

Todo list

Some todo	VII
sämtliche Abkürzungen mit glocaries einbinden und verwalten	1
Nochmal neu formulieren?	3
Detaillierte Convolutional Operation noch einfügen	6
mathematische definition einfügen?	11
Ablaufdiagramm einfügen	17
Verweis auf Github Repo noch erwähnen?	17
Bsp. Bild für jede Klasse zeigen (aus Paper nehmen!; Quellenverlinkung für text und bild nicht vergessen)	18
Hinweis: Für eine bessere Anschaulichkeit sollen beispielhafte Bilder für jede Klasse aus der Originalpublikation übernommen und entsprechend mit Quellenangabe referenziert werden.	19
ab jetzt nur für analyse	21
text\todo{Ref zur Formatbeschreibung einfügen}	22
Verweis auf entsprechende Funktion	23
ref einfügen	24
ref einfügen	24
ref einfügen	26
ref funktion dafür	26
ref funktion einfügen	27
Beispielbilder für Randlage einfügen	27

Fold Probleme bei VEDAI erwähnen?	27
tabelle mit folds einfügen; Verteilung beschreiben; dann übergang zu weiteren punkten . .	31
Vergleichsbild einfügen oder verweis	31
Bild für Vergleich der BB Area einfügen (zw. obb, aab aab old)	31
Bild einfügen best val dataset on val data	32
Tabelle einfügen Comparison of best performed Fold by mAP50-95 (s. Defence)	32
Bild einfügen; best val dataset on test data	32
Tabelle einfügen	32
beide confusionsmatrizen einfügen	33
Differenzmatrix RGBIR vs RGIR einfügen	33
Differenzmatrix RGBIR vs RGB einfügen	33
Differenzmatrix RGBIR vs IRGB einfügen	33
Differenzmatrix RGBIR vs RIRB einfügen	34
Differenzmatrix RGBIR vs GBNDBVI und RGBNDVI einfügen	34
Tabelle mit Best Folds Ablation Studies einfügen	34

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

Supervisor:
Sebastian Thiele

First assessor:
Prof. Dr. Benjamin Risse

Second assessor:
Dr. Christian Knoth

Submitted by:
Timo Lietmeyer

Münster, August 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

This document was compiled in final mode but can also be compiled in draft mode with an extra big margin for todo notes. To enable this, goto `preamble.tex` and replace the line `line draft=false` with `final`. Remember to switch it of when compiling your thesis for printing.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Some todo

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Structure of the thesis	2
2. Fundamentals	3
2.1. Computer Vision	3
2.2. Machine and Deep Learning	3
2.2.1. Backpropagation	6
2.2.2. Vanishing-Gradient-Problem	6
2.2.3. Exploding-Gradient-Problem	6
2.2.4. Overfitting	7
2.3. One Stage and Two Stage Detectors	8
2.4. Historical Development of YOLO Architectures	8
2.5. Evaluation Metrics	10
2.5.1. K Fold Cross Validation	10
2.5.2. Intersection over Union (IoU)	10
2.5.3. Mean Average Precision (mAP)	11
3. State of research	13
4. Methodology	17
4.1. Database	18
4.1.1. Vehicle Detection on Aerial Images (VEDAI) Dataset	18
4.1.2. Dataset for Objectdetection in Aerial Images (DOTA) 1.5	19
4.2. YOLOv9	19
4.2.1. YOLOv9u by Wong Kin Yiu	19
4.2.2. YOLOv9e by Ultralytics	20
4.3. Programming Environment	20
4.3.1. Python Packages	21
4.3.2. Weitere Python-Packages zur Datenaufbereitung und -analyse	21

4.4. Implementation Preprocessing	22
4.4.1. DOTA Dataset	22
4.4.2. VEDAI Dataset	23
4.4.3. Fold Creation	26
4.5. Challenges Preprocessing	27
4.6. 5 Fold Cross Validation	27
4.7. High-Performance-Cluster PALMA	29
4.8. Bash Scripte	29
5. Results	31
5.1. Oriented Bounding Boxes and Axis Aligned Bounding Boxes	31
5.2. Comparision at Mean Average Precision 50-95	32
5.3. Comparision with Confusion Matrices	33
5.4. Ablation Studies	34
6. Discussion	35
7. Conclusion and Outlook	37
A. Appendix	39
List of Abbreviations	41
List of Figures	45
List of Tables	47

1 | Introduction

sämtliche
Abkürzungen
mit glocaries
einbinden und
verwalten

1.1. Motivation

Durch die leichte Verfügbarkeit von hochauflösenden Satelliten- und Luftbildern ist Detektion von Objekten weltweit möglich. Da außerdem die hohe Rechenleistung, die für maschinelles Lernen erforderlich ist, immer preiswerter wird, wird die Anwendung von künstlicher Intelligenz zur vollautomatischen Analyse immer leichter.

Das Monitoring von zerstörten (militärischen) Gerät oder Gebäudestrukturen in Konfliktregionen, wie Darfur [1] oder der Ukraine, kann durch künstliche Intelligenz unterstützt werden. Hier können Luftbilder gut genutzt werden, weil das Erstellen von Fotos vor Ort durch die Sicherheitslage am Boden in Konfliktregionen gefährlich sein kann.

Das Monitoring von militärischem Gerät an Ländergrenzen könnte benutzt werden um eine genauere Lageeinschätzung zu generieren, ob ein Konflikt bevorsteht. Es ist auch möglich die Arbeit von menschlichen Analysten unterstützen, da die Modelle Vorschläge für mögliche Detektionen und Klassifizierungen von Objekten liefern.

Maschinelles Lernen kann zur Automatisierung angewendet werden, um größere Datensätze zu verarbeiten. Eine Herausforderung hier besteht im Training der Algorithmen, da die Verfügbarkeit von Trainingsdaten limitiert ist.

Ein Anwendungsszenario wäre die Auswertung des menschlichen Mobilitätsverhaltens im Rahmen des Zählen von Autos auf großen öffentlichen Parkplätzen. Hier kann auch ein Tracking von Verkehrsaufkommen durch die KI landesweit erfolgen, wenn Fahrzeuge auf einem Satellitenbild detektiert werden können. Dieser Ansatz kann genutzt werden um abzuschätzen ob mehr Parkflächen erforderlich sind oder die Anzahl der aktuellen Flächen ausreicht. Weitere Fragestellungen wären, ob es möglich ist fahrende und stehende Autos zu unterscheiden.

