Hochschule München University of Applied Sciences

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

Masterarbeit

Objektbasierte Einzelbaumdetektion in urbanen Gebieten auf Grundlage von Orthophotos aus OpenData

angefertigt von

Hubertus Carlos Antonio Vier

vorgelegt bei

Prof. Dr. Sebastian Briechle
(Hochschule München)

betreut von

Dr. Michael Wurm
(Deutsches Zentrum für Luft- und Raumfahrt)

München, den 13.11.2025

Diese Masterarbeit wurde im Deutschen Fernerkundungsdatenzentrum (DFD) innerhalb des Earth Observation Centers (EOC) des Deutschen Zentrums für Luft- und Raumfahrt (DLR) in Oberpfaffenhofen erarbeitet.

Abteilung Georisiken und zivile Sicherheit
Abteilungsleiter Prof. Dr. Hannes Taubenböck

Kurzfassung

Diese Arbeit untersucht die automatisierte Erkennung von Bäumen in städtischen Gebieten mithilfe moderner KI-basierter Objekterkennung. Ziel war es, ein Verfahren zu entwickeln, das Bäume auf hochauflösenden Orthophotos zuverlässig identifiziert und dabei schnell genug ist, um große Stadtflächen effizient auszuwerten.
Als Datengrundlage dienten frei verfügbare Orthophotos von fünf Städten aus dem Open-Data-Angebot des Bayerischen Landesamts für Digitalisierung, Breitband und Vermessung. Um eine ausgewogene Repräsentation unterschiedlicher städtischer Strukturen zu gewährleisten, erfolgte ein gezieltes Szenen-Sampling auf Basis des europäischen Urban Atlas. Dieser bietet eine detaillierte Klassifizierung urbaner Landnutzungs- und Bebauungsstrukturen und wurde herangezogen, um die Flächen der Städte in vier übergeordnete Hauptkategorien zu unterteilen. Dadurch konnten typische urbane Landschaftsszenen – etwa dicht bebaute Innenstädte, Wohngebiete mittlerer Dichte und Gewerbeareale gezielt in den Datensatz integriert werden. Die so ausgewählten Szenen bildeten die Grundlage für die anschließende manuelle und halbautomatische Annotation von Baumobjekten. Durch dieses Verfahren wurde eine konsistente und vielfältige Trainingsbasis geschaffen, die sowohl die Heterogenität urbaner Räume als auch unterschiedliche Vegetationsstrukturen realitätsnah abbildet.
Für die Aufgabe der Baumdetektion wurden gezielt drei aktuellen Architekturen YOLOv11, YOLOv12 und RT-DETR in allen verfügbaren Modellgrößen auf dem gesamten Datensatz mit einem zufälligen Trainings und Validierungs-Split getestet. Dadurch konnte der Einfluss der Modellkomplexität auf die Trainingszeit, Erkennungsleistung und die Inferenzgeschwindigkeit systematisch untersucht werden. Es zeigte sich, dass größere Modelle zwar eine höhere Genauigkeit erreichen, jedoch mit deutlich höheren Trainings- und Inferenzzeiten verbunden sind. Bei der Übertragung auf die praktische Anwendung mit unbekannten Valdierungs- und Testdaten stellte das Modell YOLOv12m hierbei den besten Kompromiss dar: Es erzielte bei leichten Abschlägen in der Lokalisierungsgenauigkeit nahezu die gleiche Präzision wie die größten Varianten, bei etwa halber Inferenzzeit. In der Anwendung auf bisher unbekannte Städte konnte das Modell eine hohe Detektionsleistung erzielen, insbesondere in stark versiegelten Innenstadtbereichen mit begrenzten Grünflächen. Dies unterstreicht die Robustheit des Ansatzes und die Übertragbarkeit der trainierten Architektur auf unterschiedliche urbane Strukturen.
Die Ergebnisse zeigen, dass eine schnelle und verlässliche Erfassung von Baumstandorten aus frei zugänglichen Orthophotos möglich ist. Durch die Erweiterung des Datensatzes, den Einsatz kleinerer Kacheln und höher aufgelöster Orthophotos kann die Genauigkeit künftig weiter gesteigert werden. Aufgrund der geringen Rechenzeit könnte sich die Methode besonders für die regelmäßige Aktualisierung kommunaler Baumkataster und die datenbasierte Identifikation potenzieller Pflanzstandorte eignen.

Abstract

This study investigates the automated detection of trees in urban areas using modern AI-based object detection methods. The objective was to develop a procedure capable of reliably identifying trees in high-resolution orthophotos while remaining fast enough to efficiently analyze large urban areas.
The dataset consisted of freely available orthophotos from five cities provided through the open data platform of the Bavarian Agency for Digitization, Broadband, and Surveying. To ensure a balanced representation of different urban structures, a targeted scene sampling was carried out based on the European Urban Atlas. This dataset provides a detailed classification of urban land use and built-up structures and was used to divide the city areas into four overarching main categories. This approach allowed the targeted inclusion of typical urban landscape scenes—such as densely built-up inner cities, medium-density residential areas, and commercial zones—into the dataset. The selected scenes then served as the basis for manual and semi-automated annotation of tree objects. This process created a consistent and diverse training foundation that realistically represents both the heterogeneity of urban areas and the variability of vegetation structures.
For the task of tree detection, three current architectures—YOLOv11, YOLOv12, and RT-DETR - were systematically tested in all available model sizes on the entire dataset using a random training and validation split. This allowed for a systematic analysis of the impact of model complexity on training time, detection performance, and inference speed. The results showed that larger models achieve higher accuracy but require significantly more training and inference time. When transferring the models to practical applications with previously unseen validation and test data, YOLOv12m proved to be the best compromise: it achieved nearly the same precision as the largest variants, with only minor reductions in localization accuracy, while requiring roughly half the inference time. When applied to previously unseen cities, the model achieved high detection performance, particularly in densely built-up inner-city areas with limited green space. This highlights the robustness of the approach and the transferability of the trained architecture across diverse urban environments.
The results demonstrate that fast and reliable detection of tree locations from freely available orthophotos is feasible. By expanding the dataset, using smaller tiles, and incorporating higher-resolution orthophotos, accuracy can be further improved in future work. Due to its low computational cost, the method is particularly well suited for the regular updating of municipal tree inventories and the data-driven identification of potential planting sites.

Erklärung

gemäß § 16 Abs. 10 APO in Zusammenhang mit § 35 Abs. 7 RaPO

Name: Vier

Vorname: Hubertus

Geburtsdatum: 05.03.1995

Studiengang: Geomatik (M.Eng.)

Studiengruppe: 3G

Matrikel-Nr.: 07396320

Semester: Wintersemester 2025/ 2026

Betreuer: Prof. Dr. Sebastian Briechle

Hiermit versichere ich, dass ich die vorliegende Arbeit eigenständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle übernommenen Inhalte sowie mit Unterstützung von KI generierten Inhalte wurden entsprechend den anerkannten wissenschaftlichen Grundsätzen oder entsprechend der Regelungen zur Kennzeichnung von KI-Inhalten kenntlich gemacht. Ausgenommen von der Kenntlichmachung sind orthografische oder grammatikalische Korrekturen, Übersetzungen sowie nicht-sinnverändernde Verbesserungen von Formulierungen. Ich bin mir bewusst, dass mit KI generierte Texte keine Garantie für die Qualität von Inhalten und Text bieten. Daher erkläre ich, dass ich KI-Werkzeuge lediglich als Hilfsmittel genutzt habe, die von

KI generierten Inhalte kritisch überprüft habe und mein eigenständiger kognitiver sowie kreativer Einfluss in dieser Arbeit überwiegt. Ich versichere, dass ich die Inhalte meiner Arbeit vollständig verstanden habe und selbstständig vertreten kann. Ich versichere, dass ich ausschließlich KI-Werkzeuge verwendet habe, deren Nutzung vom Prüfer oder der Prüferin als Hilfsmittel zugelassen wurden.

München, 13.11.2025

Ort, Datum

Unterschrift

Danksagung

Hiermit möchte ich allen Personen danken, die mich während dieser Arbeit begleitet und wesentlich zum Gelingen beigetragen haben.

Besonderen Dank schulde ich Prof. Dr. Sebastian Briechle, der stets eine offene Tür für mich hatte und sich immer Zeit für meine Fragen nahm.

Weiterer Dank gilt Dr. Michael Wurm, in dessen Team ich meine Arbeit am Deutschen Institut für Luft- und Raumfahrt durchführen durfte.

Für die praktische Betreuung am Institut möchte ich mich bei Dr. Thomas Stark bedanken.

Zu guter Letzt möchte ich mich bei meiner Familie und meiner Freundin bedanken. Eure Unterstützung während des gesamten Arbeitsprozesses war von unschätzbarem Wert!

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AP	Average Precision
ASCII	American Standard Code for Information Interchange
bDOM	Bildbasiertes Digitales Oberflächenmodell
CORINE	Coordination of Information on the Environment (Land Cover)
DETR	DEtection TRansformer
DGM	Digitales Geländemodell
DOM	Digitales Oberflächenmodell
EPSG	European Petroleum Survey Group Geodesy
F-RCNN	Faster Region-based Convolutional Neural Network
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
IoU	Intersection over Union
QGIS	Quantum Geographic Information System
ReLU	Rectified Linear Unit
RGB	Rot–Grün–Blau
RT-DETR	Real-Time Detection Transformer
SL	Soil Sealing / Bodenversiegelung
SSD	Single Shot Detector
TrueDOP	Digitales True Orthophoto
UA	Copernicus Urban Atlas
UTM	Universal Transverse Mercator
YOLO	You Only Look Once

1 Einleitung

1.1 Motivation

Mit dem Fortschreiten des Klimawandels und der zunehmenden Urbanisierung gewinnen Bäume im städtebaulichen Kontext stark an Bedeutung. Die steigenden Temperaturen in Städten führen zu sogenannten urbanen Wärmeinseln, die das Mikroklima erheblich beeinflussen und die Lebensqualität der Bevölkerung mindern. Ein wesentlicher Treiber dieses Effekts ist die zunehmende Bodenversiegelung: Asphalt, Beton und andere künstliche Oberflächen speichern tagsüber Wärme und geben sie nachts nur langsam wieder ab. Dadurch steigen die nächtlichen Temperaturen, was insbesondere während sommerlicher Hitzewellen zu einer deutlichen Zunahme thermischer Belastung führt (Ribeiro u. a., 2021).
Bäume bilden in diesem Zusammenhang einen zentralen Bestandteil der grünen Infrastruktur. Sie übernehmen vielfältige ökologische und klimatische Funktionen und tragen entscheidend zur Regulierung des Stadtklimas bei. Durch Verschattung, Verdunstungskühlung, Luftverbesserung und Regenwasserrückhalt erbringen sie wichtige Ökosystemleistungen. Die Beschattung von Gebäudefassaden und versiegelten Flächen reduziert die Wärmerückstrahlung und damit die thermische Belastung. Zudem kann die Verdunstungskühlung die Lufttemperatur unter der Baumkrone um etwa 1°C senken. Auch die Luftqualität wird durch die Aufnahme von CO_2 verbessert, und bei Starkregen können Bäume Wasser aufnehmen und zur Entlastung der Kanalisation beitragen (Pauleit u. a., 2019).
Zahlreiche Studien belegen den positiven Einfluss von Bäumen auf das Stadtklima. García de León u. a. (2025) zeigen am Beispiel Münchens, dass eine zunehmende Baumkronenbedeckung über verschiedene Landnutzungsklassen hinweg signifikant mit niedrigeren Landoberflächentemperaturen korreliert. Besonders in stark versiegelten Bereichen wie Misch- oder Verkehrsflächen verringert sich die Oberflächentemperatur um bis zu 0,08°C pro zusätzlichem Prozent Baumkronenanteil. Auch Modellierungen von Pauleit u. a. (2019) bestätigen, dass Bäume in mitteleuropäischen Städten die physiologisch äquivalente Temperatur um bis zu 18% senken können. Neben der Verschattung spielt insbesondere die Verdunstung eine zentrale Rolle bei der Abkühlung der Umgebung.
Diese positiven Effekte zeigen, dass Bäume ein entscheidendes Element einer klimaresilienten Stadtentwicklung sind. Ihre Integration in städtebauliche Planungsprozesse ist daher unerlässlich. In der Praxis stellt sich jedoch die Frage, wie Bäume systematisch in die Stadtplanung einbezogen werden können, wenn oftmals unklar ist, wo sie sich befinden, wie dicht sie verteilt sind oder in welchem Zustand sie sich befinden.
Klassische Baumkataster bieten hierfür eine wichtige Datengrundlage, sind jedoch mit erheblichem personellem und finanziellem Aufwand verbunden. Die Erfassung erfolgt in der Regel manuell und kann häufig nur auf öffentlichen Flächen durchgeführt werden, da der Zugang zu privaten Grundstücken begrenzt ist. Dadurch entsteht ein unvollständiges

Bild des gesamten städtischen Baumbestands.
Eine vielversprechende Lösung bietet der Einsatz moderner Methoden der Fernerkundung. Mithilfe hochaufgelöster Luft- und Satellitenbilder sowie KI-gestützter Bildanalyseverfahren können Bäume automatisiert erkannt und klassifiziert werden. Dadurch ist es möglich, den Baumbestand großflächig, objektiv und regelmäßig zu erfassen – unabhängig von Grundstücksgrenzen und mit wesentlich geringeren Erhebungsaufwand. Diese Ansätze können eine effiziente Grundlage für datenbasierte Stadtplanung und klimatische Bewertungen bilden.

1.2 Vorarbeiten

Velasquez-Camacho u. a. (2023) haben sich mit der automatisierten Detektion und Geolokalisierung von Bäumen im urbanen Raum befasst. Dabei nutzten sie hochaufgelöste Satelliten- und Luftbilder mit räumlichen Auflösungen zwischen 10 und 60cm für die Stadt Lleida in Spanien. Ihr Ansatz kombinierte die Kronendetektion aus der Luftbildperspektive mit lokalen Datenquellen wie Google Street View, um sowohl horizontale als auch vertikale Perspektiven urbaner Bäume zu erfassen. Zur Detektion aus der Luft kamen die Modelle *DeepForest* und *Faster R-CNN* zum Einsatz, während für die Erkennung in Google-Street-View-Bildern das Modell *YOLOv5x* verwendet wurde. Ziel der Studie war es, Bäume automatisch zu erkennen, zu zählen und ihre Positionen präzise georeferenziert zu bestimmen. Mit dieser Methode gelang es, rund 79% der Straßenbäume mit einer mittleren Positionsgenauigkeit von etwa 0,6m korrekt zu identifizieren.

Beloiu u. a. (2023) untersuchten die automatische Detektion einzelner Baumkronen sowie die Artidentifikation in heterogenen mitteleuropäischen Wäldern unter Verwendung von Orthophotos aus Open-Data-Quellen. Die Orthophotos stammten vom Bundesamt für Landestopografie Schweiz (Swisstopo) und besitzen eine Bodenauflösung von 10cm. Das Untersuchungsgebiet umfasste die Regionen des Schweizer Mittellands und des Juras, in denen vier häufige Baumarten – *Picea abies* (Gemeine Fichte), *Abies alba* (Weißtanne), *Pinus sylvestris* (Waldkiefer) und *Fagus sylvatica* (Rotbuche) – analysiert wurden. Zur Kronendetektion und Artklassifikation wurde das Objekterkennungsmodell *Faster R-CNN* mit einem *ResNet-101*-Backbone eingesetzt. Die Autor:innen entschieden sich für dieses Modell, da es als zweistufige Objekterkennungsarchitektur eine besonders hohe Präzision bei der Lokalisierung kleiner und teilweise überlappender Objekte ermöglicht und sich damit besser für komplexe Kronenstrukturen eignet als schnellere, aber einfachere Ein-Schritt-Modelle wie *YOLO* oder *SSD*. Zudem erlaubt die integrierte Region-Proposal-Network-Komponente eine effiziente End-to-End-Optimierung, wodurch Genauigkeit und Rechenaufwand in einem ausgewogenen Verhältnis stehen. Die Trainingsgrundlage bildeten mehr als 10000 manuell annotierte Einzelbäume aus nationalen Forstinventuren und weiteren Referenzdatenbanken. Zusätzlich wurde der Einfluss von Datenaugmentation (90°-Rotation) sowie verschiedener Stand- und Beleuchtungsbedingungen untersucht.
Ein besonderer Fokus der Studie lag auf dem Vergleich zwischen spezifisch für eine einzelne

Art trainierten Modellen und sogenannten Multi-Species-Modellen. Erstere wurden ausschließlich auf Bildausschnitte einer Art trainiert und lieferten eine hohe Genauigkeit für klar abgegrenzte Kronenformen, wie etwa bei *Picea abies* (F1 = 0.86). Ihre Leistung nahm jedoch ab, wenn Kronen anderer Arten ähnliche visuelle Merkmale aufwiesen oder sich überlappten. Die Multi-Species-Modelle hingegen wurden mit Trainingsdaten mehrerer Arten gleichzeitig entwickelt. Dadurch konnte das Modell nicht nur artenspezifische Muster, sondern auch Unterschiede und Übergänge zwischen Arten lernen. Dies führte zu einer geringeren Verwechslungsrate und einer insgesamt höheren Robustheit, insbesondere in Mischbeständen. Während Multi-Species-Modelle längere Trainingszeiten erforderten (bis zu sechs Stunden statt 30 Minuten), reduzierten sie Fehlklassifikationen signifikant und verbesserten die Erkennungsleistung schwächer vertretener Arten wie *Pinus sylvestris* (F1 = 0.92). Die Studie belegt, dass *Faster R-CNN* auch mit freiverfügbaren RGB-Orthofotos eine präzise Kronendetektion und Baumartenidentifikation ermöglicht und damit eine praktikable Alternative zu LiDAR- oder multispektralen Ansätzen bietet.

