-14%) oder *Background* (-12%). Insgesamt ist RGBIR in drei Klassen (*Ship*, *Camping Car*, *Vehicle*) mindestens 5 % besser, während IRGB keine höhere Genauigkeit bei einer Klasse aufweist.

Abbildung 5.10d zeigt ein gemischtes Bild: RGBIR weist Vorteile bei der Unterscheidung Objekt/*Background* mit den Klassen *Vehicle* und *Camping Car* auf, während RIRB bei *Camping Car* eine moderate Verbesserung bei der *Background*/Objekt Unterscheidung (6 %) erzielt. Auffällig sind die weniger häufigen Fehlklassifikationen im RIRB-Modell, bei denen *Van* (21 %) und *Ship* (12 %) nicht als *Background* erkannt werden. Insgesamt ist RGBIR in zwei Klassen (*Ship*, *Van*) besser, während RIRB in einer Klasse (*Camping Car*) eine höhere Genauigkeit aufweist.

5.2. Permutation Experiments

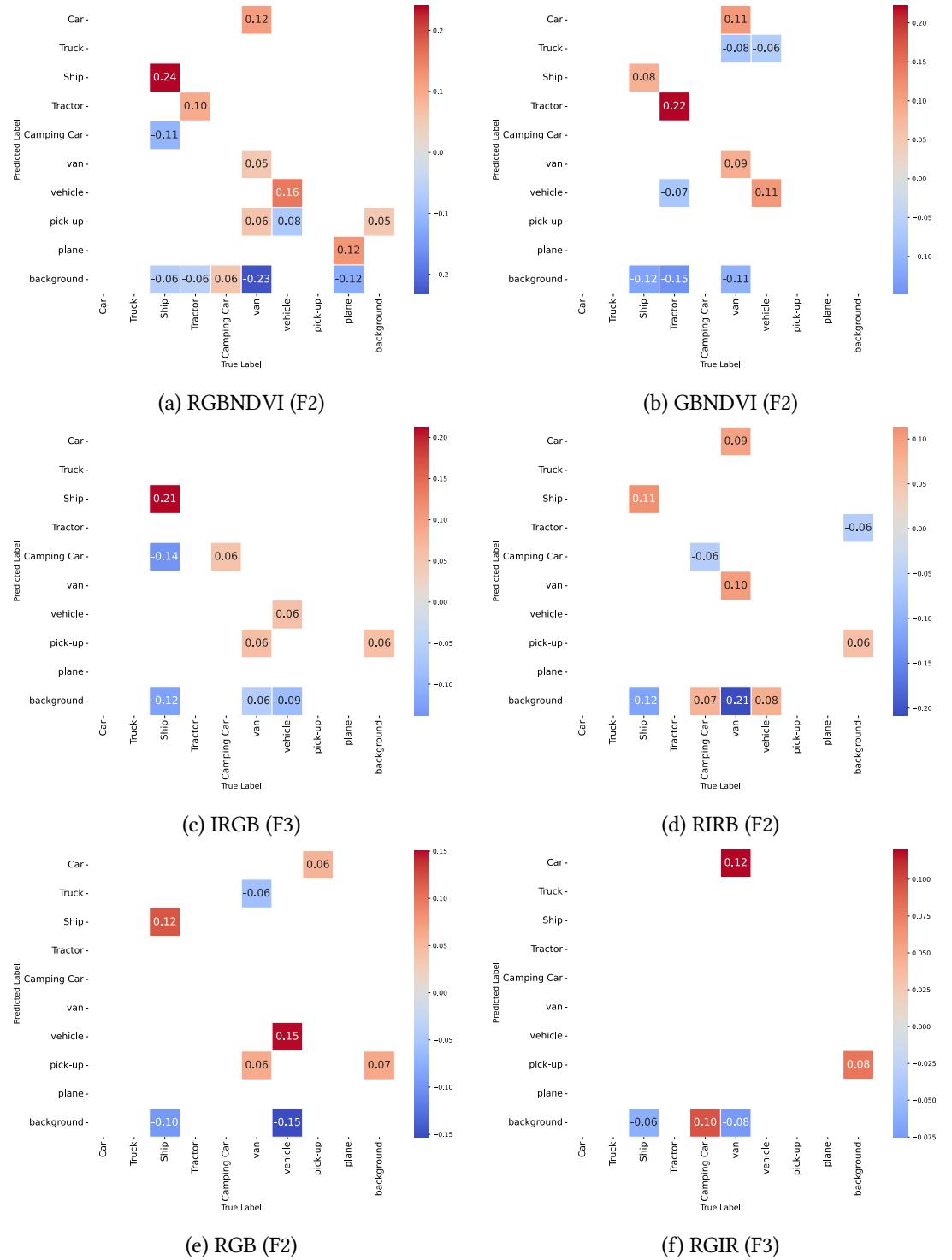


Figure 5.10.: Difference Matrices for all models compared against RGBIR (Fold 2). Red indicates better performance for RGBIR, blue for the respective comparison model. All values falling within the interval $[-0.05, 0.05]$ are excluded from the matrices. Clear differences exist between channel permutations. *Ship* and *Tractor* are best recognised by RGBIR, while RGIR shows minimal differences, suggesting the blue channel is less relevant. Object-background distinction remains challenging for all models, with RGIR yielding the most stable results (own representation).

In Abbildung 5.10e zeigen sich deutliche Unterschiede zugunsten des RGBIR-Modells bei den Klassen *Ship* (12 %) und *Vehicle* (15 %). Dennoch treten auch hier moderate Fehlklassifikationen auf, etwa die Verwechslung von *Pick Up* mit *Car* (6 %) oder *Van* mit *Pick Up* (6 %). Während RGB *Vehicle* (15 %) und *Ship* (10 %) weniger häufig als *Background* verkennt, klassifiziert RGBIR den *Background* häufiger fälschlich als *Pick Up* (7 %). Insgesamt ist RGBIR in zwei Klassen (*Ship*, *Vehicle*) überlegen.

Der letzte Vergleich in Abbildung 5.10f zwischen RGBIR und RGIR (Fold 3) zeigt nahezu einen Gleichstand. Während RGBIR *Van* häufiger mit *Car* verwechselt (12%) und *Camping Car* öfter als *Background* klassifiziert (10%), ist RGIR bei *Ship* (6%) und *Van* (8%) moderat besser. Insgesamt zeigen beide Modelle eine vergleichbare Leistung, ohne dass eines der beiden eine höhere Klassifikationsgenauigkeit bei einer Klasse erzielt.

Abbildung 5.11 verdeutlicht beispielhaft die Erkennungsleistung der verschiedenen Modelle für die im VEDAI Dataset vorhandenen Klassen. Besonders deutlich wird dabei die herausragende Klassifikationsleistung aller Modelle für die Klasse *Car*. Ähnlich verhält es sich bei der Klasse *Truck*, wobei das IRGB-Modell im gezeigten Beispiel die Objekte teilweise sowohl als *Truck* als auch als *Vehicle* klassifiziert. Ebenfalls sehr gute Ergebnisse zeigen die Modelle für die Klassen *Camping Car*, *Tractor*, *Plane*, *Pick Up* und *Van*, wobei beim GBNDVI-Modell eine Verwechslung von *Van* mit *Car* auftritt, jedoch mit geringer Konfidenz. Die Klasse *Ship* wird von fast allen Modellen als *Camping Car* mit geringer Konfidenz fehlklassifiziert; bei RGBIR und IRGB erfolgt zudem eine Doppelklassifikation als *Ship* und *Camping Car*. Lediglich das RGBNDVI-Modell erkennt *Ship* korrekt. Hervorstechend ist, dass die Klasse *Vehicle* von keinem der Modelle zuverlässig erkannt wird.

	Car	Truck	Camping Car	Tractor	Plane	Ship	Vehicle	Pick-Up	Van
GT									
rgbir									
rgb									
irgb									
rirb									
rgir									
gbndvi									
rgbndvi									

Figure 5.11.: Examples of different classes for the object recognition performance of different permutation models. The numbers displayed on the bounding boxes indicate the confidence of the model in the classification of the respective box. For Color Scheme explanation see Tab. A.1. The colour differences between the individual images result from the channel order of CV2 and merely represent display deviations. The class *Car* is consistently recognised by all models with high confidence, which highlights the influence of large training datasets more strongly than changes in spectral information. *Truck* is also reliably recognised due to its distinctive shape, although IRGB occasionally leads to confusion with *Ship* due to double classifications. The generic class *Vehicle* is not correctly classified by any model. Other classes are reliably recognised by all models. A confusion of *Van* with *Car* occurs in GBNDVI, but with low confidence (Own representation).

5.3. Ablation Studies

Die Boxplots in Fig. 5.12 der 6-fold Cross-Validation zeigen, dass die Modelle für die Kanäle Red, Green, Blue und IR im Bereich von $mAP50-95 = 0,52-0,54$ auf dem Testdatensatz liegen. Ein einzelner Wert im IR-Modell erreicht 0,5777, liegt damit jedoch deutlich außerhalb des zentralen Bereichs der anderen Folds und ist als Ausreißer zu interpretieren. Insgesamt lassen sich zwischen den Modellen Red, Green, Blue und IR keine signifikanten Unterschiede feststellen, während NDVI mit einem Median von etwa 0,44 signifikant schlechter abschneidet. Ähnliche Ergebnisse sind in Fig. 5.13 für die Performance auf den Validierungsdaten zu sehen.