Außerdem kann in dieser Arbeit evaluiert werden, ob die Präzision der Objekterkennung und -klassifizierung von Fahrzeugen in elektro-optischen, multi-spektralen Luftaufnahmen durch die Nutzung von mehr Bildkanälen (NIR, IR) verbessert werden kann. Dies kann durch einen Vergleich

der Ergebnisse beim Trainieren desselben Deep Learning Modells geschehen, welches jeweils nur mit 3 und einmal mit mehr Kanälen trainiert wurde. Da Fahrzeuge relativ klein auf hochauflösenden Satellitenbildern dargestellt werden, kann auch evaluiert werden, wie gut sich ein Deep Learning Modell zur Klassifizierung sehr kleiner Objekte eignet, bzw. ob ein weiterer Kanal die Präzision verbessert.

1.2. Structure of the thesis

2 | Fundamentals

2.1. Computer Vision

Nochmal neu formulieren?

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 [2]. Zur Informationsgewinnung und zur Simulation menschlicher visueller Wahrnehmung werden Algorithmen sowie optische Sensoren eingesetzt [3]. 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 [4].

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 [5]. 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 [6], [7]. Diese Fähigkeit eröffnet neue Möglichkeiten bei der Lösung komplexer Probleme, die sich mit konventionellen, regelbasierten Methoden nur schwer oder gar nicht bearbeiten lassen [8].

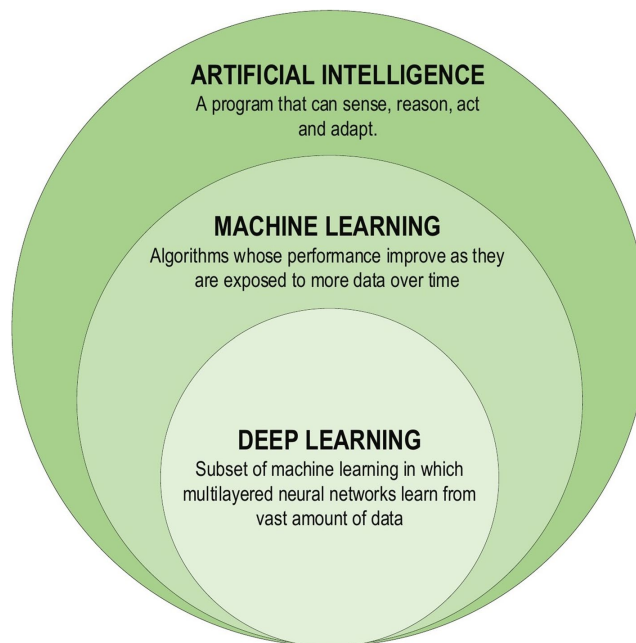


Figure 2.1.: Overview over Artificial Intelligence, Machine Learning and Deep Learning. [9]

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 [9].

Zu den zentralen Vorteilen von DL zählen seine universelle Anwendbarkeit in verschiedensten 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[9].

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 [9].

Innerhalb der Vielzahl an Netzarchitekturen haben sich CNNs als die wohl bekannteste und am häufigsten genutzte Variante etabliert. Inspiriert von biologischen Neuronenstrukturen identifizierte [8] 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 [9]. 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 durch bessere Generalisierung, Verhinderung von Overfitting durch Weight Sharing, flexible Nutzung extrahierter Merkmale und einfache Skalierbarkeit aus. Zu den zentralen Komponenten einer CNN-Architektur gehören Convolutional Layers mit trainierbaren Filtern, Pooling Layers zur

Detaillierte
Convolutional
Operation
noch einfü-
gen

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 [9].

2.2.1. Backpropagation

Der Lernprozess in Deep-Learning-Netzen basiert auf dem Prinzip der Backpropagation [8]. 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 [9].

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 [9].

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 [9].

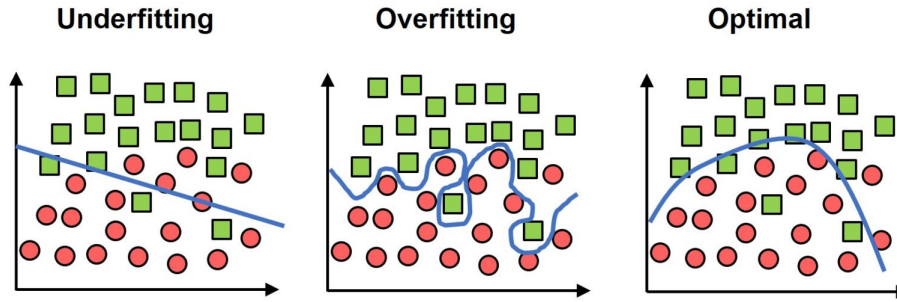


Figure 2.2.: Underfitting, Overfitting and Optimal at Deep Learning Trainings. [10]

2.2.4. Overfitting

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 [10].

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 [9].

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 [9].

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 [11].

Two-Stage-Detektoren, wie beispielsweise Region-Based Convolutional Neural Network (R-CNN) [12] oder Mask R-CNN [13], 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 [11].

Im Gegensatz dazu behandeln One-Stage-Detektoren, wie You Only Look Once (YOLO) [14] oder der Single Shot Multibox Detector (SSD) [15], 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 [11].

2.4. 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 [14]. 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 [16].

Auf YOLOv1 aufbauend, verbesserte YOLOv2, bzw. YOLO9000 [17], [18], 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 [19].

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 [20], [21], [22].

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 leistungstarker Hardware [16], [23]. Auf dieser Basis bauten die nachfolgenden Versionen YOLOv5 bis YOLOv11 auf, wobei der Fokus auf verbesserter Skalierbarkeit, reduzierten Rechenanforderungen und optimierten Echtzeit-Leistungsmetriken lag.

YOLOv6, vorgestellt 2022 von einem Team eines chinesischen E-Commerce-Plattformbetreibers [24], verfügte über eine neuartige Backbone- und Neck-Architektur. Zudem wurden fortschrittliche Trainingstechniken wie Anchor-Aided Training (AAT) und Self-Distillation implementiert [16], [24]. YOLOv7 [25], [26] führte weitere Innovationen ein, darunter trainable bag of freebies, also Optimierungen zur Steigerung der Genauigkeit ohne zusätzliche Inferenzkosten, sowie dynamische Label-Zuweisungen [16], [25], [26].