Stark u. a. (2025) untersuchten die Detektion einzelner Bäume in urbanen Gebieten anhand sehr hochauflösender Luftbilder (10cm Bodenauflösung) einer etwa 13ha großen Wohnzone in München. Ziel war es, die Leistungsfähigkeit zweier aktueller Deep-Learning-Modelle, *YOLOv8* und *DeepForest*, für die präzise Erkennung einzelner Baumkronen zu vergleichen. Der Datensatz bestand aus manuell annotierten RGB-Orthophotos, in denen Baumkronen durch Bounding Boxes markiert wurden. Durch Vorverarbeitungsschritte wie Pixelnormalisierung und Datenaugmentation (Rotation, Spiegelung, Skalierung und Farbvariationen) wurde die Robustheit der Modelle gegenüber variierenden Umweltbedingungen erhöht. Zusätzlich wurde eine feinjustierte Version des DeepForest-Modells trainiert, die auf den Münchner Datensatz angepasst wurde, um die Generalisierungsleistung in urbanen Umgebungen zu verbessern. Beide Modelle wurden mit identischer Datengrundlage und Bewertungsmetriken (*Precision*, *Recall*, *F1-Score*) getestet. Die Ergebnisse zeigen, dass *YOLOv8s* mit einem F1-Score von 0.81 die besten Resultate erzielte, während das feinjustierte *DeepForest*-Modell mit einem F1-Score von 0.80 nahezu gleichwertig abschnitt. Das Standard-*DeepForest*-Modell ohne Anpassung erreichte dagegen nur einen F1-Score von 0.74. Die Studie verdeutlicht, dass moderne domänenspezifisch optimierte Deep-Learning-Modelle wie *YOLOv8* und ein *DeepForest* präzise und skalierbare Ergebnisse für die urbane Baumdetektion liefern können.

1.3 Problemstellung und Zielsetzung

Die automatisierte Erfassung von Bäumen in urbanen Räumen ist eine zentrale Grundlage für klimaorientierte Stadtplanung und ökologische Analysen. Klassische Objekterkennungsverfahren wie Faster R-CNN stoßen dabei insbesondere auf großen Flächen an ihre Grenzen, da sie als sogenannte Two-Stage-Detector arbeiten und somit aufgrund der zweistufigen Verarbeitung vergleichsweise hohe Rechenzeiten und geringe Inferenzgeschwindigkeiten aufweisen. Um eine effizientere und zeitnahe Analyse zu ermöglichen, kommen echtzeitfähige

Objekterkennungsarchitekturen zum Einsatz, die durch hohe Verarbeitungsgeschwindigkeit und Genauigkeit überzeugen. Ziel dieser Arbeit ist es, das Potenzial aktueller Objekterkennungsmodelle für die Anwendung der automatisierten Baumdetektion in urbanen Räumen zu untersuchen und zu bewerten.
In dieser Arbeit werden die drei aktuellen Modelle YOLOv11, YOLOv12 (mit teilweiser Transformer-Integration) sowie RT-DETR (nahezu vollständig transformerbasiert) in allen verfügbaren Modellgrößen hinsichtlich ihrer Leistungsfähigkeit untersucht und miteinander verglichen. Neben der Genauigkeit der Detektion werden dabei auch die Trainingszeiten und Inferenzzeiten systematisch bestimmt, um einen umfassenden Überblick über die Performanz der Modelle auf einem eigens erstellten Baum-Datensatz zu erhalten.
Zur Evaluierung wird ein eigener Datensatz erstellt, der durch gezieltes Szenen-Sampling verschiedene städtebauliche Strukturen abbildet und damit eine differenzierte Leistungsbeurteilung im Kontext der Baumdetektion im urbanen Raum ermöglicht. Aufbauend darauf wird ein praktischer Workflow entwickelt, um das geeignetste Modell in realen Anwendungsszenarien zu testen und dessen Übertragbarkeit auf neue Gebiete zu prüfen.
Das übergeordnete Ziel dieser Arbeit ist die Entwicklung eines reproduzierbaren Workflows zur automatisierten, georeferenzierten Baumdetektion in Orthophotos, der eine präzise und effiziente Analyse großer urbaner Flächen erlaubt und damit eine fundierte Grundlage für nachhaltige Stadtplanung und Umweltmonitoring schafft.

2 Daten und Software

2.1 Datengrundlage

Zur Ausführung der Experimente wurden Daten des Landesamtes für Digitalisierung, Breitband und Vermessung (LDBV) sowie des europäischen Erdbeobachtungsprogramms Copernicus verwendet. Alle verwendeten Daten sind im Rahmen von OpenData frei verfügbar, unterliegen aber unterschiedlichen Lizenzbeschränkungen und Nutzungsbedingungen. Die Bereitstellung erfolgt jeweils über Download Portale. Die downloadbaren Daten des LDBVs werden einheitlich im Referenzsystem UTM32 (EPSG:25832) angeboten, bei Copernicus stehen je nach Produkt andere Referenzsyteme zur Auswahl (LDBV, 2025d)(EU-Copernicus Programme, 2025).

2.1.1 Digitales Orthophoto

Das Digitale Orthophoto (DOP) ist eine entzerrte, maßstabsgetreue und orthographische Darstellung der Geländeoberfläche. Als Berechnungsgrundlage dienen Luftbilder, deren Aufnahmepositionen (GNNS+IMU), innere Orientierung (Bildhauptpunkte und Brennweite), Bodenpasspunkte und das digitale Geländemodell (DGM). Mittels Aerotriangulation wird erst für jedes Luftbild die äußere Orientierung (RW, HW, h, ω ,ϕ , κ) bestimmt. Danach wird klassischerweise unter Einbezug des DGMs das Orthophoto berechnet. Der Einbezug des DGMs dient zur Entzerrung des Orthophotos. Da das DGM lediglich die Geländeoberfläche beschreibt, werden Fassaden beispielsweise umgeklappt dargestellt. Um dies zu vermeiden, wird heutzutage das Digitale Oberflächen Modell miteinbezogen, da es alle Objekte auf der Geländeoberfläche miteinbezieht. Durch das DOM werden alle Verkippungen eliminiert, das Ergebnis wird als True Orthophoto bezeichnet.
Das DOP ist mit einer Bodenauflösung von 20cm oder 40cm verfügbar und wird im TIFF-Format zum Downlaod angeboten (LDBV, 2025e).

2.1.2 Digitales Oberflächenmodel

Durch präzise orientierte Luftbilder aus der Aerotriangulation kann per dichten Bildzuordnung (Dense Matching) die Oberfläche des Geländes rekonstruiert werden. Ergebnisse der Berechnung können bildbasierte 3D-Punktwolken oder digitale Oberflächenmodelle (DOM) sein. Das digitale Oberflächenmodel beschreibt hierbei die Erdoberfläche mit allen darauf befindlichen Objekten (z.B. Gebäude und Vegetation). Die Darstellung erfolgt in Gitterform, wobei jedem Punkt ein Höhenwert zu geordnet wird. Durch die Erweiterung

eines zweidimensionalen Gitters mit einem Höhenwert, spricht man bei einem DOM von einer 2.5-D Darstellung.
Das DOM ist mit einer Bodenauflösung von 20cm oder 40cm verfügbar und wird im TIFF-Format zum Downlaod angeboten. Die Daten unterliegen der CC BY 4.0 Lizenz. (LDBV, 2025b).

2.1.3 Digitales Geländemodel

Das digitale Geländemodell (DGM) beschreibt durch eine 2.5-D Darstellung die Geländeoberfläche ohne Bebauung oder Vegetation. Das DGM wird hauptsächlich aus Laserdaten abgeleitet.
Es liegt mit einer Gitterweite von einem Meter vor und wird im TIFF- oder ASCII-Format zum Download angeboten. Die Daten unterliegen der CC BY 4.0 Lizenz. (LDBV, 2025a).

2.1.4 Verwaltungsgrenzen

Die bayerischen Verwaltungsgrenzen werden vom LDBV als Komplettdatesatz im Shape-Format angeboten. Im Komplettdatensatz sind folglich alle Grenzen als Polygone dargestellt. Jedes Polygon hat folgende Attribute: gml_id, oid, aktualit, art, name, rs, args, uebobjekt, ueboname. Es werden folgende Grenzenbeschrieben: Gemeinde, Landkreis, Regierungsbezirke und Bundesland. Die Daten unterliegen der CC BY 4.0 Lizenz (LDBV, 2025c).

2.1.5 Copernicus Urban Atlas

Der Urban Atlas (UA) ist ein Produkt des Copernicus Land Monitoring Services (CLMS). Der UA stellt hochauflösende Landnutzungs-/ Landbedeckungsdaten (LU/LC) für europäische Städte bereit und wird seit 2006 im sechs-Jahres-Turnus veröffentlicht. Die aktuelleste Veröffentlichung stammt aus dem Jahre 2018. Die Auswahl der im UA abgedeckten Städte erfolgt über die Einwohnerzahl. In der ersten Veröffentlichung 2006 lag der Schwellwert zur Aufnahme in den Atlas bei >100.000 Einwohnern. In den Produkten von 2012 und 2018 wurde der Schwellwert auf >50.000 Einwohner heruntergesetzt, womit rund 800 Städte im UA abgedeckt werden. Als Datenbasis dienen primär Satellitenbilder mit sehr hoher Auflösung (<5m). Zur Ergänzung werden Sentinel-2 Satellitenbilder, Open Street Map Daten, kommerzielle Navigationsdaten und topographische Karten herangezogen. Zusätzlich zum Atlas sind auch Straßenbaumlayer (Street Tree Layer - STL), normalisierte Gebäudehöhen (Digital Height Model - DHM) und Veränderungskartierungen (2006-2012 und 2012-2018) im UA enthalten (EEA, 2018). Die Daten unterliegen der CC BY 4.0 Lizenz.
Die Mindestkartiereinheit des Atlas beträgt 0,25 ha im urbanen Raum und 1 ha im ruralen Raum. Die Mindestkartierbreite einer Fläche darf 10 m nicht unterschreiten. Die Klassenstruktur basiert auf den Copernicus CORINE Land Cover Produkt und ist in

fünf Hauptgruppen aufgeteilt: 1.Artifical Surfaces (Siedlungen, Industrie, Verkehr, Freizeit etc.), 2.Agricultural Areas, 3.Natural/ Semi-natural Areas, 4.Wetlands und 5.Water. Die Klassifikationsgenauigkeit wird mit $\geq 85\%$ im urbanen und $\geq 80\%$ im ruralen Raum angegeben. Eine Besonderheit bei urbanen Flächen ist die Klassendifferenzierung nach Versiegelungsdichte (Soil Sealing, vgl. Abbildung 2.1). Diese beschreibt den Anteil der versiegelten Fläche innerhalb eines Polygons. Als Versiegelung zählen beispielsweise Gebäude-, Asphalt- und Betonflächen. Alle nach Versieglungsdichte unterschiedenen Klassen zählen zu der 1.Hauptgruppe. Sind Flächen zu mehr als 80% versiegelt, gelten sie als kontinuierlich versiegelt (EEA, 2018). Eine vollständige Übersicht aller Klassen ist in Tabelle 2.1

Tabelle 2.1: Urban Atlas: Klassenübersicht (Version 6.3) (EEA, 2018)

Code	Klasse	Beschreibung (gekürtzt)
1. Artificial surfaces		
11100	Continuous urban fabric	Sealing level (S.L.) > 80%
11210	Discontinuous dense urban fabric	S.L. 50–80%
11220	Discontinuous medium density urban fabric	S.L. 30–50%
11230	Discontinuous low density urban fabric	S.L. 10–30%
11240	Discontinuous very low density urban fabric	S.L. < 10%
11300	Isolated structures	Individual houses / small groups
12100	Industrial, commercial, public, military and private units	Factories, offices, barracks
12210	Fast transit roads and associated land	Motorways, service areas
12220	Other roads and associated land	Primary/secondary roads, parking
12230	Railways and associated land	Stations, tracks, freight areas
12300	Port areas	Quays, docks, transport/storage
12400	Airports	Runways, terminals, incl. military
13100	Mineral extraction and dump sites	Quarries, mines, dumps
13300	Construction sites	Land under construction
13400	Land without current use	Brownfields, unused land
14100	Green urban areas	Parks, cemeteries
14200	Sports and leisure facilities	Stadiums, sports grounds, leisure parks
2. Agricultural areas		
21000	Arable land (annual crops)	Cropland
22000	Permanent crops	Vineyards, orchards, olive groves
23000	Pastures	Grassland for grazing
24000	Complex and mixed cultivation patterns	Mixed agriculture
3. Natural and semi-natural areas		
31000	Forests	Broadleaved, coniferous, mixed
32000	Herbaceous vegetation associations	Shrubland, grassland
33000	Open spaces with little or no vegetation	Bare rock, sparsely vegetated
4. Wetlands		
40000	Wetlands	Marshes, peat bogs
5. Water		
50000	Water bodies	Rivers, lakes, reservoirs, sea
9. No data		
91000	No data (Clouds and shadows)	—
92000	No data (Missing imagery)	—

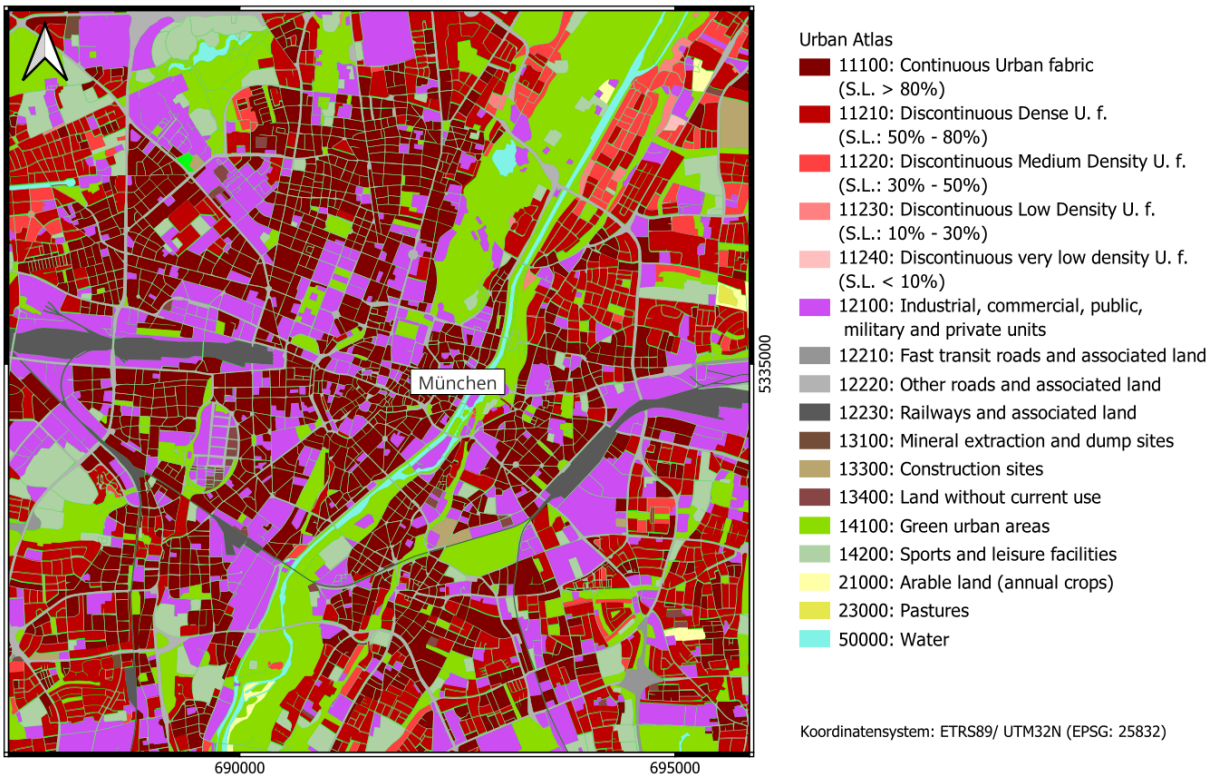

Abbildung 2.1: Visualisierung des Urban Atlas für einen Ausschnitt der Stadt München