Die Auswertung der Ergebnisse in Tab. 5.4 zeigt außerdem, dass insbesondere Fold 4 und Fold 2 die besten Resultate auf dem Validierungsdatensatz erzielen konnten. Auf dem Testdatensatz hingegen erreichen die Folds 0, 2 und 3 die besten Leistungen.

Insgesamt weist das IR-Modell die höchste Performance auf und erzielt den höchsten mAP -Wert im Vergleich zu den anderen untersuchten Varianten.

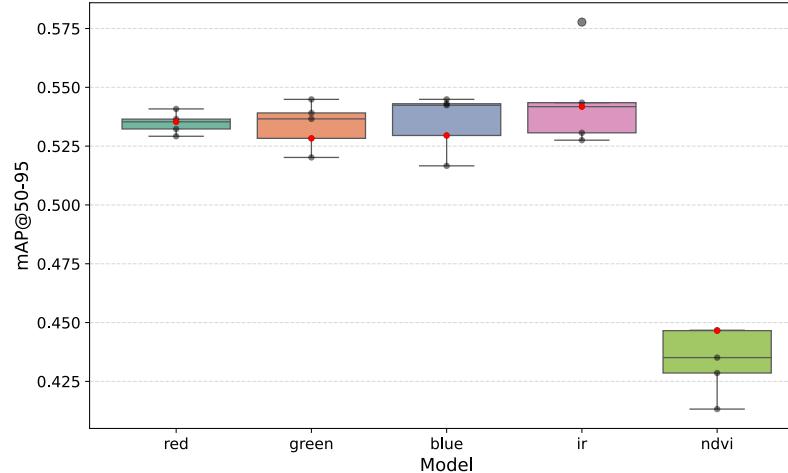


Figure 5.12.: Comparison of $mAP50-95$ values for channels: red, green, blue, ndvi (Best validation model on test dataset). The red dot is the fold, that has the best $mAP50-95$ at the validation dataset (see fig. 5.13). Minimal fluctuations in quantiles between individual channels indicate that a single channel is fundamentally suitable for object recognition, but that significantly better results can be achieved in combination with other channels. The grey value gradients of the individual channels are similar (with the exception of NDVI), although models with NDVI perform significantly worse, presumably due to greater gradations in the grey values (Own representation).

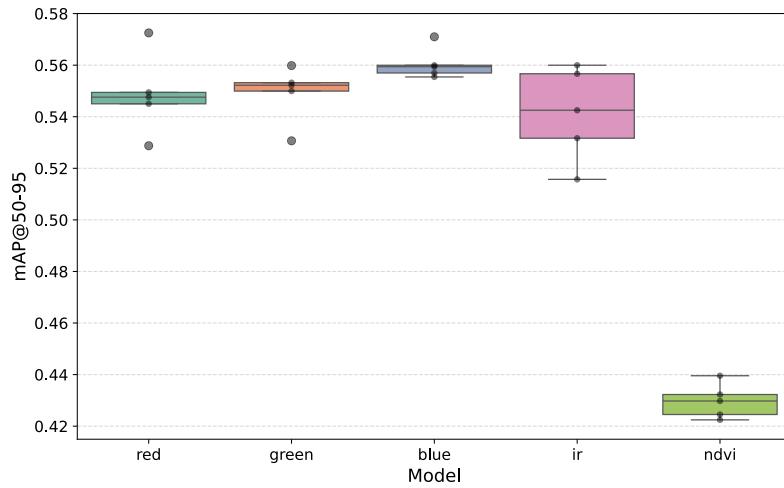


Figure 5.13.: Comparison of mAP₅₀₋₉₅ values for channels: red, green, blue, ndvi (Best validation model on validation dataset). The quantile fluctuations of the R, G and B channels are very low, as the visible spectral colours probably contain similar information individually. The IR channel shows significantly broader quantiles, presumably due to stronger grey value fluctuations. NDVI perform significantly worse due to the pronounced gradations in the grey values. (Own representation)

Im Anschluss an die Analyse der Permutations-Experimente (vgl. Abschnitt 5.2.1) wird nun die Klassifikationsleistung der Modelle unter Verwendung der Confusion Matrix des IR-Modells als Referenz untersucht (siehe Abbildung 5.14a). Die Auswertung erfolgt analog zur Kategorisierung der Unterschiede in Tabelle 5.3.

Das IR-Modell zeigt eine perfekte Erkennungsrate von 100 % bei der Klasse *Plane*. Sehr hohe Erkennungsraten von über 80 % konnten zudem für die Klassen *Car* und *Tractor* erzielt werden.

Eine gute Performance mit Erkennungsraten zwischen 60 % und 80 % wurde für die Klassen *Truck*, *Ship*, *Camping Car* sowie *Pick Up* erreicht. Demgegenüber liegen die Erkennungsraten für die Klassen *Van* und *Vehicle* im Bereich von 40–60 %, was lediglich eine moderate Klassifikationsleistung darstellt.

Auffällig ist insbesondere die fehlerhafte Klassifikation der Klasse *Vehicle*, die in 45 % der Fälle dem *Background* zugeordnet wurde. Darüber hinaus treten erhöhte Fehlklassifikationen in mehreren weiteren Kategorien auf: *Pick Up* (17%), *Ship* (18%), *Camping Car* (28 %) sowie *Van* (39 %) weisen besonders hohe Raten an Fehlzuordnungen auf. Diese Ergebnisse verdeutlichen die spezifischen Schwächen des IR-Modells im Vergleich zu den in den Permutations-Experimenten untersuchten Varianten.

5. RESULTS

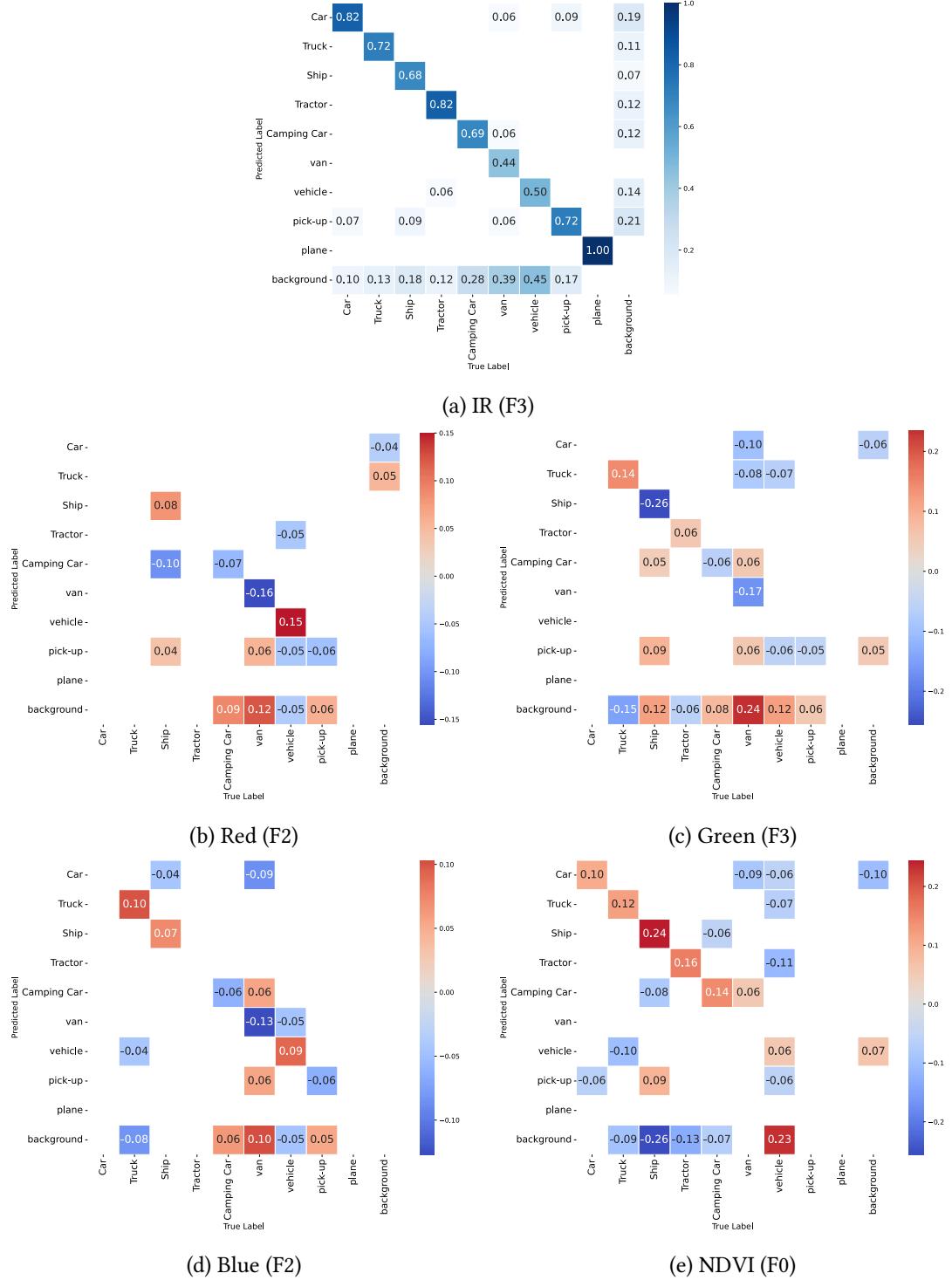


Figure 5.14.: Confusion Matrix for IR Model at Fold 3 and the Difference Matrices compared to this Model. Red indicates better performance for IR, blue for the respective comparison model. All values falling within the interval $[-0.05, 0.05]$ are excluded from the matrices. IR generally outperforms other channels, indicating that its additional spectral information improves feature discrimination. High performance for *Car*, *Tractor*, and *Pick Up* reflects the influence of abundant training data, while persistent confusion for the generic *Vehicle* class highlights limitations in distinguishing objects without distinctive shape or spectral cues (own representation).