YOLOv8, veröffentlicht 2023 von Ultralytics [27], zeichnete sich durch eine effizientere Architektur, verbesserte Trainingstechniken und erweiterte Unterstützung für große Datensätze aus [16], [27]. YOLOv9 [28] 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 [16], [28].

YOLOv10, entwickelt 2023 von chinesischen Wissenschaftlern [29], präsentierte einen neuartigen Ansatz für die Echtzeit-Objekterkennung. Durch den Verzicht auf Non-Maximum-Suppression (NMS) und die Optimierung zentraler Modellkomponenten wurden Einschränkungen früherer YOLO-Versionen überwunden, was zu erheblichen Effizienz- und Leistungssteigerungen führte [29]. YOLOv11 verbesserte die Backbone- und Neck-Architektur weiter, wodurch die Merkmalsextraktion optimiert und ein ausgewogenes Verhältnis zwischen Präzision und Recheneffizienz erreicht wurde [16], [30].

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 [16], [31].

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 Modelauswahl 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 [32]. 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 [33].

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)

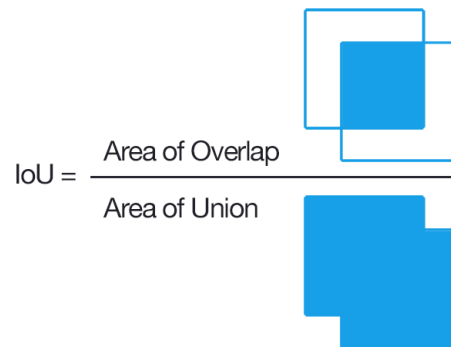


Figure 2.3.: Graphical Visualiziation for IoU. [34]

Die Intersection-over-Union (IoU), 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 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 [35].

Aufgrund dieser Unabhängigkeit basiert eine Vielzahl von Leistungsmaßen in den Bereichen Segmentierung [36], [37], [38], [39], Objekterkennung [39], [40] und Objektverfolgung [41], [42] auf der IoU. Dabei entspricht eine der beiden BB der Ground Truth, 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:

$$\text{IoU}_k = \begin{cases} \frac{|B_t \cap B_p|}{|B_t \cup B_p|}, & \text{if } |B_t \cup B_p| > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2.1)$$

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 [43].

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 [8]. 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} .

mathematische
definition ein-
fügen?

$$P_{int}(r) = \max_{r \leq y} P(y) \quad (2.2)$$

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. 2.4), die als Mittelwert der Average Precision (AP) (s. Form. 2.3) über alle Klassen ermittelt wird (s. Def. 2.5.3)[44]. 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 Ground Truth und Vorhersage und ist besonders geeignet, wenn die vorhergesagten Positionen der Bounding Boxes sehr präzise sein müssen [44].

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:

$$AP_c = \int_0^1 P_{int}(r) dr \quad (2.3)$$

$$mAP = \frac{1}{|C|} \sum_c AP_c \quad (2.4)$$

3 | State of research

Deep Learning kann zusammen mit Fernerkundungsdaten für eine Vielzahl von Anwendungszwecken eingesetzt werden, beispielsweise für Landnutzungs- und Landbedeckungsklassifikation, Szenenklassifikation oder Objekterkennung [45]. Darüber hinaus finden sich Einsatzgebiete in der Hyperspektralanalyse, in der Interpretation von SAR-Bildern, in der Interpretation hochauflöster Satellitenbilder, in der multimodalen Datenfusion sowie in der 3D-Rekonstruktion. Besonders hervorzuheben sind hierbei die Hyperspektralanalyse und die Interpretation hochauflöster Satellitenbilder. In der Hyperspektralanalyse wurde beispielsweise ein SAE genutzt, um hierarchische Merkmale in der spektralen Domäne zu extrahieren [46]. Ebenso konnten überwachte Convolutional Neural Networks erfolgreich für die Klassifizierung von Anbauarten eingesetzt werden [47].

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 Satellitensysteme wie Sentinel-1, Sentinel-2 oder Landsat vergleichsweise geringe räumliche Auflösungen (typischerweise im Bereich von 10–30 m) aufweisen [48]. Demgegenüber besitzen luftgestützte Aufnahmen – wie sie etwa im Vehicle Detection on Aerial Images (Dataset)-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 [49]. 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 [50]. Sensoren können mehrere Bänder simultan aufzeichnen; ein Echtfarbbild besteht typischerweise aus den Bändern R (Rot), G (Grün) und B (Blau), die durch weitere Bänder wie Near-Infrared (NIR) oder andere Infrarotbänder erweitert werden können. Aus solchen Kanälen lassen sich zusätzlich Indizes (z. B. NDVI) für die Analyse ableiten [2]. 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% [51].

Für die Interpretation hochauflöster Satellitenbilder spielt Deep Learning eine zentrale Rolle. Ein wichtiges Beispiel ist die Szenenklassifikation, bei der jedem Bild automatisch ein seman-

tisches Label zugeordnet wird. Weit verbreitete Anwendungsbereiche sind Objekterkennung [52], [53], Change Detection [54], 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 unterscheidungskräftigere 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 hochaufgelösten Satellitenbildern [45].

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 [45]. 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 hochaufgelösten Satellitenbildern sicher zu detektieren und korrekt zu klassifizieren [55]. Moderne Satellitenmissionen wie die von Planet (50 cm/Pixel) [56] oder die Airbus Pléiades Neo Satelliten (30 cm/Pixel) [57] 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 [58].

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 [59]. Weitere Arbeiten kombinierten tiefe, aus vortrainierten CNNs extrahierte Umgebungsmerkmale mit klassischen lokalen Merkmalen wie Histogrammen orientierter Gradienten [60] zur zuverlässigen Detektion von Öltanks [61]. 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 [62]. Das Problem variierender Objektorientierungen wurde durch die Nutzung kombinierter CNN-Layer-Merkmale adressiert, die orientierungsrobuste Erkennungen in einem groben Lokalisierungsrahmen ermöglichen [63].