2.1.6 Untersuchungsgebiete

Für die späteren Experimente wurden die fünf einwohnerstärksten Städte Bayerns ausgewählt. Diese sind in absteigender Reihenfolge: München, Nürnberg, Augsburg, Regensburg und Ingolstadt. Als Eingrenzung dienen die Verwaltungsgrenzen der Städte. Die genannten Städte befinden sich in verschiedenen Befliegungslosen und unterscheiden sich stark in ihrer geographischen Lage (siehe Abbildung 2.2). Alle Stadtgebiete beinhalten eine Vielzahl an Beispielen für die im UA enthaltenen Klassen, womit für die späteren Experimente eine abwechslungsreicher Datensatz erstellt werden kann. Wie in Tabelle 2.2 dargestellt, unter-

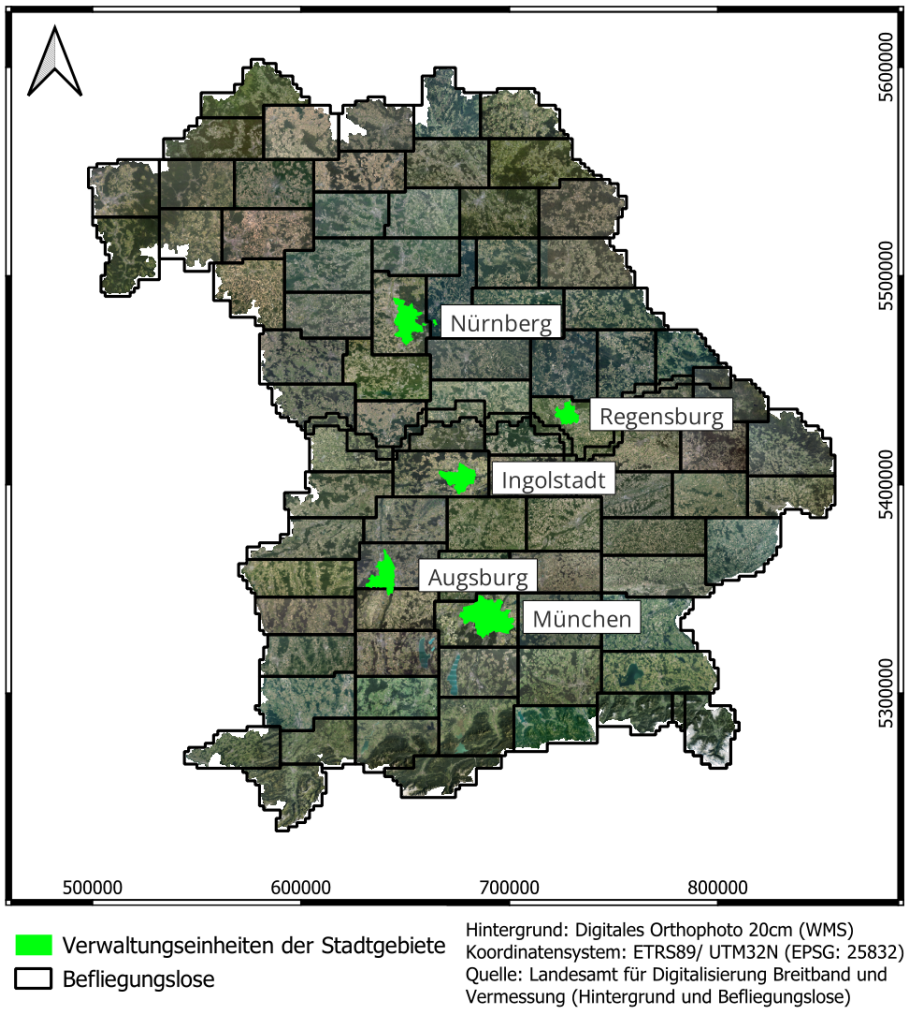

Abbildung 2.2: Untersuchungsgebiete

scheiden sich die zugrundeliegenden Befliegungslose zeitlich. Die Aufnahmedaten reichen von den Jahren 2022 bis 2024 und Monaten Mai bis September. Städte wie Nürnberg und Regensburg wurden in mehreren Teilbefliegungen erfasst, was sich in mehreren Einträgen widerspiegelt.

Tabelle 2.2: Aufnahmedatum der Befliegungslose der Untersuchungsgebiete (LDBV, 2025e)

Stadt	Aufnahmedatum
München	24.08.2024
Nürnberg	27.05.2023
	29.05.2023
	16.09.2023
Regensburg	06.06.2023
Augsburg	18.06.2022
	14.08.2022
Ingolstadt	23.08.2024

2.2 Software

Für die automatisierte Verarbeitung und Analyse der Geodaten kam eine Kombination aus frei zugänglichen Open-Source-Werkzeugen zum Einsatz. Im Zentrum stand dabei die Programmiersprache Python, die sich durch ihre Vielseitigkeit, eine große Entwicklergemeinschaft und eine breite Palette spezialisierter Bibliotheken für wissenschaftliche und geographische Anwendungen auszeichnet. Python bietet durch seine klare Syntax und modulare Struktur ideale Voraussetzungen, um komplexe Datenprozesse zu automatisieren, statistisch zu analysieren und Ergebnisse sowohl numerisch als auch visuell aufzubereiten. Ergänzend dazu wurde QGIS als grafische Plattform für die manuelle Überprüfung, Label-Erstellung und Darstellung der Ergebnisse genutzt.

2.2.1 Python Bibliotheken

Im Rahmen dieser Arbeit kamen verschiedene Python-Pakete zum Einsatz, die unterschiedliche Aufgaben in der Datenverarbeitung, Analyse und Visualisierung übernehmen:

- **NumPy (2.2.5):** Dient als grundlegendes Werkzeug für numerische Berechnungen in Python. NumPy stellt effiziente Datenstrukturen für Matrizen und Vektoren bereit und ermöglicht schnelle lineare Algebra-Operationen. Dadurch bildet es die Basis für viele weiterführende Bibliotheken der wissenschaftlichen Datenverarbeitung (Harris u. a., 2020).
- **pandas (2.2.3):**
  Ermöglicht die strukturierte Handhabung von Daten in Tabellenform (*DataFrames*) und bietet vielfältige Funktionen zur Datenbereinigung, Filterung und Aggregation. pandas wurde insbesondere zur Analyse von Metriken, Auswertung von Modellresultaten und Erstellung statistischer Kennzahlen verwendet (McKinney, 2010).
- **matplotlib (3.10.1):**
  Diente zur grafischen Aufbereitung und Visualisierung der Ergebnisse. Mit matplotlib lassen sich individuelle Diagramme und Abbildungen erstellen, die Trends,

Verteilungen und Zusammenhänge innerhalb der Daten veranschaulichen. Dadurch konnten beispielsweise Metriken der Modellperformance oder räumliche Verteilungen anschaulich dargestellt werden (Hunter, 2007).

- **seaborn (0.13.2):** Erweitert die Funktionalität von *matplotlib* um eine Vielzahl an statistischen Visualisierungsmöglichkeiten. Die Bibliothek bietet eine ansprechende Standardästhetik und erleichtert die Darstellung komplexer Datensätze durch integrierte statistische Auswertungen, etwa Regressionslinien, Boxplots oder Heatmaps. In dieser Arbeit wurde seaborn insbesondere zur explorativen Datenanalyse und zur visuellen Gegenüberstellung von Modellmetriken eingesetzt. Durch seine enge Integration mit pandas können Daten direkt aus DataFrames visualisiert werden, was den Analyseprozess deutlich beschleunigt und vereinfacht (Waskom u. a., 2021).

- **rasterio (1.4.3):**
  Eine spezialisierte Bibliothek zur Verarbeitung von Rasterdaten, insbesondere von GeoTIFFs. rasterio ermöglicht das Einlesen, Zuschneiden, Transformieren und Schreiben georeferenzierter Bilddaten und unterstützt dabei die Arbeit mit verschiedenen Koordinatensystemen. In dieser Arbeit wurde es verwendet, um Höhen- und Orthophotodaten zu verarbeiten (Gillies u. a., 2019).

- **geopandas (1.1.1):**
  Erweitert die Funktionalität von pandas um geographische Objekttypen wie Punkte, Linien und Polygone. geopandas erlaubt die einfache Durchführung räumlicher Operationen (z.B. Verschneiden, Pufferbildung, Koordinatentransformationen) und wurde zur Analyse sowie Kombination von Vektorlayern genutzt (Jordahl u. a., 2020).

- **ultralytics (8.3.123):**
  Framework zur Implementierung und Auswertung moderner Objekterkennungsmodelle, insbesondere der YOLO- und RT-DETR-Architekturen. Über ultralytics konnten Trainings- und Inferenzprozesse gesteuert sowie die Modellmetriken automatisch berechnet werden. In dieser Arbeit wurde das Framework für die Objektdetektion von Bäumen eingesetzt (Jocher, 2025).

- **tqdm (4.67.1):**
  Sorgt für übersichtliche Fortschrittsbalken bei der Ausführung von Schleifen und längeren Prozessen. Dies erleichtert die Nachvollziehbarkeit und Überwachung von Berechnungen während der Datenverarbeitung und Modellinferenz (da Costa-Luis, 2019).

- **pycocotools (2.0.10):**
  Implementiert die Evaluierung nach dem *COCO*-Standard (*Common Objects in Context*) und ermöglicht die Berechnung gängiger Objekterkennungsmetriken wie *Precision*, *Recall* und *mean Average Precision (mAP)*. Die Bibliothek wurde genutzt, um die Leistungsfähigkeit der Modelle objektiv zu quantifizieren und miteinander zu vergleichen (Lin u. a., 2025).

2.2.2 QGIS

QGIS (Quantum Geographic Information System) ist ein Open-Source-Geoinformationssystem, das umfangreiche Werkzeuge zur Analyse, Bearbeitung und Darstellung von Geodaten bereitstellt. Es unterstützt sowohl Raster- als auch Vektordatenformate und ermöglicht deren kombinierte Nutzung innerhalb eines einheitlichen Koordinatenreferenzsystems. Durch die integrierten Funktionen zur Georeferenzierung können Bilddaten (z.,B. Orthophotos oder Luftbilder) präzise in geographische Koordinatensysteme eingebunden und mit anderen Datensätzen überlagert werden. Ein zentraler Bestandteil im Kontext dieser Arbeit ist die Nutzung von QGIS für das Labelling und die Überprüfung von Referenzdaten. Durch die manuelle Erfassung und Attributierung von Objekten lassen sich Trainings- und Testdaten für Verfahren der Objektdetektion erzeugen. Dabei bietet QGIS präzise Werkzeuge zur Vektorisierung und Attributbearbeitung, wodurch eine qualitativ hochwertige Datengrundlage geschaffen wird. Für die graphische Darstellung von Geodaten stellt QGIS vielfältige Visualisierungsmöglichkeiten bereit. Rasterdaten können farblich skaliert, überlagert oder transparent dargestellt werden, während Vektordaten mit Symbolisierungen, Farbcodierungen und Beschriftungen versehen werden können. Dadurch lassen sich sowohl Analyseergebnisse als auch Modellvorhersagen anschaulich und kartographisch korrekt visualisieren. Darüber hinaus bietet QGIS zahlreiche Operationen zur Analyse und Transformation von Geodaten, etwa das Zuschneiden, Verschneiden oder Verschmelzen von Layern, die Berechnung geometrischer Attribute sowie die statistische Auswertung räumlicher Beziehungen. Damit fungiert QGIS als eines der zentralen Werkzeuge für den gesamten Datenverarbeitungsprozess – vom Labeling bis hin zur finalen Darstellung der Ergebnisse (QGIS Entwicklungsteam, 2024).

2.2.3 KI-Assistenzsystem: ChatGPT

Für die Implementierung und Optimierung der Python-Skripte wurde das KI-Assistenzsystem ChatGPT (Modelle GPT-4.5 und GPT-5) eingesetzt. ChatGPT basiert auf sogenannten großen Sprachmodellen (*Large Language Models*, LLMs), die mithilfe von Deep-Learning-Architekturen trainiert werden, um natürliche Sprache zu verstehen und zu generieren. Das Modell verwendet ein Transformer-Netzwerk, das auf Milliarden von Textbeispielen trainiert wurde, um Zusammenhänge, Syntax und semantische Muster in Sprache zu erkennen. Dadurch ist es in der Lage, auf Texteingaben kontextbezogen zu reagieren und neue Inhalte zu erzeugen, die sprachlich und logisch konsistent sind. Im Rahmen dieser Arbeit wurde ChatGPT als interaktives Werkzeug genutzt, um Python-Code zu entwickeln, zu optimieren und Fehlerquellen zu identifizieren. Die vom Modell vorgeschlagenen Lösungen dienten als Unterstützung bei Programmier- und Syntaxfragen. Alle generierten Codeabschnitte wurden vom Autor geprüft, angepasst und eigenständig in das Projekt integriert. Das System diente somit als intelligente Assistenz zur Effizienzsteigerung und Qualitätssicherung bei der Skripterstellung, ohne die wissenschaftliche Eigenleistung oder inhaltliche Verantwortung der Arbeit zu beeinträchtigen. Im Verlauf dieser Arbeit wurden die Version 4.5 und 5 verwendet (OpenAI, 2025).

3 Methodik

Im Nachfolgendem werden die für die Experimente verwendeten Methoden und entwickelten Arbeitsabläufe erläutert. Die Beschreibungen umfassen dabei die Datenvorbereitung, die Objekterkennung mit einer Auswahl an aktuellen Architekturen und deren Funktionsweisen, sowie die Aufbereitung und Evaluierung der Ergebnisse für den Gebrauch in der Praxis.
Die Datenvorbereitung beschreibt die Schaffung eines Datensatzes für das Training, die Validierung und den Test von Objekterkennungsalgorithmen. Darin enthalten sind die Auswahl der Messgebiete, das Szenen Sampling mit Hilfe des Urban Atlas von Copernicus und die gewählten Methoden zum Sampling des Datensatzes. Im Teil der Objekterkennung wird die Funktionsweise der verwendeten Architekturen grundlegend beschrieben. Die Aufbereitung und Evaluierung der Vorhersagen beinhalten gewählte Filtermethoden und die Beschreibung von Metriken, mit denen sich die Ergebnisse der durchgeführten Versuche bewerten lassen.

3.1 Datenvorbereitung

Um Objekterkennungsarchitekturen für anwendungsspezifische Aufgaben einsetzen zu können, ist die Erstellung geeigneter Datensätze entscheidend. Diese Datensätze müssen die Vielfalt der realen Bedingungen abbilden, unter denen das Modell später eingesetzt werden soll. Daher sollten sie möglichst viele und unterschiedliche Beispiele enthalten – sowohl hinsichtlich der Farb- und Texturmerkmale als auch der räumlichen Zusammensetzung und Umgebungsstrukturen. Grundsätzlich gilt: Je größer die Variation im Trainingsdatensatz, desto robuster und generalisierungsfähiger wird das Modell bei der Anwendung auf bisher unbekannte Szenen.
Im Kontext der Einzelbaumdetektion bedeutet dies, dass die Daten eine breite Palette an Baumarten, Größen, Kronenformen und Umgebungen abdecken sollten. Ebenso ist es wichtig, dass die Szenen radiometrisch variieren, also unterschiedliche Beleuchtungs- und Aufnahmebedingungen aufweisen. Nur so kann das Modell lernen, relevante Merkmale der Zielobjekte unabhängig von äußeren Einflüssen zu erkennen.
Um diese Abwechslung gezielt zu erreichen, wurde in dieser Arbeit ein Ansatz für ein Szenen-Sampling entwickelt. Dabei werden Orthophotos ausgewählt, die verschiedene städtebauliche Strukturen und Vegetationsdichten abbilden. Diese Szenen dienen anschließend als Grundlage für das Labelling, bei dem die in den Bildern enthaltenen Baumobjekte manuell oder halbautomatisch markiert werden.
Im Folgenden wird der Prozess des Szenen-Samplings detailliert beschrieben und darauf eingegangen, wie die vorbereiteten Daten anschließend für das Training, die Validierung und den Test der Modelle aufbereitet werden.

3.1.1 Auswahl der Untersuchungsgebiete

Für die Auswahl der Untersuchungsgebiete wurden gezielt einwohnerstarke Städte herangezogen. Es wird davon ausgegangen, dass diese aufgrund ihrer Größe und funktionalen Vielfalt ein breites Spektrum an städtebaulichen Strukturen und landschaftlichen Szenen abbilden. Dazu zählen dichte Innenstädte, Wohngebiete mit unterschiedlicher Bebauungsdichte sowie Gewerbeflächen. Da sich die ausgewählten Städte in unterschiedlichen Befliegungslosen befinden, wurden die zugrundeliegenden Orthophotos zu verschiedenen Zeitpunkten und unter variierenden Beleuchtungsbedingungen aufgenommen. Diese Unterschiede in der Radiometrie – etwa durch abweichende Sonnenstände, Schattenverläufe oder Aufnahmesensorik – erhöhen die Vielfalt des Datensatzes zusätzlich und fördern die Generalisierungsfähigkeit der Modelle.