Die Analyse der Differenzmatrizen verdeutlicht die jeweilige Leistung einzelner Kanäle im Vergleich zum IR-Referenzmodell (vgl. Abbildung 5.14a).

Abbildung 5.14b zeigt, dass IR eine deutlich höhere Leistung bei der Erkennung der Klasse *Vehicle* im Vergleich zum reinen roten Kanal aufweist (15%). Für die Klasse *Van* erweist sich hingegen der rote Kanal als vorteilhaft, da er hier eine um 16 % höhere Erkennungsrate erzielt. Das IR-Modell zeigt dagegen Vorteile bei der Unterscheidung Objekt/*Background* von *Van* (12 %), *Camping Car* (9 %) sowie moderat bei *Pick Up* (6 %).

In Abbildung 5.14c wird der Vergleich mit dem grünen Kanal dargestellt. Hier erzielt der grüne Kanal eine deutlich höhere Leistung bei der Erkennung von *Ship* (26 %) sowie eine hohe Verbesserung bei *Van* (17 %). Das IR-Modell ist dagegen bei der Erkennung von *Truck* (14 %) überlegen. Besonders auffällig ist die Unterscheidung von Objekten gegenüber dem Hintergrund: Hier weist IR bei *Van* mit 24 % eine deutlich bessere Leistung auf. Auch bei *Ship* und *Vehicle* erreicht IR eine hohe Präzision, während bei *Camping Car* (8 %) und *Pick Up* (6 %) moderate Vorteile bei der Unterscheidung Objekt/*Background* bestehen. Demgegenüber zeigt der grüne Kanal bessere Resultate bei *Truck* (15 %) und moderat bei *Tractor* (6 %). Bei der Klasse *Car* ergibt sich kein signifikanter Unterschied. Insgesamt ist IR in zwei Klassen (*Truck*, *Tractor*) mindestens 5 % besser, während der grüne Kanal in drei anderen Klassen (*Van*, *Camping Car*, *Pick Up*) Vorteile aufweist.

Abbildung 5.14d stellt den Vergleich mit dem blauen Kanal dar. Dieser zeigt moderate Vorteile bei *Camping Car* (6 %) und *Pick Up* (6 %) sowie eine deutliche Verbesserung bei *Van* (13 %). Das IR-Modell ist hingegen bei *Truck* (10 %) sowie moderat bei *Ship* (7 %) und *Vehicle* (9 %) überlegen. Zudem gelingt es IR, *Van* besser vom *Background* zu unterscheiden (10 %) und moderate Vorteile bei dieser Unterscheidung bei *Camping Car* (6 %) sowie *Pick Up* (5 %) zu erzielen. Der blaue Kanal zeigt hingegen moderate Verbesserungen bei der Unterscheidung von *Truck* (8 %) und *Vehicle* (5 %) gegenüber dem *Background*. Insgesamt ist IR in drei Klassen (*Camping Car*, *Van*, *Pick Up*) klar überlegen, während der blaue Kanal in zwei Klassen (*Truck*, *Vehicle*) Vorteile aufweist.

Im Vergleich mit NDVI weist IR eine höhere Präzision bei den Klassen *Car* (10 %), *Truck* (12 %), *Tractor* (16 %) sowie *Camping Car* (14 %) auf und ist insbesondere bei *Ship* (24 %) deutlich exakter. Auch bei *Vehicle* erreicht IR eine moderate Verbesserung (6 %). NDVI zeigt dagegen Vorteile bei der Unterscheidung von Hintergrund und Objekt, insbesondere moderat bei *Truck* (9 %) und *Camping Car* (7 %), sowie eine höhere Präzision bei *Ship* (26 %) und *Tractor* (13 %). Insgesamt ist IR in sechs Klassen (*Car*, *Truck*, *Ship*, *Tractor*, *Camping Car*, *Vehicle*) mindestens 5 % besser als NDVI, während NDVI lediglich in vier Fällen Vorteile bei der Objekt-*Background*-Unterscheidung aufweist.

Zusammenfassend lässt sich festhalten, dass sämtliche Modelle – mit Ausnahme des NDVI-Modells – eine hohe Klassifikationsleistung in der Kategorie *Car* aufweisen, was sich auch in den Beispielabbildungen deutlich widerspiegelt (vgl. Abbildung 5.15). Gleichzeitig zeigen sowohl die Differenz-

matrizen als auch die Beispielabbildungen, dass Fehlklassifikationen zwischen Hintergrund und Objekt weiterhin in erheblichem Maße auftreten, insbesondere innerhalb der Klasse *Vehicle* (vgl. Confusion Matrix 5.14a). Auffällig ist hierbei der IR-Kanal, bei den in Abbildung 5.15 dargestellten Beispielen eine besonders hohe Verwechslungsrate zwischen Objekten und Hintergrund aufweist, während in den übrigen Kanälen insgesamt mehr Objekte korrekt erkannt werden. Darüber hinaus wird die Verwechslung der Klasse *Van* mit *Car* im Red-Modell in der Differenzmatrix 5.14c bestätigt. Im Vergleich zum Referenzmodell (6 %, vgl. Abbildung 5.14a) ist aufgrund dieser Verwechslungsrate weder eine Leistungssteigerung noch eine Verschlechterung festzustellen. Das NDVI-Modell weist hingegen eine deutlich reduzierte Leistungsfähigkeit auf: In Abbildung 5.15 wird sichtbar, dass – mit Ausnahme der korrekten Klassifikation von *Van* sowie der Fehlklassifikation von *Truck* als *Vehicle* – keine zutreffenden Ergebnisse erzielt wurden. Sämtliche weiteren Klassen wurden fälschlicherweise dem Hintergrund zugeordnet.

Channel	Best mAP Fold (Validation)	Best mAP Fold (Test)
red	4	2 (0.5408)
green	4	3 (0.5449)
blue	4	2 (0.5449)
ir	4	0 (0.5777)
ndvi	2	0 (0.4467)

Table 5.4.: Best fold per model on validation and test sets, with test mAP@50-95 values

	Car	Truck	Camping Car	Tractor	Plane	Ship	Vehicle	Pick-Up	Van
GT									
IR									
Red									
Green									
Blue									
NDVI									

Figure 5.15.: Examples of different classes for the object recognition performance of individual channels in the ablation study. The numbers displayed on the bounding boxes indicate the confidence of the model in the classification of the respective box. For Color Scheme explanation see Tab. A.1. The classes *Car* and *Truck* are well recognised across all channels, with the exception of NDVI, where *Truck* is misclassified as *Vehicle* with low confidence. IR shows significant difficulties in separating objects from the background for *Camping Car*, *Tractor* and *Pick Up*. *Tractor* is correctly recognised with high confidence by three models. *Ship* is misclassified either as background or as *Camping Car* with low confidence. The Red model confuses *Van* with *Car* because the shape is similar and colour is irrelevant due to the greyscale. Overall, the results show that the shape of the objects is crucial for classification, especially for *Car*, *Pick Up* and *Van*. *Plane* is also correctly recognised by all models except NDVI, presumably due to its characteristic shape.

6 | Discussion

Die Experimente zur Verwendung von Axis-Aligned Bounding Boxes (ABB) im Vergleich zu Oriented Bounding Box (OBB) zeigen interessante Ergebnisse. OBB verlaufen exakter entlang der Objektgrenzen und erfassen weniger umliegendes Areal, was theoretisch zu einer höheren Genauigkeit bei der Objekterfassung führen sollte. Dies zeigt sich in den geringeren Flächeninhalten der Bounding Boxes. Überraschenderweise zeigt jedoch das ABB-Modell eine bessere Performance in Bezug auf die mean Average Precision (mAP). Eine mögliche Erklärung liegt darin, dass kleine Abweichungen in der Orientierung der OBB die Überlappung mit der Ground Truth stark reduzieren können. Selbst minimale Winkelabweichungen führen zu deutlich niedrigeren Intersection-over-Union-Werten, wodurch die mAP der OBB-Modelle sinkt. ABB bietet somit eine stabilere Performance, während OBB die Bounding Boxes genauer an das Objekt anpasst und damit Rechenressourcen effizienter nutzen kann.