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 [64]. Im Bereich der Fahrzeugdetektion wird das panchromatische Band für die Fahrzeugklassifikation eingesetzt, während multispektrale Informationen zur Maskierung von Vegetation und Schatten genutzt werden [55].

Für die Detektion von Fahrzeugen in hochauflösenden Satellitenbildern wurden hybride CNN-Modelle entwickelt, welche Feature-Maps aus Convolutional- und Pooling-Layern in Blöcke variabler Größe unterteilen, um mehrskalige Merkmale zu extrahieren [65]. Ein alternativer Ansatz nutzt graphbasierte Superpixel-Segmentierung, um Bildausschnitte zu extrahieren, die anschließend von einem CNN daraufhin klassifiziert werden, ob ein Fahrzeug vorhanden ist [66].

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 Leistung durch zusätzliche Kanäle verbessern, da Vegetation (bspw. durch Hinzufügen des NDVI) besser separierbar ist und IR zusätzliche informationsreiche Signale liefern kann. 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. Diese Fragestellungen bilden die Grundlage und Motivation der vorliegenden Arbeit.

4 | Methodology

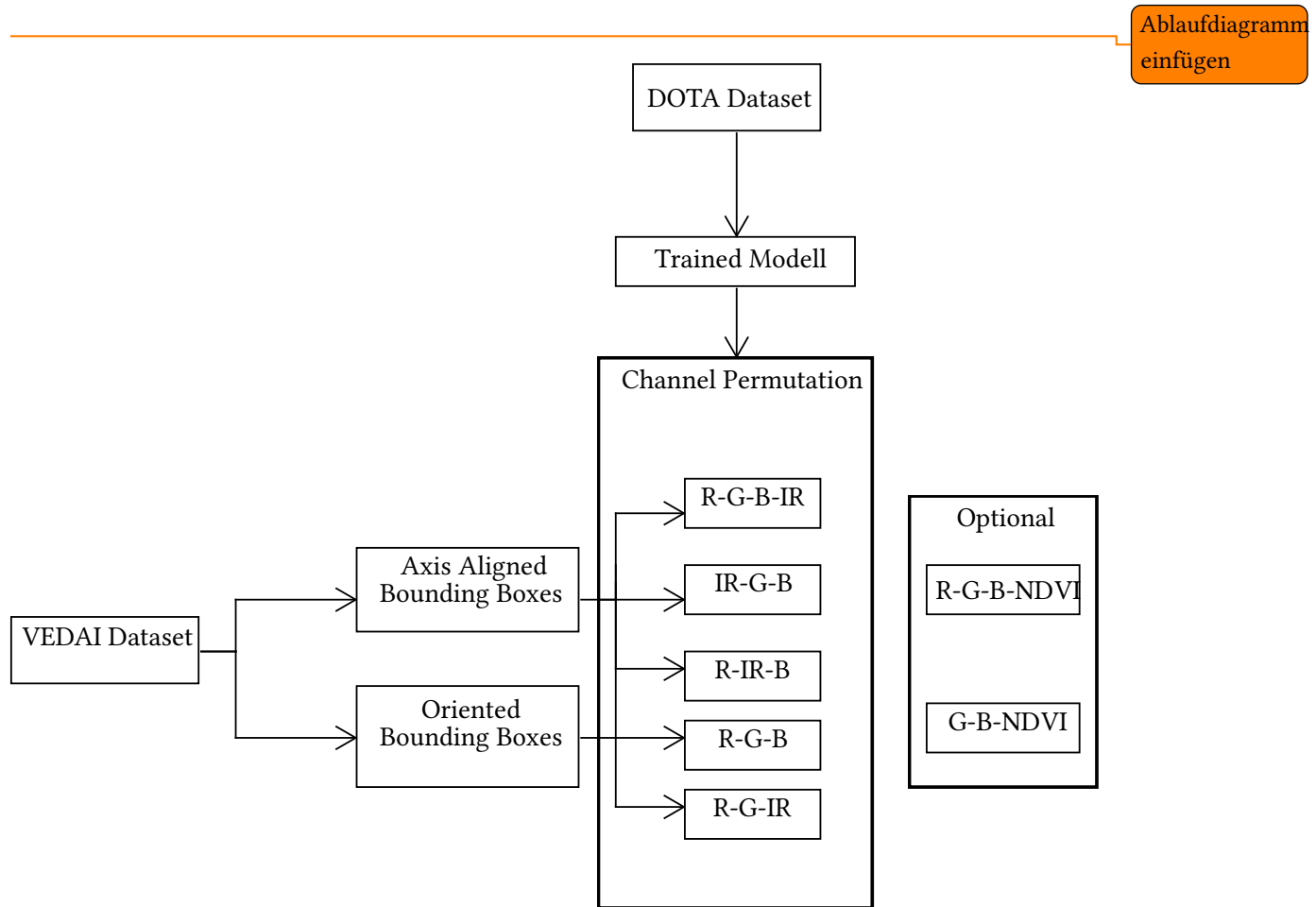


Figure 4.1.: Flowchart for workflow. Source: own representation

Die methodische Vorgehensweise dieser Arbeit ist im zugehörigen Flowchart (s. Fig. 4.1) dargestellt. 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. Danach werden die zentralen Herausforderungen im Preprocessing dargestellt. Anschließend werden die Ergebnisse der 5-Fold-Cross-Validation präsentiert, wobei eigene Folds erstellt wurden, die für alle Experi-

Verweis auf Github Repo noch erwähnen?

mente mit abb, obb sowie den Permutationen (s. Tab. 4.1) genutzt wurden. Abschließend wird die Hardware beschrieben, auf der die Experimente durchgeführt wurden, nämlich das 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 [67] aus dem Jahr 2015 bietet sich als Datengrundlage an, da es hochauflösende Luftbilder enthält, die speziell für die Fahrzeugerkennung geeignet sind [49]. 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 NIR oder IR auf die Objekterkennung macht. Es sind mehr als 3700 Objekte in ungefähr 1200 Bildern annotiert. Diese Objekte sind in 9 Klassen (Boat, Camping Car, Car, Pickup, Plane, Tractor, Truck, Van und Others) unterteilt und der Hintergrund der Objekte ist abwechslungsreich, was die Robustheit des trainierten Modelles erhöht [49].

Die Auflösung der Bilder liegt bei 12.5 cm ×12.5 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 [49].