3.1.2 Szenen-Sampling

Der erste Schritt des Szenen-Samplings ist die Bestimmung der digitalen Orthophotos welche innerhalb des Verwaltungsgebiet der jeweiligen Städte liegen oder eine Intersektion mit der Außengrenze haben. Die meist quadratisch vorliegenden Orthophotos werden dann weiter quadratisch unterteilt in 250x250m Kacheln. Die Unterteilung dient der Reduzierung des Labellingaufwands. Die erstellten Kacheln werden dann so gefiltert, dass nur die Kacheln erhalten bleiben die innerhalb der Verwaltungsgrenzen liegen. Resultierende Abdeckungslücken an den Rändern werden hierbei als vernachlässigbar betrachtet, da die Verwaltungsgrenzen meist auch erstes Umland beinhalten, welches keine urbanen Strukturen mehr beinhaltet. Die eigentlichen Stadtgrenzen sind in den Verwaltungsgebieten nicht enthalten.
Nach dem die Orthophotos gefiltert und unterteilt worden sind, werden die Vektordaten des Copernicus Urban Atlas (UA) mit Hilfe der Verwaltungsgrenzen auf die gewählten Untersuchungsgebiete zugeschnitten. Nach dem Zuschneiden wird ermittelt, welche Klassen innerhalb des Stadtbereichs am häufigsten auftreten. Aus den Ergebnissen werden dann Kategorien für das Sampling abgeleitet. Eine Kategorie kann aus einer oder mehrere Klassen bestehen.
Nach der Festlegung der Kategorien, erfolgt das Sampling. Dafür werden zunächst die Kachelgrenzen der Orthophotos ermittelt. Dann wird mittels der Kachelgrenze eines jeden Orthophotos der entsprechende Bereich per Differenz aus dem UA extrahiert. Nach der Extraktion werden je Kategorie die Teilflächen aufsummiert. Übertrifft die Summe der Teilfläche einer Kategorie einen festgelegten Schwellwert, so wird das entsprechende Orthophoto dieser Kategorie zugeordnet. Falls gleich mehrere Kategorien den Schwellwert erfüllen, wird das Orthophoto der Kategorie mit der höchsten Teilflächensumme zugeordnet. Überschreitet keine Teilflächensumme je Kategorie den Schwellwert, wird sie aus dem Prozess ausgeschlossen.
Nach dem alle geeigneten Kachel bestimmt worden sind, wird je Kategorie eine feste Anzahl an Kacheln zufällig ausgewählt und dem finalen Datensatz hinzugefügt. Dieser

Prozess wird für alle Städte wiederholt, womit sich für jede Stadt die selbe Anzahl an Szenen-Samples ergeben sollte.

3.1.3 Berechnung des nDOMs

Das Normalisierte Digitales Oberflächenmodell (nDOM) wird durch die Differenzbildung zwischen dem Digitalen Oberflächenmodell und dem Digitalen Geländemodell berechnet. Damit die Differenz korrekt gebildet werden kann, müssen beide Rasterdaten dieselbe räumliche Auflösung besitzen. Weichen die Auflösungen voneinander ab, können sogenannte Stufeneffekte entstehen. Um eine einheitliche Gitterweite beider Datensätze sicherzustellen, werden die Höhenwerte des jeweils gröberen Rasters durch bilineare Interpolation an die feinere Auflösung angepasst.

3.1.4 Labelling

Nach dem geeignete Szenen ermittelt worden sind, werden die darin enthaltenen Bäume gelabelt. Das Labelling von Beispiel-Objekten oder Flächen ist ein essentieller Arbeitsschritt bei der Erstellung von Datensätzen für die Objekterkennung. Jede Objekterkennungsarchitektur im DL-Bereich benötigt Beispiele zum Trainieren, Validieren und Testen. Ziel ist es möglichst viele und unterschiedliche Beispiele zu erhalten. Dabei werden alle alle Objekte einer Klasse auf einem Bild entsprechend als solche gelabelt. Wichtig ist hierbei, dass keine Objekte ausgelassen werden. Sind Objekte einer Klasse auf einem Bild nicht gelabelt, werden diese als Hintergrund wahrgenommen und können die Performance des Modells negativ beeinträchtigen. Der Vorgang des Labellings kann beispielsweise manuell oder mit Hilfe bereits trainierter Modelle erfolgen.
In dieser Arbeit wird nur die Klasse Baum gelabelt. Das Labelling erfolgt zuerst optisch manuell und später halb-automatisch. Es wird kein nDOM beim Labelling als Referenz verwendet. Normalisierte Höhen werden erst in der Aufbereitung der Vorhersageergebnisse verwendet.

Manuelles Labelling

Das manuelle Labelling erfolgt auf georeferenzierten Orthophotos in Form von GeoTIFF-Dateien. Dort werden dann alle darin enthaltenen Bäume ebenfalls georeferenziert mit achsparallelen Rechteck-Polygonen (Bounding-Boxen) optisch durch den Bearbeiter gekennzeichnet und als Vektordaten (Shape) abgelegt. Bei der Kennzeichnung muss darauf geachtet werden, dass die Bounding-Boxen die Baumobjekte so genau wie möglich umschließen. Ungenauigkeiten in der Kennzeichnung können einen negativen Einfluss auf die spätere Leistung der Vorhersage haben, da möglicherweise falsche Muster gelernt werden. Die Georeferenzierung ermöglicht die genaue räumliche Zuordnung und Reproduzierbarkeit. Die räumliche Zuordnung kann auch dafür verwendet werden, um Anhaltspunkte für andere Aufnahmejahre zu erhalten. Bei der Verwendung auf anderen Aufnahmejahren

muss jedoch auf Faktoren wie Wachstum und Verschiebungen geachtet werden. Da das manuelle Labelling sehr zeitaufwendig ist, wird dieser Schritt in dieser Arbeit nur einmalig für eine spezifische Klasse über alle Untersuchungsgebiete verwendet.

Halb-automatisches Labelling

Zur Beschleunigung des Labelling-Prozesses werden die Daten des manuellen Arbeitsschritts dazu verwendet, um ein erstes Modell zu trainieren. Dieses wird dann zur Detektion von Bäumen auf den Kacheln der anderen Klassen angewendet. Die Ergebnisse werden dann wieder optisch manuell durch den Bearbeiter kontrolliert, verbessert, gelöscht und ergänzt. Eine genaue Kontrolle der Vorhersage Ergebnisse ist hierbei essenziell, da sonst fehlerhafte Vorhersagen in das spätere Training einfließen. In dieser Arbeit wurde wiederholt das halb-automatische Labelling auf eine weitere Klasse angewendet und zusammen mit den vorangegangenen Daten wieder ein neuer Klassifikator trainiert. Durch Hinzufügen neuer Beispiele soll die Präzision der Vorhersagen Schritt für Schritt verbessert werden, was idealerweise den Arbeitsaufwand bei der Verbesserung Schritt für Schritt reduziert. Der Workflow ist in Abbildung 3.1 zusammengefasst.

3.1.5 Vorbereitung für Training und Validierung

Um den erzeugten Datensatz aus georeferenzierten GeoTIFF-Kacheln und den zugehörigen Shape-Bounding-Boxen für das Training der Modelle nutzbar zu machen, wurde ein Prozess entwickelt, der die Daten entsprechend den Anforderungen der verwendeten Architektur transformiert. Da die Architekturen RGB-Bilder in festgelegten Eingangsgrößen erwarten und die Labels in Bildkoordinaten vorliegen müssen, wird über jede Kachel und deren zugehörige Labels eine Schablone der Eingangsgröße geschoben, aus der die jeweiligen Inhalte extrahiert werden. Labels, die sich an den Kachelrändern befinden und nicht vollständig innerhalb des Ausschnitts liegen, werden dabei abgeschnitten. Anschließend erfolgt die Transformation der Labelkoordinaten von Welt- in Bildkoordinaten.
Für die Speicherung der Labels wurde das YOLO-Bounding-Box-Format verwendet (Redmon u. a., 2016), das aus fünf durch Leerzeichen getrennten Werten besteht: class, x_center, y_center, width und height. Dabei bezeichnet die Klasse das Objekt, während die folgenden vier Werte die normierten Koordinaten der Bounding-Box-Mitte sowie deren Breite und Höhe angeben. Bildausschnitte und ihre zugehörigen Labels werden mit identischen Dateinamen gespeichert, um eine eindeutige Zuordnung zu gewährleisten.
Um sicherzustellen, dass alle Trainingsobjekte mindestens einmal vollständig enthalten sind und keine relevanten Informationen an den Rändern verloren gehen, wurde zusätzlich eine Option zur Erstellung überlappender Ausschnitte implementiert. Dadurch wird eine vollständige Abdeckung der Szene gewährleistet und gleichzeitig die Robustheit des Trainingsdatensatzes erhöht.

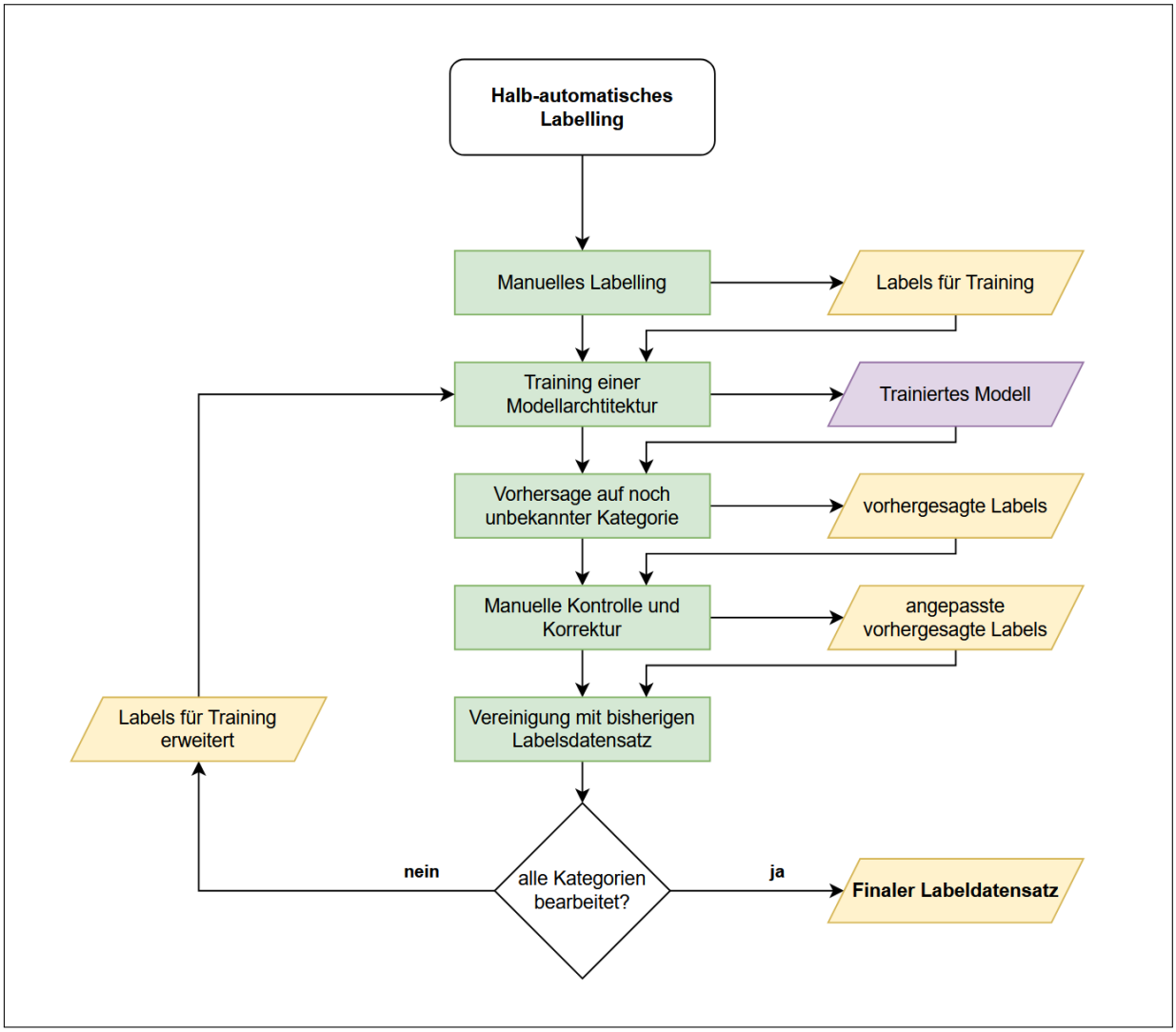

Abbildung 3.1: Workflow des halb-automatisches Labellings

3.1.6 Filterung von abgeschnittenen Labels

Um den Effekt von abgeschnittenen Labels auf die Lokalisierungsleistung der Modelle ermitteln zu können, ist eine Filterung implementiert worden. Bei der Filterung wird der Flächenanteil des abgeschnittenen Labels in Relation zu seiner ursprünglichen Gesamtfläche gestellt. Dafür wird der Quotient zwischen resultierender und ursprünglicher Gesamtfläche gebildet. Liegt der Quotient unter einem festgelegten Schwellwert, wird das Label verworfen. Die Filterung wird dann mit verschiedenen Schwellwerten durchgeführt und die Ergebnisse verglichen.

3.2 Objektbasierte Einzelbaumdetektion mittels Deep Learning

Die Detektion von Einzelbäumen in Luftbilddaten erfolgt in dieser Arbeit unter Einsatz moderner Deep-Learning-Methoden der Bildverarbeitung. Ziel ist es, einzelne Baumkronen automatisiert zu erkennen und ihre Position innerhalb des urbanen Raums präzise zu bestimmen. Hierfür werden Architekturen zur Objektdetektion eingesetzt, die eine Echtzeitauswertung hochaufgelöster Bilddaten ermöglichen. Das folgende Kapitel vermittelt zunächst die theoretischen Grundlagen der bildbasierten Objekterkennung mit Deep Learning und beschreibt anschließend vereinfacht den Aufbau sowie die Funktionsweise der in dieser Arbeit verwendeten Architekturen.

3.2.1 Grundlagen der Objekterkennung

Die Objekterkennung ist ein zentrales Teilgebiet der computerbasierten Bildanalyse und verfolgt das Ziel, mehrere Objekte innerhalb eines Bildes gleichzeitig zu lokalisieren und zu klassifizieren. Damit erweitert sie die klassische Bildklassifikation, bei der lediglich das Vorhandensein eines Objekts bestimmt wird, um die räumliche Komponente der Lokalisierung. In Abgrenzung zur semantischen Segmentierung, die jedem Pixel eine Klasse zuweist, liefert die Objekterkennung eine Begrenzungsbox (Bounding-Box) und eine zugehörige Objektklasse. Abbildung 3.2 veranschaulicht den Unterschied zwischen den grundlegenden Aufgaben der Computer Vision. Während die Bildklassifikation (a) nur das Vorkommen bestimmter Objektklassen erkennt, bestimmt die Objekterkennung (b) zusätzlich deren Position im Bild. Die semantische Segmentierung (c) weist jedem Pixel eine Objektklasse zu, ohne zwischen einzelnen Instanzen zu unterscheiden. Die Instanzsegmentierung (d) kombiniert schließlich beide Ansätze, indem sie Objekte pixelgenau segmentiert und gleichzeitig zwischen verschiedenen Instanzen derselben Klasse differenziert.