Die Analyse der Kanalpermutationen zeigt, dass die Wahl der Spektralkanäle einen signifikanten Einfluss auf die Objekterkennungsleistung hat. Modelle, die den IR-Kanal in Kombination mit RGB nutzen, erzielen die höchste mAP (0,59–0,63), während der blaue Kanal offenbar wenig zur Fahrzeugdetektion beiträgt. Dies kann möglicherweise daran liegen, dass der blaue Spektralbereich (420–490 nm) im Vergleich zu Rot (650–780 nm), Grün (490–575 nm) und Infrarot (>800 nm) nur begrenzte Informationen liefert [84]. Der NDVI-Kanal hingegen reduziert die Leistung der Modelle, vermutlich aufgrund zusätzlicher Störungen im Datensatz. Fahrzeuge sind auch ohne NDVI gut erkennbar, da die Unterscheidung von Vegetation keinen entscheidenden Vorteil für die Detektion bietet und stattdessen zu Verwirrung führt.

Die Untersuchung der Primärpermutationen zeigt, dass das IRGB-Modell die schlechteste Performance aufweist, was möglicherweise die Bedeutung des roten Kanals für die Fahrzeugdetektion unterstreicht. Modelle wie RGBIR profitieren von der Kombination aus Echtfarbkanälen und IR, während NDVI die Erkennungsleistung reduziert. Alle Modelle zeigen Schwierigkeiten bei der Trennung von Hintergrund und Objekten, insbesondere bei den Klassen *Ship* und *Van*. Flugzeuge werden hingegen zuverlässig erkannt, vermutlich aufgrund ihrer charakteristischen Form und eindeutigen Umgebung, wie beispielsweise Flughäfen oder Wasserflächen.

Die Ablationsstudien bestätigen, dass einzelne Kanäle nur begrenzten Einfluss auf die Erkennungsleistung haben. Erst die Kombination mehrerer Kanäle führt zu einer deutlichen Verbesserung der mAP. Einzelkanäle erreichen Werte unter 0,55, während Kombinationen wie RGIR Werte zwischen 0,59 und 0,63 erzielen. Der NDVI-Kanal liefert mit 0,42–0,44 die schlechteste Performance, was nicht durch die radiometrische Auflösung (8 bit) erklärt werden kann. Fahrzeuge der Klasse *Car* werden robust erkannt, selbst ohne Kanalkombination, vermutlich aufgrund der hohen Anzahl an Trainingsdaten und ihrer charakteristischen Form. Die größten Unterschiede treten bei Hintergrundobjekten auf, wobei IR die besten Ergebnisse liefert und Kombinationen der Kanäle Verbesserungen ermöglichen. Schwierigkeiten bestehen insbesondere bei generischen Klassen wie *Vehicle* und bei Verwechslungen zwischen *Ship* und *Camping Car*, da die Form oder teilweise Abdeckung der Objekte die Unterscheidung erschwert. Die Unterscheidung einzelner Fahrzeugtypen abseits von *Car* bleibt aufgrund begrenzter Daten eine Herausforderung.

Der räumliche Kontext der Objekte beeinflusst die Detektion ebenfalls maßgeblich. Fahrzeuge in strukturierten Vorstadtszenen lassen sich vergleichsweise einfach erkennen, während Fahrzeuge auf Schrottplätzen oder überwucherten Flächen schwer zu detektieren sind. Auch die Farbe spielt eine Rolle: weiße Fahrzeuge auf grauem Asphalt werden leicht erkannt, während graue oder schwarze Fahrzeuge schwerer vom Hintergrund zu unterscheiden sind. Dies korreliert mit der Verteilung der Fahrzeugfarben in den USA, wobei Weiß (24%) und Schwarz (23%) besonders häufig vorkommen [85].

YOLO-Modelle eignen sich eingeschränkt zur Fahrzeugdetektion, wenn Auflösung und Anzahl der Spektralkanäle ausreichend sind. Des Weiteren müssen die Fahrzeugtypen in Form, Farbe und Größe klar unterscheidbar sein, um Verwechslungen zu vermeiden. Bei hochauflösenden Daten (12,5 cm × 12,5 cm pro Pixel) ist eine zuverlässige Erkennung verschiedener Fahrzeugklassen möglich, insbesondere für Autos, die eine mAP von über 80 % erreichen. Dies ist wahrscheinlich auf die große Anzahl an Trainingsdaten zurückzuführen. OBB entsprechen der Definition von *very small objects*

mehr vielleicht

nach Chen et al. [52] und ermöglichen somit die Detektion kleiner Objekte mit YOLOv9. YOLOv9 kann Ground Truth-Objekte zuverlässig lokalisieren und kategorisieren, was Anwendungen wie die Analyse von Fahrzeugverteilungen über ganze Regionen erlaubt, insbesondere in Kombination mit regelmäßigen Satellitenüberflügen [57], [58]. Solche Analysen von Satellitenbildern chinesischer Krematorien wurden beispielsweise während der COVID-19-Pandemie genutzt, um die offiziellen Daten zur COVID Mortalität der Regierung zu überprüfen, indem die Auslastung der Parkplätze (für Leichenwagen) ausgewertet wurde. Dies konnte in Kombination mit Augenzeugenbefragungen dafür genutzt werden, die offiziellen Zahlen zur Sterblichkeit anzuzweifeln [86].

Die Nutzung multispektraler Kanäle und räumlicher Kontextinformationen bietet mehrere Vorteile. Das Modell kann typische Muster erkennen, wie Parkplätze oder Hafenbereiche, und semantische Informationen aus multispektralen Kanälen liefern, die die Klassifikation robuster gegenüber

Tarnung, Schatten oder Farbvariationen machen. Kanäle wie NDVI oder IR ermöglichen zudem die Bewertung von Objektzustand und Nutzung. Die Kombination mehrerer Kanäle verbessert die Generalisierung auf unterschiedliche Umgebungen wie Stadt, Wald oder Wüste und erhöht somit die Robustheit gegenüber neuen Testgebieten. Über die reine Fahrzeugerkennung hinaus können Kategorien wie Fahrzeugtyp (anhand Form und Größe), Zustand (Farbe, bspw. Rosterkennung) oder Nutzungskontext (Gebietsklassifikation, Auto in dichtbesiedelter urbaner Umgebung ist wahrscheinlicher als Flugzeug) unterschieden werden. Die Kombination mit weiteren Datenquellen, etwa SAR, LiDAR oder Höhenmodellen, ermöglicht Anwendungen wie Verkehrsdichtheanalyse, Infrastrukturzustand oder Nutzungsmuster.

Die Untersuchung zeigt, dass die Einbeziehung zusätzlicher Kanäle in das YOLO-Modell unterschiedliche Auswirkungen auf die Erkennungsleistung hat. Der IR-Kanal führt zu einer messbaren Verbesserung der Performance, während die Nutzung des NDVI als Vegetationsindex die Leistung verschlechtert. Dies deutet darauf hin, dass die Trennung von Vegetation im Kontext der Objekterkennung weniger relevant ist. Vielmehr erweisen sich charakteristische Formen und Objektmerkmale als entscheidend für die Klassifizierung, während die Unterscheidung von Hintergrund und Vegetation eine untergeordnete Rolle spielt.

Ein Vergleich zwischen axis-aligned bounding boxes (ABB) und oriented bounding boxes (OBB) verdeutlicht, dass ABB zwar eine höhere mittlere Genauigkeit (mAP) erreicht, OBB jedoch präzisere Vorhersagen liefert. Letztere schließen weniger irrelevante Hintergrundbereiche ein, was die Rechenaufwände reduziert und die Kapazitätsanforderungen des Deep-Learning-Modells verringert, da es weniger lernen muss, relevante von irrelevanten Bereichen zu unterscheiden.

Hinsichtlich der Wirkung einzelner Kanäle zeigt sich, dass die Kombination mehrerer Kanäle die Leistung deutlich verbessert. Einzelne Kanäle wie Rot, Grün oder Blau erzielen ähnliche Ergebnisse, während der IR-Kanal eine höhere Performance erreicht. Der NDVI-Kanal liefert in diesem Experiment die geringste Leistung. Optimal schneiden Kombinationen aus zwei oder drei der sichtbaren Kanäle zusammen mit dem IR-Kanal ab, beispielsweise RGIR, RIRB, RGB, RGBIR oder IRGB. Diese Ergebnisse unterstreichen die Bedeutung der synergistischen Nutzung mehrerer Spektralkanäle für eine verbesserte Objekterkennung.

drinlassen?
oder löschen?
ist eigentlich
redundant!

7 | Conclusion and Outlook

7.1. Conclusion

Die in dieser Arbeit formulierten Forschungsfragen konnten erfolgreich beantwortet werden. Es zeigte sich, dass zusätzliche Bildkanäle wie Infrarot (IR) die Detektionsleistung insbesondere bei kleinen Objekten verbessern können, während spektrale Indizes wie der NDVI nicht in jedem Fall vorteilhaft sind. Die Experimente verdeutlichen, dass bereits RGB-Daten eine robuste Grundlage für die Erkennung kleiner Fahrzeuge bieten, wenngleich IR-Daten die Genauigkeit weiter erhöhen. Zudem wurde ersichtlich, dass die Verfügbarkeit umfangreicher, qualitativ hochwertiger Trainingsdaten einen größeren Einfluss auf die Modellleistung hat als die Erweiterung der Eingangskanäle um mehr Spektralinformation. Die Untersuchungen bestätigen, dass YOLOv9 für die Detektion kleiner Objekte geeignet ist und architekturelle Weiterentwicklungen gegenüber älteren Versionen wie YOLOv3 zu messbaren Verbesserungen in Klassifikation und Genauigkeit führen.