Bsp. Bild für jede Klasse zeigen (aus Paper nehmen!; Quellenverlinkung für text und bild nicht vergessen)

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. Er 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. Die Datensätze umfassen 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 und weiteren kommerziellen oder institutionellen Anbietern.

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

4.2. YOLOv9

Um Objekte auf hochauflösenden Satellitenbildern zu detektieren, bietet sich der "You Only Look Once (YOLO)" Algorithmus an. Dieser Algorithmus, der in der Version 9 im Jahr 2024 veröffentlicht wurde [68], ermöglicht eine schnelle und präzise Objekterkennung. YOLO basiert auf einer einzigen neuronalen Netzwerkarchitektur, die das Bild in Raster unterteilt und für jedes Rasterfeld Vorhersagen über die Position und Klasse von Objekten trifft.

Die Hauptvorteile von YOLO sind seine Geschwindigkeit und Genauigkeit, die es ermöglichen, große Datensätze effizient zu analysieren. Für diese Arbeit wird YOLO verwendet, um Fahrzeuge auf Luftbildern zu detektieren und zu klassifizieren. Dabei könnten zusätzliche multispektrale Kanäle wie NIR oder IR integriert werden, um einen Vergleich zu dem reinen 3 Kanal RGB Training zu ermöglichen. Die Vergleichbarkeit kann gewährleistet werden, wenn die gleichen Trainings- und Testdaten verwendet werden, bei denen der einzige Unterschied das weitere Band ist.

4.2.1. YOLOv9u by Wong Kin Yiu

Die Implementierung von YOLOv9u basiert auf der Version YOLOv8.1.23[68] und stellt eine umfassende Weiterentwicklung der YOLO-Architektur durch den Einsatz von PGI und GELAN dar.

Hinweis:
Für eine bessere Anschaulichkeit sollen beispielhafte Bilder für jede Klasse aus der Originalpublikation übernommen und entsprechend mit Quellenangabe referenziert werden.

Neben der klassischen Objekterkennung unterstützt YOLOv9u zusätzliche Aufgaben wie Instanzsegmentierung, orientierte Objekterkennung, Posenabschätzung, Bildklassifikation sowie transformer-basierte Objekterkennung[29]. Damit bietet das Modell eine erweiterte Funktionalität gegenüber früheren YOLO-Versionen und eignet sich insbesondere für komplexe Szenarien mit mehreren Aufgabenstellungen innerhalb eines Frameworks. Diese Version wird genutzt, da sie oriented Bounding Box unterstützt.

4.2.2. YOLOv9e by Ultralytics

Die YOLOv9-Modellreihe umfasst verschiedene Varianten, die sich hinsichtlich ihrer Modellgröße, Genauigkeit und Rechenkomplexität unterscheiden. Die Bandbreite reicht dabei von der kompakten YOLOv9t-Variante mit einer Eingabebildgröße von 640×640 Pixeln, 2 Millionen Parametern und 7,7 Giga Floating Point Per Seconds (GFLOPs) bis hin zur leistungsstärksten YOLOv9e-Variante mit 58,1 Millionen Parametern und 192,5 GFLOPs (siehe Tabelle 4.2).

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.

Aufgrund der höchsten erzielten Genauigkeit ($\text{mAP}_{\text{val}}^{50-95} = 55.6$) sowie der höchsten Rechenkapazität wurde das YOLOv9e-Modell für alle Experimente mit axis-aligned Bounding Boxes verwendet. Die Wahl dieses Modells erfolgte mit dem Ziel, die bestmögliche Detektionsleistung im Rahmen der Evaluierung zu gewährleisten. Diese Version wird genutzt, da sie axis-aligned Bounding Box unterstützt.

4.3. Programming Environment

Die genutzte Programmiersprache ist Python. Die Skripte 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) [69]. Die aktuelle Version 4.12.0 wurde am 09.07.2025 veröffentlicht [70].

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.

Numpy

Ein weiteres Open-Source Projekt ist NumPy, welches 2005 gegründet wurde, um numerische Operationen in Python zu implementieren [71]. Die aktuelle Version ist 2.3.0 und wurde am 07.06.2025 veröffentlicht [72].

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 [73]. Diese Bibliothek bietet Algorithmen für wissenschaftliche Berechnungen in Python an [73]. Die Unterfunktion "ConvexHull" wird zur Flächenberechnung der oriented Bounding Boxen genutzt.

ab jetzt nur
für analyse

4.3.2. Weitere Python-Packages zur Datenaufbereitung und -analyse

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[74]. 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 [75]. 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 [76].

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.

4.4.1. DOTA Dataset

Im Rahmen der Datenvorverarbeitung wurden ausschließlich die originalen Rohbilder des DOTA-Datensatzes verwendet, ohne dass ein Zuschchnitt oder eine anderweitige Modifikation der Bildinhalte vorgenommen wurde. Der Fokus der Verarbeitung lag auf der Umwandlung der vorhandenen Labeldaten in das YOLOv9-0BB-Format, welches polygonale Objekumrandungen in Form einer Punktsequenz $(x_1, y_1, x_2, y_2, \dots)$ beschreibt (vgl.).

text\todo{Ref
zur
Formatbeschreibung
einfügen}

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 YOLOv9-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 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-Sets, perm-Datasets oder aab_vs_obb-Konfigurationen. 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. Beispielsweise 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 Dataset aus Trainings-, Validierungs- und Testdaten generiert (vgl.).

Verweis auf
entsprechende
Funktion

Erzeugung von Folds via `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 Zieldatei übernommen werden (vgl. Referenz zur Funktion). Im Anschluss wird die Funktion `create_label_file` aufgerufen, um das zugehörige Label-File im gewünschten Format zu erzeugen (vgl. Referenz zur Funktion).

ref einfügen

ref einfügen

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 der Vorgabe im `perm_object` eine kombinierte Bilddarstellung. Die Verschmelzung erfolgt mit Hilfe von `cv2.merge`. Falls gefordert, wird innerhalb dieser Funktion zusätzlich ein NDVI-Bild erzeugt. Zur numerischen Stabilisierung bei der NDVI-Berechnung wird der Nenner ($IR + R$) auf mindestens 0,01 gesetzt, um Division durch Null zu vermeiden. Die NDVI-Werte werden anschließend skaliert und als 8-Bit-Bild zurückgegeben. Vor

dem Zusammenführen werden die RGB-Kanäle immer einzeln extrahiert. Die Funktion `copy_image` übernimmt dann das Schreiben der erzeugten Bilder an den vorgesehenen Zielpfad.