Grundlage moderner Verfahren bildet die Verarbeitung von Bilddaten mittels Convolutional Neural Networks (CNNs). Diese Netzwerke analysieren Eingabebilder, die als zweidimensionale Matrizen aus Pixelwerten vorliegen, durch eine Abfolge von Faltungs- und Pooling-Operationen (Lecun u. a., 1998; Rumelhart u. a., 1986). Die Faltungsschichten (Convolutional Layers) extrahieren lokale Merkmale wie Kanten, Formen und Texturen, indem kleine, lernbare Filter über das Eingabebild geschoben werden. Die resultierenden *Feature Maps* zeigen, in welchen Bereichen des Bildes bestimmte Merkmale erkannt wurden. Während in frühen Schichten einfache Strukturen (z. B. Kanten) erkannt werden, bilden tiefere Schichten zunehmend komplexe Muster ab. Die in den Faltungsschichten verwendeten Filter besitzen Gewichte, die während des Trainings so angepasst werden, dass relevante Bildmerkmale möglichst zuverlässig repräsentiert werden.
Nach jeder Faltungsschicht werden Aktivierungsfunktionen (z. B. ReLU (Krizhevsky u. a., 2012; Nair und Hinton, 2010)) eingesetzt, um Nichtlinearitäten einzuführen. Ohne diese wäre das Netzwerk auf lineare Abbildungen beschränkt und könnte keine komplexen Strukturen erkennen. Pooling-Schichten (z. B. Max-Pooling (Krizhevsky u. a., 2012)) reduzieren anschließend die räumliche Auflösung der Feature Maps, verringern die Anzahl der

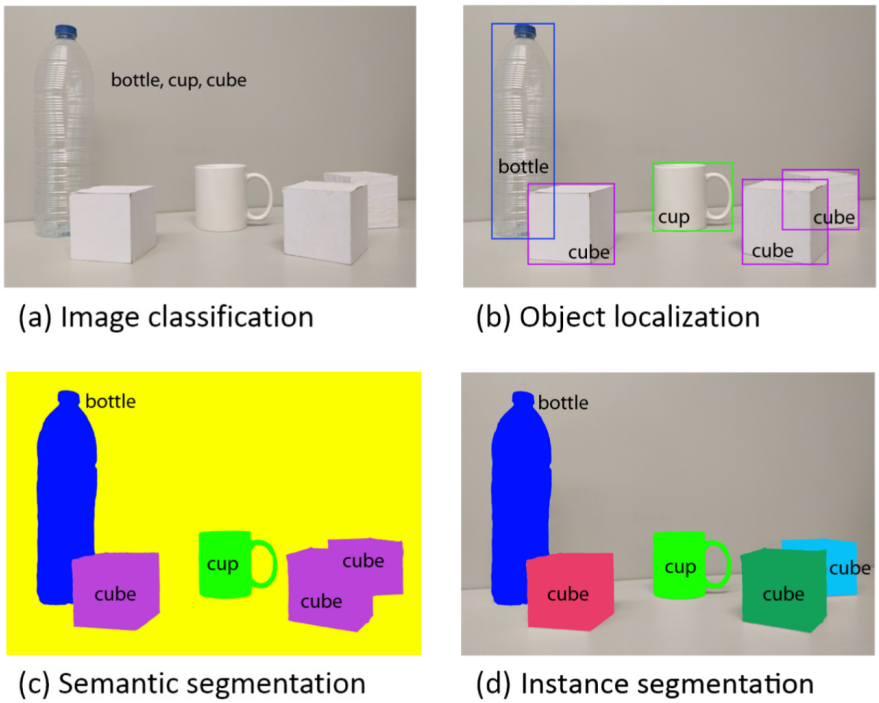

Abbildung 3.2: Vergleich der grundlegenden Aufgaben der Objekterkennung in der Computer Vision: (a) Bildklassifikation, (b) Objektlokalisierung, (c) semantische Segmentierung und (d) Instanzsegmentierung.
Quelle: Garcia-Garcia u. a. (2017).

Parameter und erhöhen die Robustheit gegenüber Verschiebungen und Skalierungen.
Während des Trainings wird das Netzwerk iterativ verbessert. In jedem Trainingsdurchgang (*Iteration*) erfolgt zunächst ein Vorwärtsdurchlauf (*Forward Pass*), bei dem das Modell für ein Mini-Batch von Eingabebildern Vorhersagen erzeugt. Anschließend wird der Verlust (*Loss*) berechnet, der misst, wie stark die Vorhersagen von den Referenzwerten (Ground Truth) abweichen. Ein hoher Loss zeigt eine schlechte, ein niedriger Loss eine gute Vorhersage an.
Im anschließenden Rückwärtsdurchlauf (*Backpropagation*) werden die Gradienten des Losses berechnet, und die Gewichte werden mithilfe eines Optimierungsverfahrens aktualisiert. Dieser Zyklus aus Vorwärtsdurchlauf, Verlustberechnung, Backpropagation und Gewichtsaktualisierung wiederholt sich über viele Epochen, bis das Modell eine stabile und generalisierbare Detektionsleistung erreicht.
Typische Verlustfunktionen in der Objekterkennung kombinieren Klassifikations- und Lokalisierungsfehler, etwa Cross-Entropy-, IoU- oder Focal-Loss (Lin u. a., 2018). Als Optimierer wird häufig *AdamW* verwendet, eine Weiterentwicklung des Adam-Algorithmus (Loshchilov und Hutter, 2019). Adam kombiniert Gradientenabstieg mit adaptiver Lernratenanpassung für jedes Gewicht, während AdamW zusätzlich einen Weight Decay-Term einführt, um Überanpassung (Overfitting) zu vermeiden. Dadurch lernt das Modell stabiler und generalisiert besser auf unbekannte Daten.

Der Aufbau moderner Objekterkennungsarchitekturen lässt sich in drei Hauptkomponenten gliedern: den *Backbone*, den *Neck* und den *Head*. Der Backbone fungiert als Feature-Extraktionsnetzwerk und erzeugt Merkmalskarten verschiedener Auflösungsebenen. Bekannte Backbones sind etwa ResNet (He u. a., 2015) oder CSPDarknet (Bochkovskiy u. a., 2020; Wang u. a., 2019). Der Neck verknüpft anschließend die Merkmalskarten unterschiedlicher Tiefe und Auflösung miteinander, um Objekte unterschiedlicher Größe zuverlässig zu erfassen – beispielsweise kleine Baumkronen oder große Baumkronen. Ein häufig eingesetztes Konzept ist hierbei das Feature Pyramid Network (FPN) (Lin u. a., 2017), dass die Multi-Scale-Fusion ermöglicht. Dabei werden semantisch tiefere Merkmale aus späteren Schichten mit höher aufgelösten Merkmalen früherer Schichten kombiniert, sodass Informationen aus unterschiedlichen Skalenebenen integriert werden (Zhao u. a., 2019). Der Head schließlich erzeugt die endgültigen Vorhersagen, also die Wahrscheinlichkeiten der Objektklassen sowie die zugehörigen Bounding-Boxes. Moderne Heads nutzen entweder vordefinierte Ankerboxen oder ankerfreie Ansätze, bei denen die Positionen direkt aus den Feature Maps gelernt werden (Zhou u. a., 2019).
Historisch lassen sich Objekterkennungsmodelle in zwei Gruppen einteilen: zweiphasige und einphasige Detektoren (Zou u. a., 2023). Zweiphasige Modelle wie Faster R-CNN (Ren u. a., 2017) generieren zunächst Region-Proposals und klassifizieren diese anschließend. Einphasige Verfahren wie YOLO (Redmon u. a., 2016) führen beide Schritte in einem Durchlauf aus und erreichen dadurch eine deutlich höhere Verarbeitungsgeschwindigkeit. Neuere Modelle integrieren zusätzlich Transformer-Komponenten, um globale Kontextinformationen innerhalb der Bildszene zu erfassen. Beispiele hierfür sind DETR (Carion u. a., 2020), RT-DETR (Zhao u. a., 2024) oder YOLOv12 (Tian u. a., 2025).

3.2.2 Verwendete Architekturen

YOLO-V11

YOLOv11 (Jocher und Qiu, 2024) ist eine Weiterentwicklung der YOLO-Reihe und zielt auf eine höhere Erkennungsgenauigkeit bei gleichzeitig reduzierter Rechenkomplexität ab. Das Modell optimiert die bestehende CNN-basierte Architektur und integriert mehrere neue Komponenten, die eine effizientere Merkmalsextraktion und eine verbesserte räumliche Fokussierung ermöglichen. Kern der Architektur ist der C3k2-Block, eine kompaktere Variante des bisherigen Cross Stage Partial-Blocks (C2f). Durch kleinere Faltungskerne werden die Berechnungen beschleunigt, während die Erkennungsleistung erhalten bleibt. Ergänzend wird der bestehende SPPF-Block (Spatial Pyramid Pooling – Fast) zur Multi-Skalen-Feature-Extraktion beibehalten und um den C2PSA-Block (Cross Stage Partial mit Parallel Spatial Attention) erweitert. Letzterer führt ein räumliches Aufmerksamkeitsmodul ein, das relevante Bildbereiche stärker gewichtet und so die Detektionsleistung insbesondere bei kleinen oder verdeckten Objekten verbessert (He u. a., 2025). Die Architektur folgt weiterhin dem klassischen Aufbau aus Backbone, Neck und Head. Der Backbone dient der Feature-Extraktion, der Neck der Fusion von Merkmalen unterschiedlicher Skalen,

und der Head der finalen Objekterkennung und Klassifikation. YOLOv11 steht in mehreren Modellgrößen zur Verfügung (n, s, m, l und x), wodurch es von leistungsschwachen Endgeräten bis zu Hochleistungsrechnern flexibel einsetzbar ist. Neben der Objekterkennung unterstützt es auch Aufgaben wie Segmentierung, Pose Estimation und orientierte Objekterkennung. Durch die Kombination aus effizienteren Blöcken und Aufmerksamkeitsmechanismen erreicht YOLOv11 im Vergleich zu früheren Versionen eine höhere mittlere Genauigkeit (mAP) bei geringerer Parameterzahl und verbessertem Laufzeitverhalten (Khanam und Hussain, 2024). Die Kernelemente sowie die Einsatzmöglichkeiten der YOLOv11-Architektur werden in Abbildung 3.3 zusammengefasst.

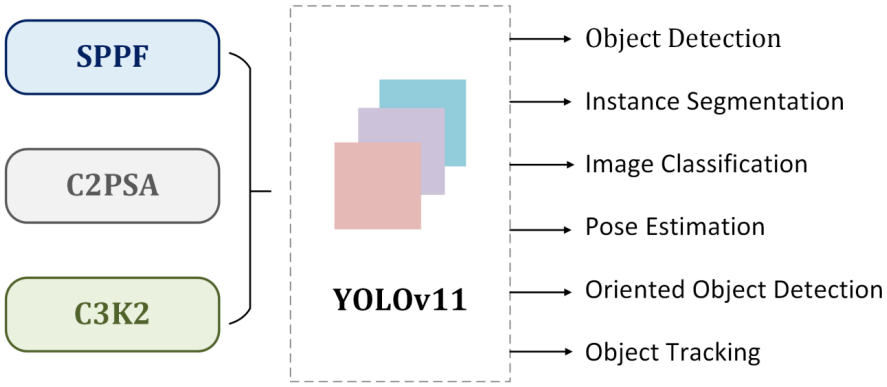

Abbildung 3.3: Kernelemente der YOLO-V11 Architektur und ihre Einsatzmöglichkeiten. Quelle: Khanam und Hussain (2024)

YOLO-V12

YOLOv12 stellt eine grundlegende Erweiterung des YOLO-Frameworks dar und führt erstmals eine attention-basierte Architektur ein, die auf Mechanismen der Vision Transformer aufbaut. Ziel ist es, die stärkere Kontextmodellierung von Attention-Strukturen mit der Effizienz konvolutionaler Netzwerke zu verbinden. Das Modell führt mit dem Area-Attention-Modul (A2) ein vereinfachtes Aufmerksamkeitsprinzip ein, das Bildbereiche in Segmente unterteilt und innerhalb dieser lokale Abhängigkeiten modelliert. Dadurch bleibt ein großes rezeptives Feld erhalten, während der Rechenaufwand gegenüber herkömmlicher Selbstaufmerksamkeit (Self-Attention) deutlich reduziert wird. Ein weiterer zentraler Bestandteil ist das Residual Efficient Layer Aggregation Network (R-ELAN), das auf der ELAN-Struktur früherer YOLO-Versionen aufbaut. R-ELAN integriert Residualverbindungen und skalierte Aggregationsmechanismen, wodurch das Training tieferer Modelle stabilisiert und der Speicherbedarf verringert wird. Zur weiteren Effizienzsteigerung wird FlashAttention eingesetzt, ein Verfahren zur speicheroptimierten Berechnung von Attention-Operationen. Weitere architektonische Anpassungen betreffen die Reduktion der Blocktiefe im Backbone, die Entfernung von Positional Encoding zugunsten eines großen Faltungskerns („Position Perceiver“) sowie eine angepasste MLP-Ratio zur besseren

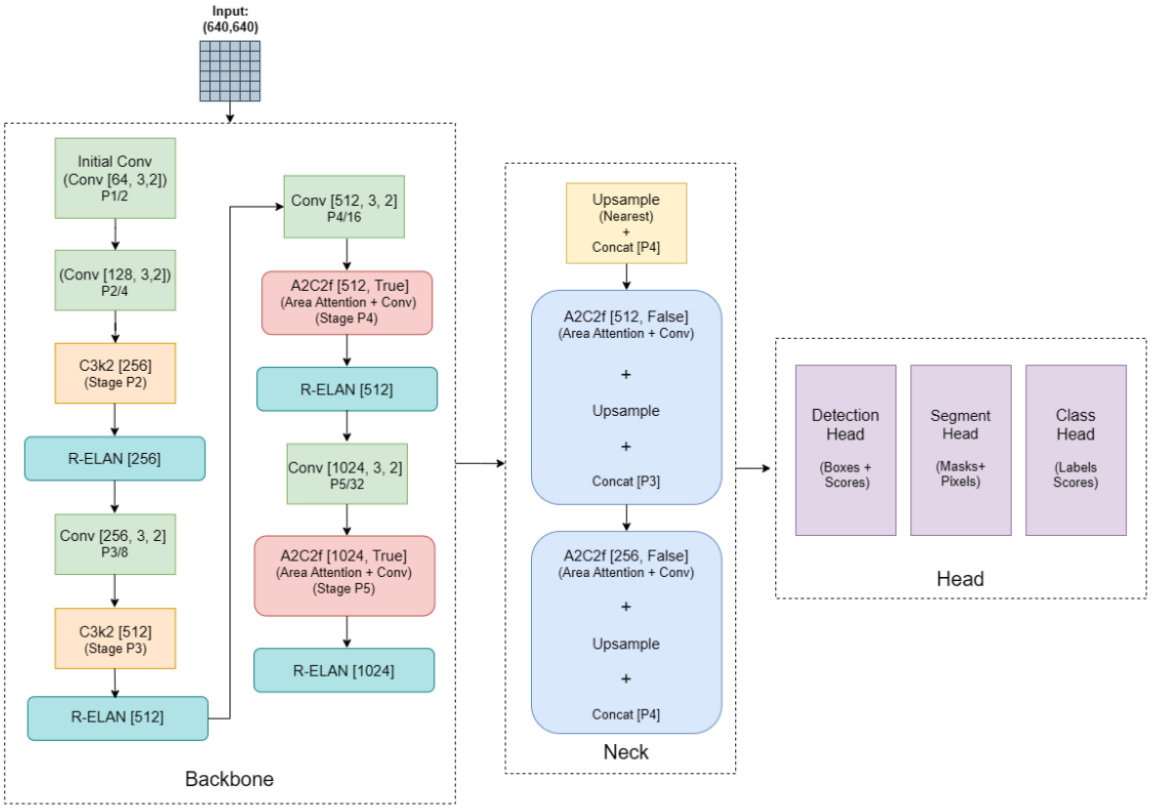

Abbildung 3.4: Aufbau der YOLO12-Architektur. Quelle:Jegham u. a. (2025).

Aufteilung der Rechenressourcen zwischen Attention- und Feedforward-Komponenten. Diese Änderungen führen zu einer verbesserten Genauigkeit bei annähernd gleicher Inferenzgeschwindigkeit wie YOLOv11. Auf dem COCO-Datensatz (Lin u. a., 2014) übertrifft YOLOv12 alle vorherigen YOLO-Versionen in der mAP, insbesondere in den kleineren Modellvarianten, und bestätigt damit die Effizienz der attention-basierten Architektur im Echtzeiteinsatz. YOLOv12 steht in mehreren Modellgrößen zur Verfügung (n, s, m, l und x), wodurch es von leistungsschwachen Endgeräten bis zu Hochleistungsrechnern flexibel einsetzbar ist. Neben der Objekterkennung unterstützt es auch Aufgaben wie Segmentierung, Pose Estimation und orientierte Objekterkennung (Tian u. a., 2025). Der Aufbau der Architektur wird in Abbildung 3.4 dargestellt.