7.2. Outlook

Zukünftige Arbeiten sollten eine systematische Integration weiterer spektraler Indizes und zusätzlicher Kanäle evaluieren, um deren Einfluss auf die Modellleistung präziser zu bestimmen. Darüber hinaus bietet die Anwendung und Modifikation weiterer YOLO-Varianten sowie der Vergleich mit zweistufigen Detektoren (z. B. R-CNN-basierte Verfahren) ein vielversprechendes Untersuchungsfeld. Ein weiterer Ansatz besteht in der Analyse von Skalierungseffekten, etwa durch die Reduktion der räumlichen Auflösung auf Satellitenniveau, um die Übertragbarkeit auf Rohdaten realer Systeme (z. B. Airbus, 30 cm/Pixel [58]) zu evaluieren. Auch die Rolle objektspezifischer Merkmale, wie etwa die Fahrzeugfarbe, sollte vertieft untersucht werden. Langfristig eröffnet insbesondere die Nutzung multitemporaler Daten neue Perspektiven, da sich damit dynamische Muster und Veränderungen im Raum-Zeit-Kontext (z. B. Pendlerverkehr versus Dauerabstellung) erfassen lassen.

A | Appendix

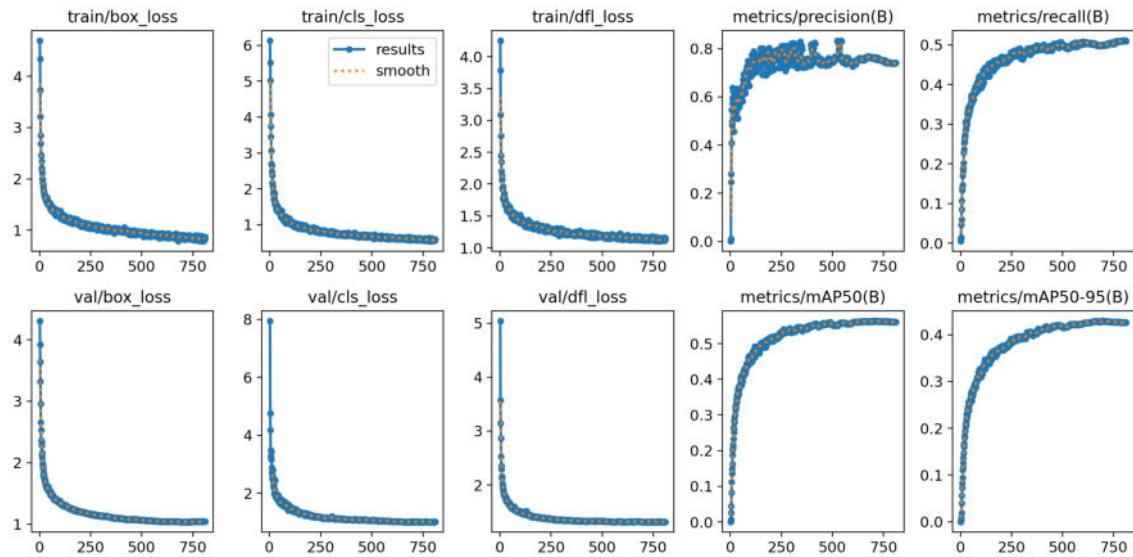


Figure A.1.: Results from DOTA Training

A.1. Full-Sized Images

A. APPENDIX



Figure A.2.: Example Classes for DOTA Dataset



(a) Two Coordinate out of range and smaller 0

(b) Two Coordinates out ouf range and higher 1

Figure A.3.: Example for label coordinates outside of the image (Full-Sized Image)

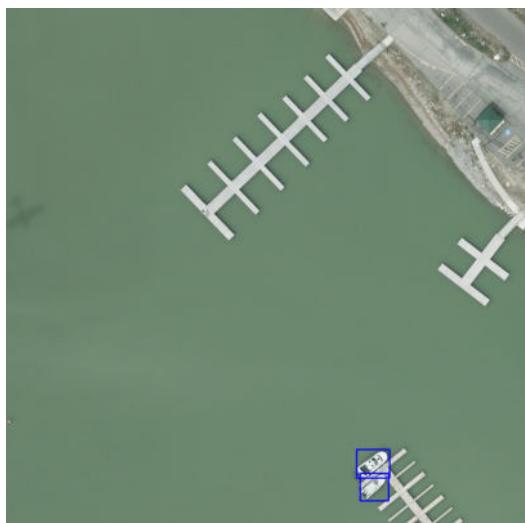
A.1. Full-Sized Images



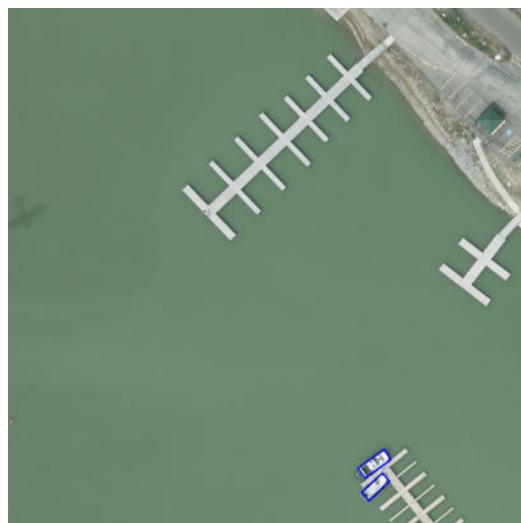
(a) Label: "Truck", abb format, area: $\approx 1378\text{px}$



(b) Label: "Truck", obb format, area: $\approx 967\text{px}$



(c) Labels: "Ship", abb format



(d) Labels: "Ship", obb format

Figure A.4.: Comparison of the bounding box formats of two different object classes (Full-Sized)

A.2. Results

A.2.1. Calculation Bounding Box Area in Percent

The total area of the image (A_{total}) is $1024 \times 1024 = 1,048,576$ pixels².

Calculation of Bounding Box Area in Percent

The percentage area ($F_{\%}$) is calculated using the following formula:

$$F_{\%} = \frac{A_{\text{Box}}}{A_{\text{total}}} \times 100\%$$

OBB (Oriented Bounding Box)

The median area is ≈ 700 pixels.

$$F_{\%,\text{obb}} = \frac{700}{1,048,576} \times 100\% \approx 0.06675\%$$

ABB (Axis-aligned Bounding Box)

The median area is ≈ 1000 pixels.

$$F_{\%,\text{abb}} = \frac{1000}{1,048,576} \times 100\% \approx 0.09536\%$$

A.2.2. Additional Tables

Class	Color
Ground Truth (GT)	[Black]
Car	[Green]
Truck	[Red]
Ship	[Blue]
Tractor	[Orange]
Camping Car	[Purple]
Van	[Brown]
Vehicle	[Pink]
Pick-up	[Cyan]
Plane	[Yellow-Green]

Table A.1.: Classes with corresponding colors, including GT

Vehicle Type	freq_val
Car	0
Pick Up	1
Camping Car	2
Truck	3
Vehicle	4
Tractor	5
Ship	6
Van	7
Plane	8

Table A.2.: Vehicle types and their corresponding freq_val (Frequency Value) assignments

Val-Fold	Train-Folds	#Train-Objects	Plane (Train)	Plane (Val)	CV _{train}	max/min _{train}
0	1,2,3,4	2531	37	4	1.0124	25.14
1	0,2,3,4	2502	30	11	1.0200	30.67
2	0,1,3,4	2542	37	4	1.0142	25.22
3	0,1,2,4	2524	23	18	1.0253	40.61
4	0,1,2,3	2525	37	4	1.0073	24.84

Table A.3.: Classbalance at 6-Fold Cross Validation (Fold 5 is Testdataset)

Table A.4.: Class distribution (train/val/test) for 6-fold CV with fold 5 as test – All classesn

Klasse	Val=0			Val=1			Val=2			Val=3			Val=4		
	Train	Val	Test												
Car	934	229	225	930	239	225	933	226	225	934	225	225	934	240	225
Truck	207	51	50	201	57	50	208	50	50	208	49	50	207	51	50
Ship	117	30	27	116	28	27	115	29	27	114	30	27	117	27	27
Tractor	123	30	30	124	32	30	126	33	30	124	30	30	125	31	30
Campingcar	269	65	63	262	72	63	270	64	63	267	69	63	265	64	63
Van	67	18	16	68	17	16	68	17	16	68	17	16	69	16	16
Vehicle	144	34	33	141	37	33	144	34	33	144	34	33	145	39	33
Pick-Up	637	164	157	640	161	157	644	157	157	641	160	157	642	159	157
Plane	37	4	7	30	11	7	37	4	7	23	18	7	37	4	7