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` 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 DOTA-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 Fahrzeuglänge, 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 einer Vorverarbeitungsfunktion (`preproc`) 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 ref einfügen separate Funktion `()` 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 ref funktion dafür seltensten vorhanden ist(). 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 dabei mit einem Gewicht von 0 (häufig) bis 8 (selten) bewertet. 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.

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.

ref funktion
einfügen

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-OB-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 oder größer als 1). Dieses Problem wurde durch den Einsatz von Exception Handling sowie durch Rundung der Werte auf den gültigen Wertebereich behoben.

Beispielbilder
für Randlage
einfügen

4.6. 5 Fold Cross Validation

Um die Robustheit der Ergebnisse sicherzustellen und eine fundierte Vergleichbarkeit verschiedener Modelle zu ermöglichen, wurde in dieser Arbeit eine 5-Fold-Cross-Validation angewendet. Insbesondere wurde Wert darauf gelegt, die Evaluation nicht nur auf einem Teildatensatz durchzuführen, son-

Fold Probleme
bei VEDAI
erwähnen?

dern eine Mehrfachteilung der Daten vorzunehmen, sodass Schwankungen durch unterschiedliche Trainings- und Validierungsaufteilungen minimiert werden.

Im Gegensatz zu den von Razakarivony et al. [49] bereitgestellten Folds, deren Zusammenstellung identische Bilder gleichzeitig im Trainings- und Validierungssatz aufweist, wurden für diese Arbeit eigene, disjunkte Folds generiert. Letztere gewährleisteten, dass keine Überlappungen zwischen Trainings- und Validierungsdaten existieren und somit die Gefahr von verfälschten Trainingsergebnissen 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 Modelle 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 [77]) 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 [78] und verfügt über die in Tab. 4.4 technische Spezifikationen [78]:

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	$R_{max} = 800$ TFLOP/s; $R_{peak} = 1,277$ 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 großen Bilddaten effizient zu verarbeiten.

4.8. Bash Skripte

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.

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

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

5 | Results

- Konsistenz aller Modelle über die 5 Folds der Kreuzvalidierung ist gut, was eine robuste Trainierbarkeit bedeutet

•

5.1. Oriented Bounding Boxes and Axis Aligned Bounding Boxes

- Training Results by the mAP50-95 of all 5 Folds for every model
- Left Side (Blue): Yolov9 (without obb) and with axis aligned Bounding Boxes
- Middle (Orange): Yolov9u with oriented bounding boxes
- Uses Ultralytics as Backend for obb support
- Right (Green): Yolov9u with axis aligned Bounding Boxes in the oriented Bounding Box format
- Result -> aab performs better

•

- This is probably due to the fact that small changes in the orientation of the bounding boxes lead to a greater deviation of the mean average precision
- Oriented Bounding boxes are smaller than the axis aligned bounding boxes -> higher inaccuracy of mAP50-95 (see next slide)
- Red box at the right picture has a small deviation from the correct (blue) label
- BB entsprechen der Definition von klein nach [51]; nehmen als obb (725 px pro Box) ca, 0.069% der Gesamtfläche des Bildes ein; und als aab 0.095%; obb liegt damit knapp unterhalb der Untergrenze (very small object) und aab oberhalb der untergrenze (small object)

•

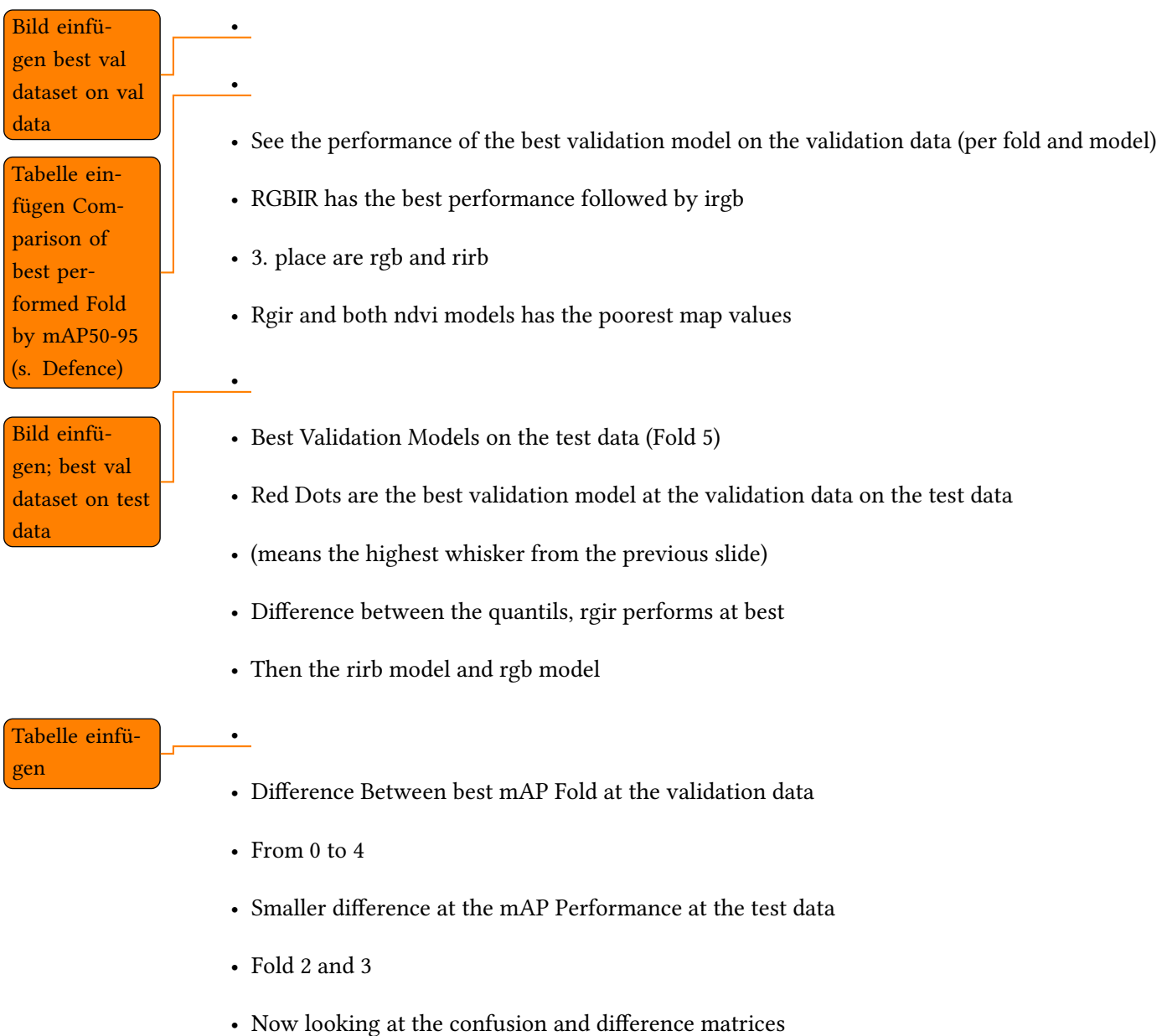
tabelle mit
folds einfügen;
Verteilung
beschreiben;
dann übergang
zu weiteren
punkten