RT-DETR

RT-DETR (Real-Time Detection Transformer) ist eine Weiterentwicklung der DETR-Architektur und stellt den ersten echtzeitfähigen end-to-end Transformer-Detektor dar (Zhao u. a., 2024). Das Modell kombiniert die Vorteile transformerbasierter Objekterkennung mit der Effizienz klassischer Echtzeitverfahren und verzichtet vollständig auf die bisher für YOLO-Modelle typische Nachbearbeitung durch Non-Maximum Suppression

(NMS). Durch den Wegfall dieses Schrittes wird die Erkennungsstabilität erhöht, da Überschneidungen direkt im Modell gelernt und aufgelöst werden.
Die Architektur von RT-DETR folgt einem Encoder-Decoder-Prinzip, das speziell für den Einsatz in Echtzeitanwendungen optimiert wurde. Zentral ist dabei der effiziente hybride Encoder, der multi-skalige Bildmerkmale verarbeitet und dabei zwei Prozesse voneinander trennt: die intra-scale Interaktion und die cross-scale Fusion (Zhao u. a., 2024). In der intra-scale Interaktion werden hochauflösende Merkmale innerhalb einer Skala durch Selbstaufmerksamkeit verknüpft, um semantische Zusammenhänge zu erfassen. Die cross-scale Fusion erfolgt hingegen über konvolutionale Fusionsblöcke, die Merkmale unterschiedlicher Skalen effizient zusammenführen. Diese Entkopplung reduziert den Rechenaufwand des Encoders deutlich und verbessert die Genauigkeit der Merkmalsaggregation.
Ein weiterer zentraler Bestandteil ist die Uncertainty-Minimal Query Selection, mit der die anfänglichen Abfragen (Queries) für den Decoder gewählt werden. Im Gegensatz zu herkömmlichen Verfahren, die nur Klassifikationsscores berücksichtigen, bewertet RT-DETR zusätzlich die Unsicherheit der Merkmale in Bezug auf Klassifikation und Lokalisierung. Nur Merkmale mit minimaler Unsicherheit werden als Ausgangs-Queries verwendet, wodurch die Qualität der Abfragen verbessert und die Konvergenz des Modells stabilisiert wird (Li u. a., 2022; Liu u. a., 2022; Zhu u. a., 2020).
Der Decoder optimiert diese Queries iterativ und erzeugt direkt ein eindeutiges Set an Bounding-Boxes und Klassen. Da jedes Objekt nur einmal vorhergesagt wird, ist keine nachträgliche Box-Unterdrückung erforderlich. Durch die modulare Struktur kann die Inferenzgeschwindigkeit flexibel angepasst werden, indem die Anzahl der Decoder-Schichten reduziert oder erweitert wird, ohne dass ein erneutes Training notwendig ist (Zhao u. a., 2024).
Insgesamt zeigt RT-DETR, dass Transformer-basierte Detektionsansätze in Echtzeitanwendungen konkurrenzfähig eingesetzt werden können. Die Kombination aus einem recheneffizienten Encoder, einer unsicherheitsbasierten Abfrageinitialisierung und dem vollständigen Verzicht auf NMS stellt einen wichtigen Schritt hin zu robusten, end-to-end trainierbaren Detektionssystemen dar (Zhao u. a., 2024). Der Aufbau der Architektur wird in Abbildung 3.5 dargestellt.

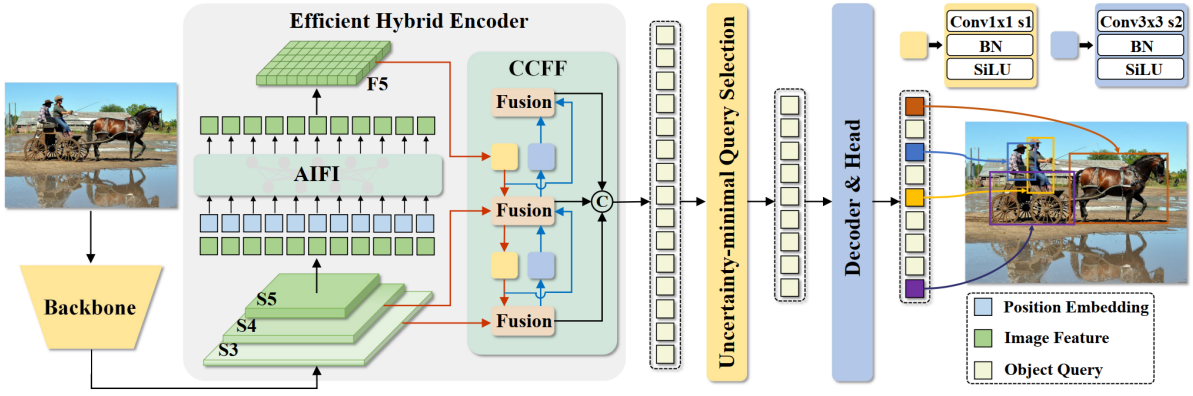

Abbildung 3.5: Aufbau der RT-DETR Architektur. Quelle: Zhao u. a. (2024).

3.2.3 Trainingsparameter

Die Auswahl geeigneter Hyperparameter ist entscheidend für die Leistungsfähigkeit und Trainingsstabilität eines neuronalen Netzes. Sie bestimmen das Lernverhalten während des Trainings und beeinflussen sowohl die Geschwindigkeit der Konvergenz als auch die Generalisierungsfähigkeit des Modells. Im Ultralytics-Framework werden zahlreiche Hyperparameter verwendet, die verschiedene Aspekte des Lernprozesses steuern – von der Anpassung der Lernrate über die Regularisierung bis hin zur Gewichtung einzelner Verlustkomponenten. Eine gezielte Abstimmung dieser Parameter ist daher unerlässlich, um ein optimales Gleichgewicht zwischen Trainingsgeschwindigkeit, Genauigkeit und Robustheit zu erreichen. Neben den Hyperparametern spielt auch die Gestaltung der Datenaugmentierung eine zentrale Rolle im Trainingsprozess. Durch gezielte Transformationen der Trainingsdaten werden zusätzliche Variationen erzeugt, die das Modell widerstandsfähiger gegenüber unterschiedlichen Aufnahmebedingungen machen. Besonders in der Objekterkennung trägt die Augmentierung dazu bei, Überanpassungen zu vermeiden und die Erkennungsleistung auf bisher ungesehene Daten zu verbessern. Im Ultralytics-Framework stehen hierzu verschiedene Augmentierungsparameter zur Verfügung, die Farb-, Geometrie- und Kompositionsveränderungen des Bildmaterials steuern und so die Vielfalt der Trainingsbeispiele erheblich erhöhen (Ultralytics, 2025).

Hyperparameter

Die Hyperparameter steuern das Lernverhalten des Modells und bestimmen maßgeblich Stabilität, Konvergenzgeschwindigkeit und des Trainingsprozesses. Desweiteren kann die Generalisierungsfähigkeit eines Modells gesteuert werden. Eine zentrale Rolle spielt die Anfangslernrate (`lr0`), die festlegt, wie stark die Modellgewichte pro Iteration angepasst werden. Sie wird über den Endfaktor der Lernrate (`lrf`) im Verlauf des Trainings schrittweise reduziert, um eine feinere Anpassung gegen Ende des Lernprozesses zu ermöglichen. Die Batchgröße (*batch* oder *mini-batch*) bestimmt wie viele Beispielbilder vom Modell verarbeitet werden, bevor die Gewichte des Modells angepasst werden. Je Epoche werden alle Beispiele in Batches zum Trainieren des Modells verarbeitet. Kleine Batch-Zahlen führen zu schnellen sprunghaften Anpassungen, während große zu langsameren und stabileren Anpassungen führen. Der Trägheitsparameter (`momentum`) wirkt dabei als Glättungsfaktor, der sprunghafte Gewichtsänderungen verhindert, während der Regularisierungsfaktor (`weight_decay`) große Gewichtswerte bestraft und so einer Überanpassung (Overfitting) vorbeugt. In den ersten Trainingsphasen wird die Lernrate durch die Anzahl der Aufwärm-Epochen (`warmup_epochs`) kontrolliert erhöht. Dieses sogenannte Warm-up stabilisiert den Lernprozess und verhindert, dass große Gradienten zu Beginn das Modell destabilisieren. Die Gewichtung der einzelnen Verlustkomponenten erfolgt über den Box-Verlustfaktor (`box`), den Klassifikationsverlustfaktor (`cls`) und den Distribution-Focal-Loss-Faktor (`dfl`). Der Box-Verlust optimiert die räumliche Genauigkeit der Bounding-Boxen, der Klassifikationsverlust bewertet die Übereinstimmung zwischen vorhergesagten und tatsächlichen Objektklassen, und der Distribution Focal Loss verfeinert die Objektgrenzen, indem er eine

fein aufgelöste Verteilung entlang der Boxränder lernt. Schließlich regelt der Parameter Abschaltzeitpunkt der Mosaic-Augmentation (`close_mosaic`), in wie vielen letzten Epochen des Trainings die Mosaikaugmentation deaktiviert wird, um eine stabilere Endkonvergenz auf realistischeren Einzelbildern zu gewährleisten.

Augmentierungsparameter

Zur Erhöhung der Robustheit und Generalisierbarkeit der Modelle werden im Ultralytics-Framework verschiedene Datenaugmentierungen eingesetzt. Sie erzeugen künstliche Variationen der Trainingsbilder und simulieren unterschiedliche Aufnahmebedingungen. Farbbezogene Transformationen umfassen die Farbtonänderung (`hsv_h`), die Sättigungsvariation (`hsv_s`) und die Helligkeitsanpassung (`hsv_v`). Diese Modifikationen sorgen dafür, dass das Modell auch bei wechselnder Farbgebung, Belichtung und Sättigung zuverlässig arbeitet. Geometrische Transformationen umfassen die Rotationsveränderung (`degrees`), die Translation (`translate`) zur zufälligen Verschiebung des Bildausschnitts, die Skalierungsanpassung (`scale`) zur Simulation unterschiedlicher Objektgrößen sowie die Schertransformation (`shear`), die leichte perspektivische Verzerrungen einführt. Spiegelungsoperationen, sowohl vertikal (`flipud`) als auch horizontal (`fliplr`), erweitern die Trainingsvariabilität und verhindern eine zu starke Orientierungsspezifität der Merkmalsextraktion. Besonders wirkungsvoll sind zusammengesetzte Augmentierungen wie die Mosaikaugmentation (`mosaic`), bei der vier verschiedene Trainingsbilder zu einem kombiniert werden. Dadurch lernt das Modell, Objekte auch in komplexen und mehrskaligen Szenen korrekt zu erkennen. Die MixUp-Augmentation (`mixup`) überlagert zwei Bilder mitsamt ihrer Labels, wodurch die Robustheit gegenüber überlappenden Objekten verbessert wird. Schließlich ermöglicht die Copy-Paste-Technik (`copy_paste`) das gezielte Einfügen von Objekten aus einem Bild in ein anderes, um die Vielfalt der Szenen und Hintergründe zu erhöhen. Diese Augmentierungen tragen insgesamt dazu bei, dass das trainierte Modell auch unter variierenden Bedingungen – etwa unterschiedlichen Schattenmustern, Vegetationsfarben oder Aufnahmewinkeln – eine stabile und verlässliche Detektionsleistung erzielt.

3.2.4 Modeltuning

Im Rahmen des Model-Tunings werden zunächst die empfohlenen Standardparameter nach Ultralytics (2025) verwendet. Anschließend erfolgt eine gezielte Anpassung der Hyperparameter, um mit dem zugrunde liegenden Datensatz eine stabile Konvergenz zu erreichen, ohne dass es zu einer Überanpassung (Overfitting) kommt. Darüber hinaus müssen die Eingangsgröße und die Batch-Size so gewählt werden, dass sie den Hardware-Beschränkungen des Systems entsprechen und den verfügbaren GPU-Speicher nicht überschreiten, da dies das Training erheblich verlangsamen würde (Auslagerung auf langsamere CPU und RAM). Je nach Komplexität des Datensatzes kann es erforderlich sein, bestimmte Parameter individuell anzupassen. In manchen Fällen genügt beispielsweise bereits eine geringe Anzahl an Epochen, um eine zufriedenstellende Modellleistung zu erzielen, während komplexere Datensätze längere Trainingsphasen benötigen. Zur Beurteilung der Modellleistung dienen

die zuvor eingeführten Kennzahlen, insbesondere der *mean Average Precision* (mAP) zur Bewertung der Lokalisierungsgenauigkeit sowie der *F1-Score* als kombinierte Metrik aus *Precision* und *Recall* (siehe Kaptiel 3.2.5).
Nachdem eine geeignete Hyperparameterkonfiguration gefunden wurde, kann durch gezielte Datenaugmentierung versucht werden, die Generalisierungsfähigkeit auf den Validierungsdaten weiter zu verbessern. Um optimale Einstellungen zu identifizieren, werden verschiedene Kombinationen der Augmentierungsparameter systematisch getestet. Eine Baseline-Messung ohne Augmentierung ist hilfreich, um zu beurteilen, inwieweit die Standard-Augmentierungsparameter des Ultralytics-Frameworks die Modellleistung beeinflussen.

3.2.5 Metriken

Um die Leistung der vorgestellten Architekturen und deren unterschiedlichen Modellgrößen bewerten zu können, werden im Folgendem die verwendeten Kennzahlen und Metriken vorgestellt (He u. a., 2025).

Grundlagen

- **True Positives (TP) – richtige positive Treffer:**
  Eine Vorhersage wird als *True Positive* gewertet, wenn ein detektierter Baum in der Vorhersage mit einem tatsächlichen Baum in der Referenzdatenbasis übereinstimmt. Dies bedeutet, dass das Modell einen Baum korrekt erkannt hat, beispielsweise wenn die vorhergesagte Position oder Bounding-Box ausreichend mit der Referenz (*Ground Truth*) überlappt, gemäß einem definierten Schwellenwert (z. B. Intersection over Union, IoU ≥ 0.5).

- **False Positives (FP) – falsche positive Treffer:**
  False Positives entstehen, wenn das Modell ein Objekt (z. B. einen Baum) detektiert, das in der Referenz gar nicht vorhanden ist. In der Praxis bedeutet dies, dass das Modell fälschlicherweise einen Baum erkennt, wo sich tatsächlich keiner befindet – etwa durch Schatten, Bodenstrukturen oder Kronenüberlappungen, die als Baum interpretiert werden.

- **True Negatives (TN) – richtige negative Treffer:**
  Ein *True Negative* liegt vor, wenn das Modell korrekt erkennt, dass in einem bestimmten Bereich **kein** Baum vorhanden ist, und dies auch mit der Referenz übereinstimmt. In der Regel spielt diese Kategorie bei der Objektdetektion jedoch eine untergeordnete Rolle, da nicht detektierte Hintergrundbereiche (z. B. Boden, Straßen oder Wasserflächen) sehr zahlreich sind und die Bewertung stark verzerren können.

- **False Negatives (FN) – falsche negative Treffer:**
  False Negatives entstehen, wenn in der Referenz ein Objekt (z. B. ein Baum) vorhanden ist, das vom Modell nicht erkannt wurde. Das Modell hat also einen existierenden Baum übersehen – zum Beispiel aufgrund von Abschattung, geringer Auflösung oder Ähnlichkeit mit der Umgebung.

- **Precision (Genauigkeit):**
  Die Precision misst, welcher Anteil der vom Modell erkannten Objekte tatsächlich korrekt ist. Eine hohe Precision bedeutet, dass nur wenige falsche Positiv-Ergebnisse vorliegen. Sie wird berechnet als:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall (Trefferquote / Sensitivität):**
  Der Recall gibt an, welcher Anteil der tatsächlich vorhandenen Objekte vom Modell erkannt wurde. Ein hoher Recall bedeutet, dass fast alle Objekte erkannt werden. Berechnung:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score:**
  Der F1-Score ist das harmonische Mittel aus Precision und Recall. Er bietet eine ausgewogene Kennzahl, wenn sowohl Precision als auch Recall berücksichtigt werden sollen:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Metriken in der Objekterkennung

- **IoU (Intersection over Union):**
  Diese Kennzahl beschreibt das Verhältnis zwischen der Schnittmenge (Überlappung, engl. Intersection) und der Vereinigung (engl. Union) zweier Flächen. Dieses Verhältnis wird in der Objekterkennung oft als Schwellwert verwendet. Überschreite die IoU zwischen Vorhersage und Referenz einen festgelegten IoU-Wert, wird die Vorhersage beispielsweise als richtig vorhergesagt gezählt.

- **mAP50 (mean Average Precision):**
  Die „mean Average Precision“beschreibt die mittlere durschnittliche Präzision einer Klasse oder über mehrere Klassen hinweg. Die Bedingung, dass eine Vorhersage als korrekt angenommen wird, liegt hierbei einem IoU-Wert von größer gleich 50%.

- **mAP50-95:**
  Diese Metrik beschreibt die durchschnittliche Präzision über verschiedenen IoU-Schwellen. Damit ist die Bewertung der Lokalisierungsgenauigkeit wesentlich strenger als bei mAP50 oder mAP75 (IoU=75%).

Berechnungen der Metriken für die Ergebnisse der praktischen Anwendung

Zur Berechnung der Evaluationsmetriken wurde eine eigene Funktion entwickelt, die sich an der `pycocotools`-Evaluierung des COCO-Datensatzes orientiert. Die Implementierung wurde mit Unterstützung von ChatGPT erstellt und für die Anforderungen dieser Arbeit angepasst. Grundlage bildet die Berechnung der IoU, der das Verhältnis der Schnitt- zur Vereinigungsfläche zwischen vorhergesagten und referenzierten Baumgeometrien beschreibt. Für jedes Vorhersageobjekt wird das Referenzobjekt mit der größten Überlappung bestimmt. Liegt der IoU-Wert über einer festgelegten Schwelle (Standard: 0.5), wird das Objekt als TP gezählt, andernfalls als FP. Jedes Referenzobjekt kann nur einmal zugeordnet werden, um Mehrfachzählungen zu vermeiden. Auf Basis der kumulativen TP- und FP-Zahlen werden Precision-Recall-Kurven erzeugt, deren Verlauf geglättet wird, um eine monoton fallende Präzisionsfunktion zu gewährleisten. Daraus werden die Precision-, Recall- und F1-Werte bei einem IoU von 0.5 bestimmt (Precision@50, Recall@50, F1@50). Zudem wird die mittlere Average Precision (mAP@50–95) berechnet, indem die Average Precision über mehrere IoU-Schwellen von 0.5 bis 0.95 (Schrittweite 0.05) gemittelt wird. Die Auswertung erfolgt für jede Testkachel einzeln und Kategorie gemittelt.