List of Abbreviations

AA2 Area Attention (AA2)-Modul. 11

AAT Anchor-Aided Training. 11

ABB Axis-aligned Bounding Box. 18, 20, 22, 23, 26, 33, 35–40, 71

ANN Artificial Neural Network. 6

AP Average Precision. 14

B Blue. 26

BB Bounding Box. 12, 13, 26

CIT Center for Information Technology. 32

CNN Convolutional Neural Network. 7, 15–17

CPU Central Processing Unit. 32

CSPN Cross-Stage-Partial-Network. 10

CV2 Computer Vision 2. 23, 27

DCNN Deep Convolutional Neural Network. 16

DL Deep Learning. 6, 7, 15, 16

DNN Deep Neural Network. 7

DOTA Dataset for Object Detection in Aerial Images. 17, 20, 21, 25, 33, 34, 61, 72

ELAN Efficient Layer Aggregation Network. 21

FCL Fully Connected Layer. 7, 8, 21

FIR Far-Infrared. 17

G Green. 26

GB Gigabyte. 32

GBNDVI Green, Blue, Normalized Difference Vegetation Index. 20

GELAN Generalized Efficient Layer Aggregation Network. 11, 22

GFLOP Giga Floating Point Per Second. 22

GHz Gigahertz. 32

GPFS General Parallel File System. 32

GPU Graphics Processing Unit. 32

GT Ground Truth. 13, 14, 39, 45, 49, 54, 65, 73

HPC Hyper-Performance-Cluster. 20, 23, 32

ID Identifier. 28

IoU Intersection-over-Union. 12, 13

IR Infrared. 17, 18, 20, 26, 27, 54

IRGB Infrared, Green, Blue. 20

KCV K-fold Cross Validation. 12

mAP Mean Average Precision. 2, 13, 14, 17, 21, 23, 36–38, 40–43, 50, 51, 71, 72

MIR Middle-Infrared. 17

NaN-Value Not A Number Value. 8

NDVI Normalized Difference Vegetation Index. 15, 17, 18, 24, 26, 27

NIR Near-Infrared. 15, 17

NMS Non-Maximum-Suppression. 11

OB_B Oriented Bounding Box. 18, 20, 25, 26, 28, 30, 33, 35–40, 55, 56, 71

PALMA Paralleles Linux-System für Münsteraner Anwender. 20, 23, 25, 32, 33

PB Petabyte. 32

PGI Programmable Gradient Information. 11, 22

PRC Precision-Recall-Curve. 13, 14

R Red. 26, 27

R-CNN Region-Based Convolutional Neural Network. 10

R-ELAN Residual Efficient Layer Aggregation Networks. 11

ReLU Rectified Linear Unit. 8

RGB Red, Green, Blue. 17, 20, 21, 27, 34, 73

RGBIR Red, Green, Blue. Infrared. 20

RGBNDVI Red, Green, Blue, Normalized Difference Vegetation Index. 20

RGIR Red, Green, Infrared. 20, 26, 41

RIRB Red, Infrared, Blue. 20

RNN Recurrent Neural Network. 7

RPN Region Proposal Network. 10

SAE Stacked Auto Encoder. 15, 16

SAR Synthetic Aperture Radar. 15

SLURM Simple Linux Utility for Resource Management. 33

SSD Single Shot Multibox Detector. 10

SVM Support Vector Machine. 7, 16

TFLOP Teraflops. 32

VEDAI Vehicle Detection on Aerial Images (Dataset). 15, 20, 26, 28–30, 34, 35, 48

YAML "YAML Ain't Markup Language" or originally "Yet Another Markup Language". 26

YOLO You Only Look Once. 2, 10, 17, 19–22, 24–26, 28, 30, 33, 35, 37, 40, 56

List of Figures

2.1. Overview over Artificial Intelligence, Machine Learning and Deep Learning. [10]	6
2.2. Underfitting, Overfitting and Optimal at Deep Learning Trainings. [11]	9
2.3. Graphical Visualiziation for IoU. [35]	12
4.1. Flowchart for workflow	19
4.2. Different YOLO Architectures	22
4.3. Example for label coordinates outside of the image	31
5.1. Comparison of the bounding box formats of two different object classes	37
5.2. Comparison of Bounding Box Area in pixels for ABB, OBB, abb in obb	38
5.3. Comparison of mAP50-95 values for ABB and OBB (Best validation model on validation dataset)	38
5.4. aab and obb: Examples of different classes for the object recognition performance	39
5.5. Comparison of mAP50-95 values for ABB and OBB (Best validation model on test dataset)	40
5.6. Comparison of mAP50-95 values for different channel permutation models (Best validation model on test dataset)	42
5.7. Comparison of mAP50-95 values for different channel permutation models (Best validation model on validation dataset)	43
5.8. Confusion Matrix for RGBIR Modell (Fold 2)	44
5.9. Comparison of image 00000034_co.png shows a discrepancy between ground truth and model prediction: vehicles are recognisable in the GT but not annotated, while the model (RGBIR, F0, (Validation on Validation)) detects and classifies them correctly. This calls into question the validity of GT-based metrics for distinguishing between background and object (Own representation).	46
5.10. Difference Matrices compared to RGBIR (Fold 2)	47
5.11. Permutation Experiments: Examples of different classes for the object recognition performance	49
5.12. Comparison of mAP50-95 values for channels: red, green, blue, ndvi (Best validation model on test dataset)	50

LIST OF FIGURES

5.13. Comparison of mAP50-95 values for channels: red, green, blue, ndvi (Best validation model on validation dataset)	51
5.14. Confusion Matrix for IR Model at Fold 3 and the Difference Matrices compared to this Model	52
5.15. Ablation Studies: Examples of different classes for the object recognition performance	55
A.1. Results from DOTA Training	63
A.2. Example Classes for DOTA Dataset	64
A.3. Example for label coordinates outside of the image (Full-Sized Image)	64
A.4. Comparison of the bounding box formats of two different object classes (Full-Sized)	65

List of Tables

4.1.	Channel permutations represented by included channels (x = channel present)	20
4.2.	Comparison of YOLOv9 model variants in terms of accuracy and computational complexity at an input resolution of 640×640 pixels. [29], [73]	23
4.3.	Class distribution across folds.	32
4.4.	System overview PALMA	33
4.5.	Key training hyperparameters at the RGB Permutation (Fold 4)	34
5.1.	Best fold per model at Bounding Box format	36
5.2.	Best mAP folds for validation and test sets	42
5.3.	Classification of differences relative to the reference	45
5.4.	Best fold per model on validation and test sets, with test mAP@50-95 values . . .	54
A.1.	Classes with corresponding colors, including GT	67
A.2.	Vehicle types and their corresponding freq_val (Frequency Value) assignments . .	67
A.3.	Classbalance at 6-Fold Cross Validation (Fold 5 is Testdataset)	67
A.4.	Class distribution (train/val/test) for 6-fold CV with fold 5 as test – All classesn .	68

Bibliography

- [1] C. Knoth and E. Pebesma, “Detecting dwelling destruction in darfur through object-based change analysis of very high-resolution imagery,” *International Journal of Remote Sensing*, vol. 38, pp. 273–295, 1 Jan. 2017, issn: 13665901. doi: 10.1080/01431161.2016.1266105. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/01431161.2016.1266105> (cit. on p. 1).
- [2] J. L. Balzer, “Deep learning zur objektdetektion auf hochauflösenden satellitenbildern,” Erstgutachter: Marlon Becker, Zweitgutachter: Prof. Dr. Benjamin Risse, Bachelorarbeit, Westfälische Wilhelms-Universität Münster, 2022 (cit. on pp. 2, 17).
- [3] V. Wiley and T. Lucas, “Computer vision and image processing: A paper review,” *International Journal of Artificial Intelligence Research*, vol. 2, p. 22, 1 Jun. 2018. doi: 10.29099/ijair.v2i1.42 (cit. on pp. 5, 15).
- [4] S. Matiacevich, D. C. Cofré, P. Silva, J. Enrione, and F. Osorio, “Quality parameters of six cultivars of blueberry using computer vision,” *International Journal of Food Science*, vol. 2013, 2013, issn: 23145765. doi: 10.1155/2013/419535 (cit. on p. 5).
- [5] D. Mery, F. Pedreschi, and A. Soto, “Automated design of a computer vision system for visual food quality evaluation,” *Food and Bioprocess Technology*, vol. 6, pp. 2093–2108, 8 Aug. 2013, issn: 19355130. doi: 10.1007/s11947-012-0934-2 (cit. on p. 5).
- [6] S. H. Shetty, S. Shetty, C. Singh, and A. Rao, “Supervised machine learning: Algorithms and applications,” *Fundamentals and Methods of Machine and Deep Learning: Algorithms, Tools, and Applications*, pp. 1–16, Jan. 2022. doi: 10.1002/9781119821908.CH1 (cit. on p. 5).
- [7] P. Fischer, “Algorithmisches lernen,” 1999. doi: 10.1007/978-3-663-11956-2. [Online]. Available: <http://link.springer.com/10.1007/978-3-663-11956-2> (cit. on p. 6).
- [8] U. Braga-Neto, “Fundamentals of pattern recognition and machine learning,” *Fundamentals of Pattern Recognition and Machine Learning*, pp. 1–357, Jan. 2020. doi: 10.1007/978-3-030-27656-0/COVER (cit. on p. 6).
- [9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org> (cit. on pp. 6–8, 13).