Vergleichsbild
einfügen oder
verweis


Bild für Ver-
gleich der BB
Area einfügen
(zw. obb, aab
aab old)

- Oriented Bounding Boxes has an area from around 600 to 900 pixels
- Axis aligned Bounding boxes from around 750 to 1300 px
- -> higher inaccuracy of mAP50-95
- For the channel permutation I used the obb model


5.2. Comparison at Mean Average Precision 50-95




5.3. Comparison with Confusion Matrices

- 
- Normalisierte Konfusions Matrix mit 9 Klassen und der Einteilung Background (Hintergrund)
- Right: Confusion Matric of RGBIR Modell on Fold 2:
- Good Performance at Cars, Trucks, Ships, Tractors, Camping Cars, Pick Up and planes
- Many confusion at vehicle with background
- Nearly the same for RGIR at Fold 3
- Both modells detected nearly all Planes
- Difference Matrics at next slide


beide confu-
sionsmatricen
einfügen

- 
- At all Difference Matrics is red for the better performance of the RGBIR Model and blue for the other modell
- RGBIR detected Trucks, Ships and Tractor more accurate than the RGIR Model
- RGBIR seems more robust for common ground vehicles
- Both modells detect all planes (in short every modell detect all planes)
- Both modells misclassifier background as any class (RGBIR is better at vehicle and Camping Car)

Differenzmatrix
RGBIR vs
RGIR einfügen

- 
- RGBIR is better for detecting Ships (12%) and Vehicles (15%), slightliy degrades performance of car detection
- Vehicle is much better than in RGB Model (15%)
- IR input helps for common vehicle recognition (vehicle includes Excavators, construction equipment, etc.)
- RGB is better in identifying ship and vehicle as background

Differenzmatrix
RGBIR vs RGB
einfügen

- 
- RGBIR recognize Ships 21 % better than the irgb model

Differenzmatrix
RGBIR vs
IRGB einfügen

- Particular stronger for camping car and van classification (6%)
- IRGB has more background-ship and background-vehicle confusion

Differenzmatrix
RGBIR vs
RIRB einfügen

-
- RGBIR detects Ships and vans better than RIRB Model
- RIRB misclassified van and ship as background more often than RGBIR

Differenzmatrix
RGBIR vs
GBNDBVI und
RGBNDVI
einfügen

-
- RGBIR have a better performance in Ship, Tractor and Vehicle
- Both ndvi models show a tendency for confusing background with other classes

5.4. Ablation Studies

Tabelle mit
Best Folds Ab-
lation Studies
einfügen

-

6 | Discussion

7 | Conclusion and Outlook

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

And after the second paragraph follows the third paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

After this fourth paragraph, we start a new paragraph sequence. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Definition 7.0.1 (σ algebra)

Let X be some set. A subset $\Sigma \in \mathcal{P}(X)$ is called a σ -algebra if it satisfies the following three properties:

1. $X \in \Sigma$
2. $A \in \Sigma \implies X \setminus A \in \Sigma$
3. $A_1, A_2, \dots \in \Sigma \implies A_1 \cup A_2 \cup \dots \in \Sigma$

For a full list of environments see `preamble.tex`

A | Appendix

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

List of Abbreviations

AA2 Area Attention (AA2)-Modul. 9

AAT Anchor-Aided Training. 9

abb axis-aligned Bounding Box. 15

ANN Artificial Neural Network. 4

AP Average Precision. 11

B Blue. 18, 24

BB Bounding Box. 10, 11, 24

CIT Center for Information Technology. 21

CNN Convolutional Neural Network. 5, 13–15

CPU Central Processing Unit. 21

CSPN Cross-Stage-Partial-Network. 8

DCNN Deep Convolutional Neural Network. 14

DL Deep Learning. 4, 5, 13, 14

DNN Deep Neural Network. 5

DOTA Dataset for Object Detection in Aerial Images. 17, 20, 23, 24, I

FCL Fully Connected Layer. 5, 6

FIR Far-Infrared. 15

G Green. 18, 24

- GB** Gigabyte. 21
- GBNDVI** Green, Blue, Normalized Difference Vegetation Index. 18
- GELAN** Generalized Efficient Layer Aggregation Network. 9
- GFLOP** Giga Floating Point Per Second. 19
- GHz** Gigahertz. 21
- GPFS** General Parallel File System. 21
- GPU** Graphics Processing Unit. 21
- HPC** Hyper-Performance-Cluster. 21
- IoU** Intersection-over-Union. 10, 11, 18
- IR** Infrared. 15, 18–20, 24, 25
- IRGB** Infrared, Green, Blue. 18
- KCV** K-fold Cross Validation. 10
- mAP** Mean Average Precision. 11, 18, 19
- MIR** Middle-Infrared. 15
- NaN-Value** Not A Number Value. 6
- NDVI** Normalized Difference Vegetation Index. 13, 15, 18, 24
- NIR** Near-Infrared. 13, 15, 19, 20
- NMS** Non-Maximum-Supression. 9
- obb** oriented Bounding Box. 15, 24
- PALMA** Paralleles Linux-System für Münsteraner Anwender. 21, 23, 24
- PB** Petabyte. 21
- PGI** Programmable Gradient Information. 9
- PRC** Precision-Recall-Curve. 11