3.3 Workflow für die praktische Anwendung

Ziel des entwickelten Workflows ist die automatische Detektion von Baumobjekten in georeferenzierten Luftbilddaten wie digitale Orthophotos unter Verwendung eines zuvor trainierten Objekterkennungsmodells. Das Verfahren erzeugt als Ergebnis georeferenzierte Bounding-Boxen im Referenzsystem der übergebenen DOPs und ist so konzipiert, dass es flexibel auf unterschiedlich großen Untersuchungsgebieten angewendet werden kann. Die georeferenzierten Bounding-Boxen werden als Polygone im Shape-Format ausgegeben.

3.3.1 Vorhersage mit überlappenden Ausschnitten

Zu Beginn des Verfahrens werden aus den übergebenen Orthophotos sich überlappende Ausschnitte in der Eingangsgröße des Modells erzeugt. Die Überlappungen dienen dazu, dass jedes Objekt mindestens einmal vollständig erfasst wird. Danach erfolgt die Vorhersage auf den Ausschnitten. Jeder Ausschnitt wird einzeln verarbeitet und dessen Ergebnisse in das Referenzsystem der Eingangsdaten transformiert. Die Ergebnisse je Ausschnitt werden dann so lange gesammelt, bis die komplette Fläche des Eingangsbild abgedeckt ist.

3.3.2 Filterung

Da sich durch die Überlappungen abgeschnittene oder doppelte Vorhersagen ergeben, müssen diese noch gefiltert werden. Zur Bereinigung der Ergebnisse werden diese anhand der über ihre geometrischen Eigenschaften, durch Non-Maximum-Suppression und abschließend der relativen Höhe in Verbindung mit Gebäudekontouren gefiltert.

Geometrische Eigenschaften

Bei der Filterung über geometrische Eigenschaften werden die resultierenden Polygone über Seitenverhältnisse und Mindestseitenlänge gefiltert. Bei der Filterung nach Seitenverhältnissen werden die beiden Seitenlängen der rechteckigen Bounding-Boxen bestimmt. Danach werden die zwei möglichen Seitenverhältnisse (a/b,b/a) abgeleitet. Überschreitet eines der beiden Seitenverhältnisse den festgelegten Schwellwert, so werden die entsprechenden Boxen verworfen. Die abgeleiteten Seitenlängen werden dann im folgenden Schritt dazu verwendet, um Boxen mit zu kleinen Seitenlängen zu filtern. Unterschreitet eine der beiden Seiten den festgelegten Schwellwert, so werden diese verworfen.

Non-Maximum-Suppression

Die Non-Maximum Suppression (NMS) ist ein Verfahren, das in der Objekterkennung eingesetzt wird, um doppelte oder stark überlappende Vorhersagen desselben Objekts zu unterdrücken. Dabei wird sichergestellt, dass für jedes erkannte Objekt nur eine einzige, möglichst präzise Bounding-Box erhalten bleibt. Dafür werden zunächst alle vorhergesagten Bounding-Boxen nach ihrer Konfidenz, also der Wahrscheinlichkeit, dass sie tatsächlich ein Objekt enthalten, sortiert. Anschließend wird die Box mit der höchsten Konfidenz ausgewählt und als Referenzbox betrachtet. Nun wird überprüft, welche der verbleibenden Boxen eine Überlappung (Intersection over Union, IoU) mit dieser Referenzbox aufweisen. Für jede dieser Boxen wird der IoU-Wert berechnet, der angibt, wie stark sich zwei Boxen überschneiden. Liegt der IoU-Wert einer Vergleichsbox über einem vorher definierten Schwellwert (z. B. 0.5), wird angenommen, dass sie dasselbe Objekt beschreibt, und diese Box wird daher unterdrückt bzw. verworfen. Boxen mit einer geringeren Überlappung bleiben erhalten, da sie vermutlich andere Objekte darstellen. Nachdem alle überlappenden

Boxen gefiltert wurden, wird die nächste Box mit der höchsten verbliebenen Konfidenz ausgewählt, und der Prozess wiederholt sich, bis keine Boxen mehr übrig sind. Auf diese Weise sorgt die Non-Maximum Suppression dafür, dass pro erkanntem Objekt nur eine Bounding-Box mit der höchsten Verlässlichkeit beibehalten wird.
Als Grundlage für die Implementierung wurde der in Fast R-CNN verwendete Non-Maximum-Suppression-Algorithmus nach Girshick (2015) übernommen. Dieser Ansatz sortiert die detektierten Objekte nach ihrer Konfidenz und unterdrückt anschließend überlappende Detektionen oberhalb eines IoU-Schwellwerts.

nDOM-Filterung

Nach der NMS-Filterung werden die Ergebnisse anhand relativer Höhenwerte gefiltert. Zusätzlich zum nDOM kommen hierbei Gebäudeumringe zum Einsatz. Ziel dieser Filterung ist es, niedrige Vegetation zu entfernen, beispielsweise Sträucher oder frisch gepflanzte Bäume. Frühere Experimente zeigten zudem, dass innerhalb dichter Baumgruppen auch Schattenbereiche fälschlicherweise als Bäume detektiert wurden. Daher wird für die Höhenprüfung nicht die gesamte Bounding-Box-Fläche herangezogen.
Da die DOMs photogrammetrisch erzeugt wurden und somit als 2,5D-Modelle vorliegen, weisen sie einen gewissen „Schleier-Effekt“ auf: Die Oberfläche wirkt geglättet, und an Objektgrenzen können Höhenwerte überlagert werden. Befinden sich Vorhersagen an Baumrändern oder überlappen benachbarte Baumkronen, führt die Auswertung der gesamten Bounding-Box zu verfälschten Höhenwerten durch Einfluss höherer Nachbarbäume. Um dies zu vermeiden, werden die Seitenlängen der Bounding-Box daher mit einem Verkleinerungsfaktor multipliziert. Bei einem Faktor von 0,3 wird beispielsweise nur etwa 9% der ursprünglichen Fläche betrachtet. Die kleinere Fläche um das Boxzentrum ermöglicht es dennoch, Höhenmaxima zu erfassen, die nicht exakt im Mittelpunkt liegen.
Da sich Bounding-Boxen in urbanen Bereichen mit Gebäuden überschneiden können, wird das nDOM vorab entsprechend angepasst. Hierzu werden die lokalen Gebäudeumringe gepuffert und alle nDOM-Höhen innerhalb dieses Bereichs auf Null gesetzt. Die Pufferung dient dazu, Dachüberstände oder kleinere Anbauten zu berücksichtigen; ein typischer Wert beträgt hierfür 2m.
Dieser leicht verengte Höhen-Filterprozess verbessert die Trennung zwischen Bäumen und niedriger Vegetation sowie Gebäudestrukturen und reduziert Fehlklassifikationen an Objektgrenzen.

3.3.3 Zusammenfassung des Workflows

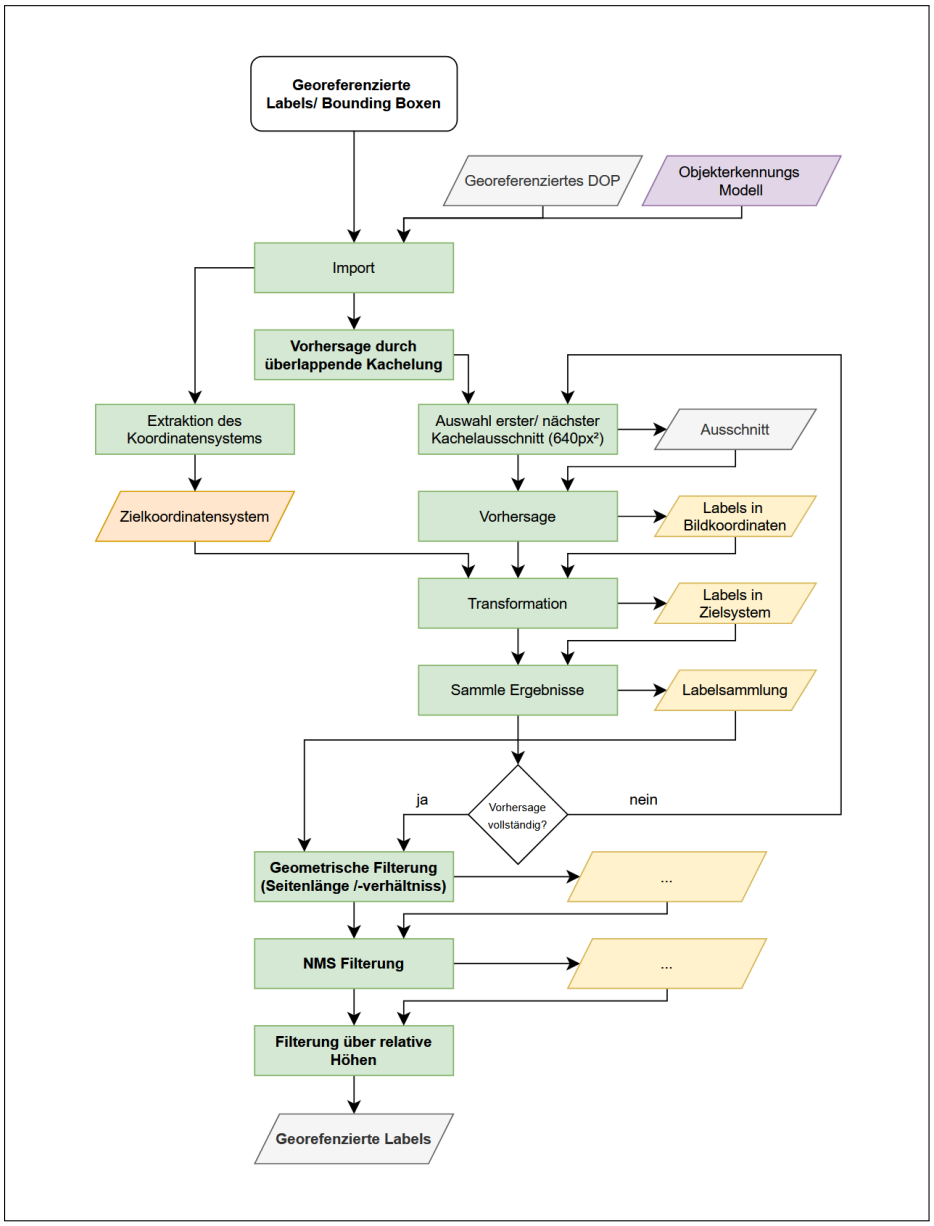

Abbildung 3.6: Zusammenfassung des Workflows für die praktische Anwendung auf Orthophotos

4 Experimente und Ergebnisse

Im Folgenden werden die durchgeführten Experimente dokumentiert. Diese umfassen den gesamten Prozess von der Datenaufbereitung über den Vergleich verschiedener Modellarchitekturen bis hin zur praktischen Anwendung auf zwei Testgebieten. Ziel der Experimente ist es, die Leistungsfähigkeit moderner Objekterkennungsmodelle für die Detektion einzelner Bäume in urbanen Umgebungen zu bewerten und eine geeignete Architektur für den praktischen Einsatz zu identifizieren.

4.1 Datenvorbereitung

Die Arbeitsschritte der Datenvorbereitung umfassen die Auswahl der Untersuchungsgebiete, das Szenen-Sampling, das Labelling sowie die abschließende Aufteilung des erzeugten Datensatzes für den reinen Leistungsvergleich und den Praxistest. Die Vorgehensweise bei den Experimente und deren Ergebnisse werden im Folgendem beschrieben.

4.1.1 Auswahl der Untersuchungsgebiete

In dieser Arbeit wurden für die Experimente die fünf einwohnerstärksten Städte Bayerns als Untersuchungsgebiet und Datengrundlage ausgewählt. Diese sind in absteigender Reihenfolge: München, Nürnberg, Augsburg, Regensburg und Ingolstadt. Tabelle 4.1 fasst die grundlegenden demografischen und geographischen Kennwerte dieser Städte zusammen. Sie enthält die Einwohnerzahl, die Gesamtfläche der jeweiligen Verwaltungsgebiete in Quadratkilometern sowie die daraus berechnete Einwohnerdichte (StaBuL, 2024). Die Flächen basieren auf kartesischen UTM32-Daten und werden in Quadratkilometern angegeben.

Tabelle 4.1: Einwohnerzahlen, Flächen und Einwohnerdichten der größten kreisfreien Städte Bayerns (nach LDBV, 2025c und StaBuL, 2024).

Stadt	**Einwohnerzahl**	**Fläche des Verwaltungsgebiets (km²)**	**Einwohnerdichte (E/km²)**
München	1 505 005	310.7	4 843.4
Nürnberg	529 508	186.4	2 840.8
Augsburg	301 105	146.8	2 051.0
Regensburg	151 389	80.9	1 871.3
Ingolstadt	141 185	133.4	1 058.8

Die Auswahl der Städte erfolgte mit dem Ziel, unterschiedliche urbane Strukturen und Versiegelungsgrade innerhalb Bayerns abzudecken. Die ausgewählten Städte bieten aufgrund ihrer großen Fläche eine große Auswahl an urbanen Strukturen auf. Diese Variation der städtischen Strukturen ist für das Training und die Validierung objektbasierter Deep-Learning-Modelle zur Baumdetektion von Bedeutung, da sie eine bessere Generalisierbarkeit auf unterschiedliche urbane Umgebungen ermöglicht.
Nach der Auswahl der Gebiete für die Datenbasis wurden die Orthophotos, DGMs und DOMs flächendeckend für die jeweiligen Verwaltungsgrenzen der Städte heruntergeladen.

4.1.2 Szenen-Sampling

Um für das Szenen-Sampling geeignete Kategorien bilden zu können, wurde zunächst eine mathematische und visuelle Analyse der im Stadtkontext am häufigsten vorkommenden Landnutzungsklassen durchgeführt. Grundlage dieser Analyse bildeten die Verwaltungsgebiete des LDBV sowie die Landbedeckungsdaten des Copernicus Urban Atlas. Im Rahmen der mathematischen Auswertung wurden die Teilflächen jeder vorhandenen Klasse aufsummiert und anschließend überprüft, welche Klassen den größten Flächenanteil einnehmen. Da die Verwaltungsgrenzen nicht exakt den Stadtgrenzen entsprechen, umfassen die Statistiken auch Gebiete, die außerhalb des eigentlichen urbanen Raums liegen. Ein exemplarischer Blick auf die Flächenzusammensetzung Münchens anhand der Urban-Atlas-Klassen (siehe Abbildung 4.1) zeigt, dass hochversiegelte Flächen einen besonders großen Anteil an der Gesamtfläche ausmachen. An zweiter Stelle folgen Industrie- und Gewerbeflächen. Auf Grundlage der optischen und mathematischen Analyseergebnisse wurden schließlich jene Klassen ausgewählt, die im städtebaulichen und klimatischen Kontext die größte Relevanz besitzen. Landwirtschaftliche Flächen, Wälder und Sportanlagen wurden aufgrund ihrer geringeren Bedeutung im urbanen Zusammenhang nicht berücksichtigt.

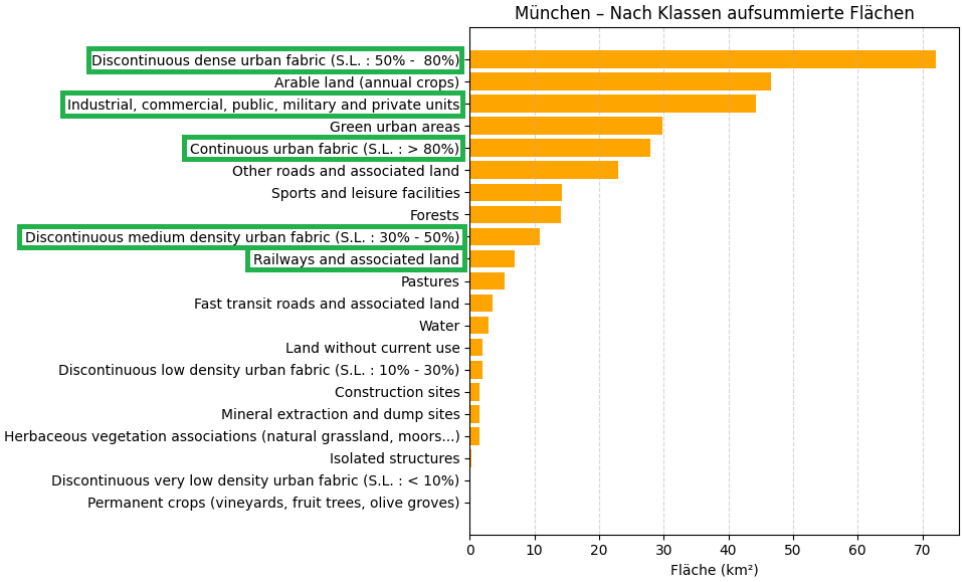

Abbildung 4.1: Flächenanteile der Urban-Atlas-Klassen im Verwaltungsgebiet München als Grundlage für die spätere Auswahl geeigneter Kategorien beim Szenen-Sampling. Grün umrandet sind die Klassen, die in die weiteren Analysen einbezogen wurden.