- [10] L. Alzubaidi et al., “Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions,” *Journal of Big Data*, vol. 8, p. 53, 1 Mar. 2021, ISSN: 2196-1115. doi: 10.1186/s40537-021-00444-8 (cit. on pp. 6–9).
- [11] S. Zivkovic, *018 pytorch - popular techniques to prevent the overfitting in a neural networks*, Nov. 8, 2021. Accessed: Aug. 27, 2025. [Online]. Available: <https://datahacker.rs/018-pytorch-popular-techniques-to-prevent-the-overfitting-in-a-neural-networks/> (cit. on p. 9).
- [12] P. Soviany and R. T. Ionescu, “Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction,” in *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE, Sep. 2018, pp. 209–214, ISBN: 978-1-7281-0625-0. doi: 10.1109/SYNASC.2018.00041 (cit. on p. 10).
- [13] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016. arXiv: 1506.01497 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1506.01497> (cit. on p. 10).
- [14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2018. arXiv: 1703.06870 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1703.06870> (cit. on p. 10).
- [15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1506.02640> (cit. on p. 10).
- [16] W. Liu et al., “Ssd: Single shot multibox detector,” in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37, ISBN: 9783319464480. doi: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2 (cit. on p. 10).
- [17] R. Sapkota et al., “Yolo advances to its genesis: A decadal and comprehensive review of the you only look once (yolo) series,” *Artificial Intelligence Review*, vol. 58, 9 Sep. 2025, ISSN: 15737462. doi: 10.1007/s10462-025-11253-3 (cit. on pp. 10, 11).
- [18] R. Li and J. Yang, “Improved yolov2 object detection model,” in *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)*, IEEE, May 2018, pp. 1–6, ISBN: 978-1-5386-6220-5. doi: 10.1109/ICMCS.2018.8525895 (cit. on p. 10).
- [19] H. Nakahara, H. Yonekawa, T. Fujii, and S. Sato, “A lightweight yolov2,” in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ACM, Feb. 2018, pp. 31–40, ISBN: 9781450356145. doi: 10.1145/3174243.3174266 (cit. on p. 10).

- [20] K.-J. Kim, P.-K. Kim, Y.-S. Chung, and D.-H. Choi, “Performance enhancement of yolov3 by adding prediction layers with spatial pyramid pooling for vehicle detection,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, Nov. 2018, pp. 1–6, ISBN: 978-1-5386-9294-3. DOI: 10.1109/AVSS.2018.8639438 (cit. on p. 10).
- [21] U. Nepal and H. Eslamiat, “Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs,” *Sensors*, vol. 22, p. 464, 2 Jan. 2022, ISSN: 1424-8220. DOI: 10.3390/s22020464 (cit. on p. 10).
- [22] M. Sozzi, S. Cantalamessa, A. Cogato, A. Kayad, and F. Marinello, “Automatic bunch detection in white grape varieties using yolov3, yolov4, and yolov5 deep learning algorithms,” *Agronomy*, vol. 12, p. 319, 2 Jan. 2022, ISSN: 2073-4395. DOI: 10.3390/agronomy12020319 (cit. on p. 10).
- [23] N. Mohod, P. Agrawal, and V. Madaan, “Yolov4 vs yolov5: Object detection on surveillance videos,” in 2023, pp. 654–665. DOI: 10.1007/978-3-031-28183-9_46 (cit. on p. 10).
- [24] Ultralytics, *Ultralytics Github*, 2020. Accessed: Aug. 26, 2025. [Online]. Available: <https://github.com/ultralytics> (cit. on p. 11).
- [25] C. Li et al., *Yolov6: A single-stage object detection framework for industrial applications*, 2022. arXiv: 2209.02976 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2209.02976> (cit. on p. 11).
- [26] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2022. arXiv: 2207.02696 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2207.02696> (cit. on p. 11).
- [27] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2023, pp. 7464–7475, ISBN: 979-8-3503-0129-8. DOI: 10.1109/CVPR52729.2023.00721 (cit. on p. 11).
- [28] G. Jocher and A. Chaurasia and J. Qiu, *Ultralytics Github*, 2021. Accessed: Aug. 26, 2025. [Online]. Available: <https://github.com/ultralytics> (cit. on pp. 11, 23).
- [29] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, *Yolov9: Learning what you want to learn using programmable gradient information*, 2024. arXiv: 2402.13616 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2402.13616> (cit. on pp. 11, 23).
- [30] C.-Y. Wang and H.-Y. M. Liao, “YOLOv9: Learning what you want to learn using programmable gradient information,” 2024 (cit. on pp. 11, 22).
- [31] Ultralytics, *Ultralytics*, 2024. Accessed: Aug. 26, 2025. [Online]. Available: <https://docs.ultralytics.com/de/models/yolo11/> (cit. on p. 11).

- [32] Y. Tian, Q. Ye, and D. Doermann, *Yolov12: Attention-centric real-time object detectors*, 2025. arXiv: 2502.12524 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2502.12524> (cit. on p. 11).
- [33] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, *The 'K' in K-fold Cross Validation*. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2012, ISBN: 9782874190490. [Online]. Available: <http://www.i6doc.com/en/livre/?GCOI=28001100967420>. (cit. on p. 12).
- [34] I. K. Nti, O. Nyarko-Boateng, and J. Aning, “Performance of machine learning algorithms with different k values in k-fold crossvalidation,” *International Journal of Information Technology and Computer Science*, vol. 13, pp. 61–71, 6 Dec. 2021, ISSN: 20749007. DOI: 10.5815/ijitcs.2021.06.05 (cit. on p. 12).
- [35] A. Rosebrock, *Intersection over union (iou) for object detection*, Nov. 6, 2016. Accessed: Aug. 26, 2025. [Online]. Available: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/> (cit. on p. 12).
- [36] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, *Generalized intersection over union: A metric and a loss for bounding box regression*, 2019. arXiv: 1902.09630 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1902.09630> (cit. on p. 12).
- [37] P. Z. Ramirez, A. Tonioni, S. Salti, and L. D. Stefano, “Learning across tasks and domains,” Oct. 2019. [Online]. Available: <http://arxiv.org/abs/1904.04744> (cit. on p. 13).
- [38] M. Cordts et al., *The cityscapes dataset for semantic urban scene understanding*, 2016. arXiv: 1604.01685 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1604.01685> (cit. on p. 13).
- [39] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 5122–5130, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.544 (cit. on p. 13).
- [40] T.-Y. Lin et al., *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1405.0312> (cit. on p. 13).
- [41] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2 Jun. 2010, ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4 (cit. on p. 13).
- [42] M. Kristan et al., “The visual object tracking vot2016 challenge results,” in 2016, pp. 777–823. DOI: 10.1007/978-3-319-48881-3_54 (cit. on p. 13).

- [43] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, *Motchallenge 2015: Towards a benchmark for multi-target tracking*, 2015. arXiv: 1504.01942 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1504.01942> (cit. on p. 13).
- [44] Ultralytics, *Intersection over union (iou)*. Accessed: Aug. 26, 2025. [Online]. Available: <https://www.ultralytics.com/glossary/intersection-over-union-iou> (cit. on p. 13).
- [45] O. Rainio, J. Teuho, and R. Klén, “Evaluation metrics and statistical tests for machine learning,” *Scientific Reports*, vol. 14, p. 6086, 1 Mar. 2024, ISSN: 2045-2322. DOI: 10.1038/s41598-024-56706-x (cit. on p. 14).
- [46] X. X. Zhu et al., “Deep learning in remote sensing: A comprehensive review and list of resources,” *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, pp. 8–36, 4 Dec. 2017, ISSN: 2168-6831. DOI: 10.1109/MGRS.2017.2762307 (cit. on pp. 15, 16).
- [47] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, “Deep learning-based classification of hyperspectral data,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, pp. 2094–2107, 6 Jun. 2014, ISSN: 1939-1404. DOI: 10.1109/JSTARS.2014.2329330 (cit. on p. 15).
- [48] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, “Deep learning classification of land cover and crop types using remote sensing data,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, pp. 778–782, 5 May 2017, ISSN: 1545-598X. DOI: 10.1109/LGRS.2017.2681128 (cit. on p. 15).
- [49] M. Wieland, S. Martinis, R. Kiefl, and V. Gstaiger, “Semantic segmentation of water bodies in very high-resolution satellite and aerial images,” *Remote Sensing of Environment*, vol. 287, p. 113 452, Mar. 2023, ISSN: 00344257. DOI: 10.1016/j.rse.2023.113452 (cit. on p. 15).
- [50] S. Razakarivony, F. Jurie, S. Razakarivony, and F. Jurie, “Vehicle detection in aerial imagery : A small target detection benchmark,” *Journal of Visual Communication and Image Representation*, 2015. [Online]. Available: <https://hal.science/hal-01122605v2> (cit. on pp. 15, 20, 21, 28, 31).
- [51] M. J. Khan, H. S. Khan, A. Yousaf, K. Khurshid, and A. Abbas, “Modern trends in hyperspectral image analysis: A review,” *IEEE Access*, vol. 6, pp. 14 118–14 129, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2812999 (cit. on p. 15).
- [52] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao, “R-cnn for small object detection,” in 2017, pp. 214–230. DOI: 10.1007/978-3-319-54193-8_14 (cit. on pp. 15, 35, 58).
- [53] S. Bhagavathy and B. Manjunath, “Modeling and detection of geospatial objects using texture motifs,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, pp. 3706–3715, 12 Dec. 2006, ISSN: 0196-2892. DOI: 10.1109/TGRS.2006.881741 (cit. on p. 16).