R Red. 18, 24

R-CNN Region-Based Convolutional Neural Network. 8

R-ELAN Residual Efficient Layer Aggregation Networks. 9

ReLU Rectified Linear Unit. 6

RGB Red, Green, Blue. 15, 18–20, 25

RGBIR Red, Green, Blue. Infrared. 18

RGBNDVI Red, Green, Blue, Normalized Difference Vegetation Index. 18

RGIR Red, Green, Infrared. 18

RIRB Red, Infrared, Blue. 18

RNN Recurrent Neural Network. 5

RPN Region Proposal Network. 8

SAE Stacked Auto Encoder. 13, 14

SSD Single Shot Multibox Detector. 8

SVM Support Vector Machine. 5, 14

TFLOP Teraflops. 21

VEDAI Vehicle Detection on Aerial Images (Dataset). 13, 17, 20, 24

YAML "YAML Ain't Markup Language" or originally "Yet Another Markup Language". 24

YOLO You Only Look Once. 8, 15, 17–19, 23, 24

List of Figures

2.1.	Overview over Artificial Intelligence, Machine Learning and Deep Learning. [9]	4
2.2.	Underfitting, Overfitting and Optimal at Deep Learning Trainings. [10]	7
2.3.	Graphical Visualiziation for IoU. [34]	10
4.1.	Flowchart for workflow. Source: own representation	17

List of Tables

4.1.	Channel permutations represented by included channels (x = channel present). . .	18
4.2.	Comparison of YOLOv9 model variants in terms of accuracy and computational complexity at an input resolution of 640×640 pixels.	20
4.3.	Class distribution across folds.	28
4.4.	System overview PALMA	29
4.5.	Key training hyperparameters at the RGB Permutation (Fold 4)	30

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] 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. 3, 13).
- [3] 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. 3).
- [4] 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. 3).
- [5] 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. 3).
- [6] 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. 4).
- [7] 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. 4).
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org> (cit. on pp. 4–6, 11).
- [9] 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. 4–7).

- [10] 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. 7).
- [11] 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. 8).
- [12] 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. 8).
- [13] 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. 8).
- [14] 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. 8).
- [15] 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. 8).
- [16] 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. 8, 9).
- [17] 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. 8).
- [18] 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. 8).
- [19] 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. 8).

-
- [20] 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. 8).
- [21] 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. 8).
- [22] 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. 8).
- [23] Ultralytics, *Ultralytics Github*, 2020. Accessed: Aug. 26, 2025. [Online]. Available: <https://github.com/ultralytics> (cit. on p. 9).
- [24] 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. 9).
- [25] 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. 9).
- [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,” 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. 9).
- [27] G. Jocher and A. Chaurasia and J. Qiu, *Ultralytics Github*, 2023. Accessed: Aug. 26, 2025. [Online]. Available: <https://github.com/ultralytics> (cit. on p. 9).
- [28] 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 p. 9).
- [29] C.-Y. Wang and H.-Y. M. Liao, “YOLOv9: Learning what you want to learn using programmable gradient information,” 2024 (cit. on pp. 9, 20).
- [30] Ultralytics, *Ultralytics*, 2024. Accessed: Aug. 26, 2025. [Online]. Available: <https://docs.ultralytics.com/de/models/yolo11/> (cit. on p. 9).
- [31] 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. 9).

- [32] 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. 10).
- [33] 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. 10).
- [34] 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. 10).
- [35] 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. 10).
- [36] 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. 11).
- [37] 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. 11).
- [38] 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. 11).
- [39] 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. 11).
- [40] 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. 11).
- [41] 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. 11).
- [42] 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. 11).
- [43] Ultralytics, *Intersection over union (iou)*. Accessed: Aug. 26, 2025. [Online]. Available: <https://www.ultralytics.com/glossary/intersection-over-union-iou> (cit. on p. 11).

-
- [44] 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. 12).
- [45] 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. 13, 14).
- [46] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyper-spectral 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. 13).
- [47] 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. 13).
- [48] 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. 13).
- [49] 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. 13, 18, 28).
- [50] 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. 13).
- [51] 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. 13, 31).
- [52] 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. 14).
- [53] 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. 14).

- [54] 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. 14).
- [55] 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. 14, 15).
- [56] *Planet Labs - Main*. Accessed: Aug. 22, 2025. [Online]. Available: <https://www.planet.com/products/high-resolution-satellite-imagery/> (cit. on p. 14).
- [57] *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 p. 14).
- [58] 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. 14).
- [59] 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. 14).
- [60] 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. 14).
- [61] 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. 14).
- [62] "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. 15).
- [63] 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. 15).

-
- [64] 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. 15).
- [65] 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. 15).
- [66] 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. 15).
- [67] *Vehicle detection in aerial imagery (vedai) : A benchmark*. [Online]. Available: <https://downloads.greyc.fr/vedai/> (cit. on p. 18).
- [68] 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. 19).
- [69] *About - OpenCV*. Accessed: Aug. 4, 2025. [Online]. Available: <https://opencv.org/about/> (cit. on p. 21).
- [70] *Releases - OpenCV*. Accessed: Aug. 4, 2025. [Online]. Available: <https://opencv.org/releases/> (cit. on p. 21).
- [71] *NumPy - About Us*. Accessed: Aug. 4, 2025. [Online]. Available: <https://numpy.org/about/> (cit. on p. 21).
- [72] *NumPy*. Accessed: Aug. 4, 2025. [Online]. Available: <https://numpy.org/> (cit. on p. 21).
- [73] *SciPy - Main*. Accessed: Apr. 8, 2025. [Online]. Available: <https://scipy.org/> (cit. on p. 21).
- [74] *matplotlib - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://matplotlib.org/> (cit. on p. 22).
- [75] *seaborn - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://seaborn.pydata.org/> (cit. on p. 22).
- [76] *pandas - Main*. Accessed: Aug. 4, 2025. [Online]. Available: <https://pandas.pydata.org/> (cit. on p. 22).
- [77] 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. 29).
- [78] 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. 29).

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, August 30, 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, August 30, 2025