- [54] G. Cheng, P. Zhou, and J. Han, “Learning rotation-invariant convolutional neural networks for object detection in vhr optical remote sensing images,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, pp. 7405–7415, 12 Dec. 2016, ISSN: 0196-2892. DOI: 10.1109/TGRS.2016.2601622 (cit. on p. 16).
- [55] X.-L. Chen, H.-M. Zhao, P.-X. Li, and Z.-Y. Yin, “Remote sensing image-based analysis of the relationship between urban heat island and land use/cover changes,” *Remote Sensing of Environment*, vol. 104, pp. 133–146, 2 Sep. 2006, ISSN: 00344257. DOI: 10.1016/j.rse.2005.11.016 (cit. on p. 16).
- [56] L. Eikvil, L. Aurdal, and H. Koren, “Classification-based vehicle detection in high-resolution satellite images,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, pp. 65–72, 1 Jan. 2009, ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2008.09.005 (cit. on pp. 16, 17).
- [57] *Planet Labs - Main*. Accessed: Aug. 22, 2025. [Online]. Available: <https://www.planet.com/products/high-resolution-satellite-imagery/> (cit. on pp. 16, 58).
- [58] *Airbus - Pleiades Neo*. Accessed: Aug. 22, 2025. [Online]. Available: <https://space-solutions.airbus.com/imagery/our-optical-and-radar-satellite-imagery/pleiades-neo/> (cit. on pp. 16, 58, 61).
- [59] P. Liu, K.-K. R. Choo, L. Wang, and F. Huang, “Svm or deep learning? a comparative study on remote sensing image classification,” *Soft Computing*, vol. 21, pp. 7053–7065, 23 Dec. 2017, ISSN: 1432-7643. DOI: 10.1007/s00500-016-2247-2 (cit. on p. 16).
- [60] P. Zhou, G. Cheng, Z. Liu, S. Bu, and X. Hu, “Weakly supervised target detection in remote sensing images based on transferred deep features and negative bootstrapping,” *Multidimensional Systems and Signal Processing*, vol. 27, pp. 925–944, 4 Oct. 2016, ISSN: 0923-6082. DOI: 10.1007/s11045-015-0370-3 (cit. on p. 16).
- [61] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 886–893, 2005. DOI: 10.1109/CVPR.2005.177 (cit. on p. 17).
- [62] L. Zhang, Z. Shi, and J. Wu, “A hierarchical oil tank detector with deep surrounding features for high-resolution optical satellite imagery,” *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, pp. 4895–4909, 10 Oct. 2015, ISSN: 21511535. DOI: 10.1109/JSTARS.2015.2467377 (cit. on p. 17).
- [63] “Convolutional neural network based automatic object detection on aerial images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 13, pp. 740–744, 5 May 2016, ISSN: 1545598X. DOI: 10.1109/LGRS.2016.2542358 (cit. on p. 17).

- [64] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, “Orientation robust object detection in aerial images using deep convolutional neural network,” *Proceedings - International Conference on Image Processing, ICIP*, vol. 2015-December, pp. 3735–3739, Dec. 2015, ISSN: 15224880. doi: 10.1109/ICIP.2015.7351502 (cit. on p. 17).
- [65] K. Takumi, K. Watanabe, Q. Ha, A. Tejero-De-Pablos, Y. Ushiku, and T. Harada, “Multispectral object detection for autonomous vehicles,” in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, ACM, Oct. 2017, pp. 35–43, ISBN: 9781450354165. doi: 10.1145/3126686.3126727 (cit. on p. 17).
- [66] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, “Vehicle detection in satellite images by hybrid deep convolutional neural networks,” *IEEE Geoscience and Remote Sensing Letters*, vol. 11, pp. 1797–1801, 10 Oct. 2014, ISSN: 1545-598X. doi: 10.1109/LGRS.2014.2309695 (cit. on p. 17).
- [67] Q. Jiang, L. Cao, M. Cheng, C. Wang, and J. Li, “Deep neural networks-based vehicle detection in satellite images,” *4th International Symposium on Bioelectronics and Bioinformatics, ISBB 2015*, pp. 184–187, Dec. 2015. doi: 10.1109/ISBB.2015.7344954 (cit. on p. 17).
- [68] Airbus, *Airbus Ship Detection Challenge*, 2018. Accessed: Sep. 23, 2025. [Online]. Available: <https://www.kaggle.com/competitions/airbus-ship-detection/data> (cit. on p. 17).
- [69] Timo Lietmeyer, *Master Thesis - Deep Learning for Small Vehicle Detection and Classification on High-Resolution Multispectral Remote Sensing Imagery*, 2025. Accessed: Sep. 23, 2025. [Online]. Available: https://github.com/Timo123456789/master_thesis (cit. on p. 20).
- [70] *Vehicle detection in aerial imagery (vedai) : A benchmark*. [Online]. Available: <https://downloads.greyc.fr/vedai/> (cit. on p. 20).
- [71] C.-Y. Wang and H.-Y. M. Liao, “Yolov1 to yolov10: The fastest and most accurate real-time object detection systems,” Aug. 2024. [Online]. Available: <http://arxiv.org/abs/2408.09332> (cit. on p. 22).
- [72] Wing Kin Yiu, *YOLOv9u - Github Repository*, Feb. 22, 2024. Accessed: Aug. 4, 2025. [Online]. Available: <https://github.com/WongKinYiu/yolov9/tree/yolov9u?tab=readme-ov-file> (cit. on p. 22).
- [73] n. A., *Ultralytics Github*. Accessed: Sep. 17, 2025. [Online]. Available: <https://docs.ultralytics.com/models/yolov9/#supported-tasks-and-modes> (cit. on p. 23).
- [74] *About - OpenCV*. Accessed: Aug. 4, 2025. [Online]. Available: <https://opencv.org/about/> (cit. on p. 24).
- [75] *Releases - OpenCV*. Accessed: Aug. 4, 2025. [Online]. Available: <https://opencv.org/releases/> (cit. on p. 24).

- [76] *NumPy - About Us*. Accessed: Aug. 4, 2025. [Online]. Available: <https://numpy.org/about/> (cit. on p. 24).
- [77] *NumPy*. Accessed: Aug. 4, 2025. [Online]. Available: <https://numpy.org/> (cit. on p. 24).
- [78] *SciPy - Main*. Accessed: Apr. 8, 2025. [Online]. Available: <https://scipy.org/> (cit. on p. 24).
- [79] *matplotlib - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://matplotlib.org/> (cit. on p. 25).
- [80] *seaborn - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://seaborn.pydata.org/> (cit. on p. 25).
- [81] *pandas - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://pandas.pydata.org/> (cit. on p. 25).
- [82] A. S. Inc., *Uv-an extremely fast python package and project manager, written in rust*. Accessed: Aug. 26, 2025. [Online]. Available: <https://docs.astral.sh/uv/> (cit. on p. 32).
- [83] Center for Information Technology, University of Münster, *Hpc documentation palma ii hardware*, Accessed: 2025-07-30. [Online]. Available: <https://palma.uni-muenster.de/documentation/> (cit. on p. 33).
- [84] Bundesamt fuer Strahlenschutz, *Was versteht man unter Licht?* Accessed: Sep. 18, 2025. [Online]. Available: <https://www.bfs.de/DE/themen/opt/sichtbares-licht/einfuehrung/einfuehrung.html> (cit. on p. 57).
- [85] Nextstar Media Wire and iSeeCars, *These are the most popular car colors in every state*, 2023. Accessed: Sep. 18, 2025. [Online]. Available: <https://www.abc4.com/news/national/these-are-the-most-popular-car-colors-in-every-state/> (cit. on p. 58).
- [86] *Corona in china: Satellitenbilder von krematorien deuten auf deutlich mehr tote hin - der spiegel*, 2023. [Online]. Available: <https://www.spiegel.de/ausland/corona%5C--satellitenbilder%5C--chinesischer%5C--krematorien-deuten-auf%5C--deutlich%5C--mehr%5C--tote-hin%5C--a%5C-c4305852%5C-d092%5C-4e54%5C-a210%5C-c88b820d564c> (cit. on p. 58).

Declaration of Academic Integrity

I hereby confirm that this thesis, entitled

Deep Learning for Small Vehicle Detection and Classification on High-Resolution Multispectral Remote Sensing Imagery

is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited. I am aware that plagiarism is considered an act of deception which can result in sanction in accordance with the examination regulations.

Timo Lietmeyer, Münster, September 23, 2025

I consent to having my thesis cross-checked with other texts to identify possible similarities and to having it stored in a database for this purpose.

I confirm that I have not submitted the following thesis in part or whole as an examination paper before.

Timo Lietmeyer, Münster, September 23, 2025