Aus den Erkenntnissen der Analyse wurden vier Kategorien gebildet. Die ersten drei Kategorien sind identisch mit den Klassen für Urbane Strukturen mit Bodenversiegelungsinformationen. Sie beinhalten nicht durchgängige urbane Strukturen mit 30-80% Bodenversieglung und durchgängige urbane Strukturen mit einer Bodenversieglung größer 80%. Als vierte Kategorie wurden die Kategorien Industrie und Gewerbe, sowie Eisenbahnen und dazugehöriges Land ausgewählt. Im Folgendem sind die Kategorien und deren Klassen samt zusätzlicher Informationen aufgelistet:

- **Kategorie 1:** Continuous Urban Fabric – S.L. > 80 % (UA-Code: 1.1.1)
- **Kategorie 2:** Discontinuous Urban Fabric – S.L. 50 %–80 (UA-Code: 1.1.2.1)
- **Kategorie 3:** Discontinuous Urban Fabric – S.L. 30 %–50 (UA-Code: 1.1.2.2)
- **Kategorie 4:** Industrial, Commercial, Public, Military and Private Units + Railways and Associated Land (UA-Codes: 1.2.1 und 1.2.2.3)

Nach der Kategorisierung wurde für jede Kachel geprüft, ob eine der o.g. Kategorien mindestens 50% der Gesamtfläche einnimmt. Wenn eine Kachel diese Bedingung erfüllt hat, dann wurde sie für das spätere Sampling in den Datenpool der jeweiligen Kategorie aufgenommen. Nachdem potenzielle Kacheln ermittelt worden sind, wurden je Stadt und Kategorie zufällig zehn Beispiele aus dem Datenpool ausgewählt und dem finalen Datensatz hinzugefügt. Aus fünf Städten, vier Kategorien mit je zehn Beispiel wurde damit ein Datensatz mit 200 Kacheln erzeugt.

In Tabelle 4.2 sind die nach dem Szenen-Sampling resultierenden Flächenanteile der Urban-Atlas-Klassen für die einzelnen Städte dargestellt. Dabei wurden die Teilflächen aller vorkommenden Klassen aufsummiert und ins Verhältnis zur Gesamtfläche der ausgewählten Kacheln je Stadt gesetzt. Wie zu erwarten, dominieren in allen Städten die Flächen der zuvor definierten urbanen Kategorien.

Tabelle 4.2: Flächenanteile der Kategorien nach Szenen-Sampling je Stadt(in %)

Stadt	**Kat. 1**	**Kat. 2**	**Kat. 3**	**Kat. 4**	**Sonstige**
München	18.4	24.3	17.8	34.3	5.2
Nürnberg	17.9	22.3	23.6	28.6	7.6
Augsburg	28.7	23.0	6.8	31.5	10.0
Regensburg	16.4	21.9	18.6	32.3	10.8
Ingolstadt	15.4	23.7	23.7	23.6	13.6
Mittelwert	**19.4**	**23.0**	**18.1**	**30.1**	**9.4**

Zusätzlich ist zu erwähnen, dass für die Stadt Augsburg nicht genügend Beispiele für die Kategorie 3 gefunden wurden. Dort konnten nur drei Kacheln gefunden werden, welche die Kriterien des Samplings erfüllen. Zum Ausgleich der fehlenden Kacheln wurden zusätzliche sieben Kacheln aus der Kategorie 3 von der Stadt München dem Datensatz hinzugefügt.

4.1.3 Labelling

Für den initialen Aufbau des Datensatzes wurden Baum-Objekte manuell in QGIS anhand der im Szenen-Sampling ausgewählten Orthophotos annotiert. Das Labeling erfolgte durch das Einzeichnen von Bounding Boxes um Bäume, basierend auf einer ausschließlich optischen Einschätzung. Auf eine zusätzliche Prüfung über Höheninformationen wurde in dieser Phase bewusst verzichtet, um den Arbeitsaufwand zu reduzieren. Der Gedanke hierbei war, dass potenzielle Fehlannotationen (z. B. durch Strukturen mit ähnlicher Textur) in einem nachgelagerten Verarbeitungsschritt über eine Höhenfilterung (z. B. mithilfe eines normalisierten Digitalen Oberflächenmodells, nDOM) korrigiert werden können.
Das manuelle Labeling wurde zunächst auf eine Szenenkategorie (Kategorie 2) beschränkt, um den Prozess zu testen und erste Trainingsdaten zu generieren. Für diese Kategorie wurden pro Stadt zehn Kacheln manuell gelabelt. Auf dieser Basis wurde ein erster Klassifikator trainiert, der anschließend dazu diente, in den noch nicht annotierten Szenen automatisch neue Bäume zu detektieren. Dieses Vorgehen wurde iterativ durchgeführt: Der Klassifikator wurde jeweils auf den verfügbaren gelabelten Kategorien trainiert und anschließend genutzt, um neue Kategorien halbautomatisch zu annotieren. Die daraus resultierenden Vorhersagen wurden im Anschluss manuell kontrolliert und korrigiert, bevor sie in das nächste Trainingsset integriert wurden. So entstand schrittweise ein erweiterter und abwechslungsreicherer Datensatz, während der manuelle Aufwand sukzessive reduziert

werden konnte.
Nach dem durchgeführten Sampling zeigt sich, dass die Städte Augsburg, Ingolstadt, Nürnberg und Regensburg eine vergleichbare Gesamtanzahl an Bäumen aufweisen (vgl. Tabelle 4.3). Augsburg verzeichnet mit 6 056 Bäumen die niedrigste, Regensburg mit 6 931 Bäumen die höchste Gesamtzahl innerhalb dieser vier Städte. München hebt sich mit insgesamt 11 621 Bäumen deutlich von den übrigen Städten ab. Diese erhöhte Zahl ist darauf zurückzuführen, dass fehlende Kacheln aus dem Gebiet Augsburgs durch Beispiele aus München ausgeglichen wurden. Insgesamt wurden über alle Städte und Kategorien hinweg rund 38 000 Bäume erfasst und gelabelt.

Tabelle 4.3: Gesamtanzahl und durchschnittliche Anzahl gelabelter Bäume pro Kachel für jede Stadt im Datensatz.

Datensatz	**Gesamtanzahl**	**Dursch. Anz./Kachel**
München	11621	247.26
Nürnberg	6249	156.22
Augsburg	6056	183.52
Regensburg	6931	173.28
Ingolstadt	6845	171.12
$\sum$	37702	

Bei der Betrachtung der Verteilung nach Kategorien zeigt sich ein weitgehend einheitliches Muster: In nahezu allen Städten weist Kategorie 3 die höchste Gesamtzahl an Bäumen auf, während Kategorie 4 die geringste Anzahl verzeichnet. Eine Ausnahme bildet Augsburg, wo aufgrund unzureichender Daten in Kategorie 3 die meisten Bäume in Kategorie 1 auftreten (vgl. Tabellen 4.4–4.8).

Tabelle 4.4: Gesamt-, minimale, maximale und durchschnittliche Anzahl gelabelter Bäume pro Kachel für den Datensatz München, aufgeschlüsselt nach Kategorien.

Kategorie	**Gesamtanz.**	**min. Anz./ K.**	**max. Anz./ K.**	**drs. Anz./ K.**
1	2048	40	353	204.8
2	2467	164	376	246.7
3	5301	178	399	311.82
4	1805	103	283	180.5

Die dargestellten Ergebnisse sind nicht als absolute Kennwerte für die gesamte Stadtfläche zu interpretieren, da die Datengrundlage auf einem Szenen-Sampling basiert. Dieses Verfahren hängt direkt vom gewählten Schwellwert ab, der vergleichsweise niedrig angesetzt wurde, um eine ausreichende Zahl an Beispielen zu gewährleisten. Die Ergebnisse sind daher nur bedingt repräsentativ für die tatsächliche Baumverteilung, erlauben jedoch eine relative Bewertung zwischen den untersuchten Städten.

Tabelle 4.5: Gesamt-, minimale, maximale und durchschnittliche Anzahl gelabelter Bäume pro Kachel für den Datensatz Nürnberg, aufgeschlüsselt nach Kategorien.

Kategorie	Gesamtanz.	min. Anz./ K.	max. Anz./ K.	drs. Anz./ K.
1	1110	38	165	111.0
2	1501	94	190	150.1
3	2433	153	369	243.3
4	1205	19	190	120.5

Tabelle 4.6: Gesamt-, minimale, maximale und durchschnittliche Anzahl gelabelter Bäume pro Kachel für den Datensatz Regensburg, aufgeschlüsselt nach Kategorien.

Kategorie	Gesamtanz.	min. Anz./ K.	max. Anz./ K.	drs. Anz./ K.
1	796	25	171	79.6
2	1870	132	304	187.0
3	2841	183	412	284.1
4	1424	19	253	142.4

Tabelle 4.7: Gesamt-, minimale, maximale und durchschnittliche Anzahl gelabelter Bäume pro Kachel für den Datensatz Augsburg, aufgeschlüsselt nach Kategorien.

Kategorie	Gesamtanz.	min. Anz./ K.	max. Anz./ K.	drs. Anz./ K.
1	1794	42	354	179.4
2	1602	120	217	160.2
3	1025	281	376	341.67
4	1635	41	353	163.5

Tabelle 4.8: Gesamt-, minimale, maximale und durchschnittliche Anzahl gelabelter Bäume pro Kachel für den Datensatz Ingolstadt, aufgeschlüsselt nach Kategorien.

Kategorie	Gesamtanz.	min. Anz./ K.	max. Anz./ K.	drs. Anz./ K.
1	1495	68	252	149.5
2	1674	133	224	167.4
3	2448	204	335	244.8
4	1228	27	268	122.8

4.1.4 Aufteilungen des Datensatzes für die Experimente

Für die Experimente wird der erzeugte Datensatz auf zwei verschiedene Arten aufgeteilt. Zunächst wurde der gesamte Datensatz zufällig in Trainings- und Validierungsdaten unterteilt. Da keine Veränderungen aufgrund der Validierungsdaten gemacht werden, wird kein zusätzlicher Test-Split erstellt. Diese Aufteilung dient primär der Evaluierung verschiedener Modellarchitekturen und -größen. Durch die standardisierte Aufteilung in Trainings- und Validierungsdaten kann die Leistung der Modelle objektiv anhand von Metriken wie mean Average Precision (mAP), Präzision, Recall und F1-Score verglichen werden. Diese Vorgehensweise erlaubt eine direkte Bewertung der Leistungsfähigkeit der unterschiedlichen Modelle unter gleichen Bedingungen und stellt sicher, dass die Ergebnisse nicht durch städtetypische Besonderheiten der Bilder verzerrt werden.

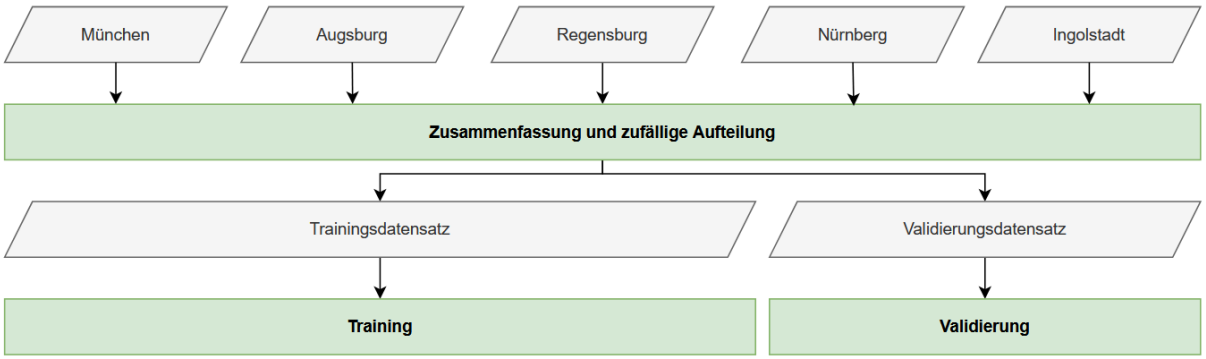

Abbildung 4.2: Aufteilung des Datensatzes für den Leistungsvergleich mehrerer Architekturen und Modellgrößen.

Für die Überprüfung der Praxistauglichkeit wurde eine stadtbasierte Aufteilung vorgenommen. Die Daten folgender Städte wurde für das Training und die Validierung verwendet: München, Regensburg und Augsburg. Für den Test auf nie zu vorgesehenen Daten wurden die Städte Nürnberg und Ingolstadt ausgewählt. Ziel dieser Vorgehensweise ist es, die Generalisierbarkeit des Modells auf neue urbane Umgebungen zu untersuchen und seine Anwendbarkeit in realen Szenarien zu evaluieren. Auf diese Weise können Einschränkungen bei der Übertragbarkeit auf andere Städte identifiziert und mögliche Optimierungen für den Einsatz in unterschiedlichen urbanen Kontexten abgeleitet werden. Durch die Aufteilung ergeben sich 120 Kacheln für das Training und die Validieren, sowie 80 Kacheln für den unabhängigen Test.

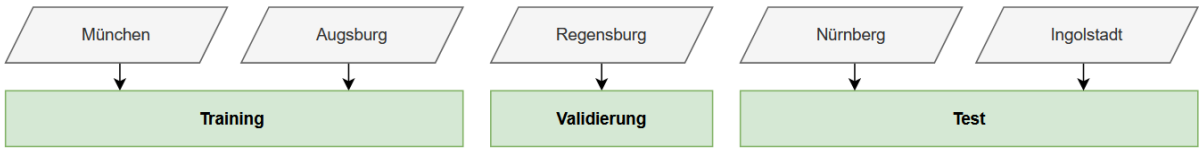

Abbildung 4.3: Aufteilung des Datensatzes für die praktische Anwendung eines Modells.

4.2 Vergleich aktueller Echtzeit-Objekterkennungsarchitekturen

Um die Leistungsfähigkeit aktueller Echtzeit-Objekterkennungsarchitekturen auf den geschaffenen Baumdatensatz unter vergleichbaren Bedingungen zu bewerten, wird in dieser Arbeit zunächst die Performance der Modelle auf diesen verglichen. Der Fokus liegt dabei ausschließlich auf der reinen Erkennungsleistung, unabhängig von der späteren Übertragbarkeit auf andere Datensätze oder Anwendungsszenarien. Untersucht werden drei aktuelle Modellfamilien: YOLOv11, YOLOv12 und RT-DETR. Für jede Architektur werden mehrere Modellgrößen und deren Genauigkeit, Trainings- und Inferenzzeit bestimmt. Alle Varianten werden unter identischen Trainingsbedingungen getestet und deren Validierungsergebnisse miteinander verglichen. Auf individuelles Tuning wird dabei verzichtet. Der Vergleich erfolgt sowohl innerhalb der einzelnen Architekturvarianten als auch zwischen den verschiedenen Modellfamilien. Zusätzlich wird für die jeweils leistungsstärkste Architekturvariante der Einfluss der Filterung von abgeschnittenen Labels während des Trainings untersucht.
Ziel dieser Untersuchung ist es, die Modelle hinsichtlich ihrer reinen Detektionsleistung zu beurteilen und dasjenige zu identifizieren, das unter diesen Bedingungen die höchste Genauigkeit bei gleichzeitig effizienter Trainings- und Inferenzzeit erzielt.

4.2.1 Architekturen und Modellgrößen

In Tabelle 4.9 sind die zu untersuchenden Modellarchitekturen mit ihren jeweiligen Größenvarianten, Inferenzgeschwindigkeiten und mAP50–95-Werten auf dem COCO-Datensatz dargestellt. Die Größenvarianten der YOLO-Modelle werden mit den Buchstaben n, s, m, l und x bezeichnet und decken Parameterbereiche von 2,6 bis 59,1 Millionen ab. Für RT-DETR werden von Ultralytics lediglich die Varianten l (42,3 Mio. Parameter) und x (76,5 Mio. Parameter) bereitgestellt. Die empfohlene Eingangsauflösung beträgt für alle Modelle 640 px, wobei andere Eingangsgrößen stets Vielfache von 32 px sein müssen (z. B. 320, 512, 640, 960 oder 1024 px). Vergleicht man die beiden YOLO-Generationen, zeigt YOLOv12 über alle Größen hinweg eine leicht verbesserte Erkennungsleistung bei nahezu identischer Inferenzgeschwindigkeit. Im Mittel liegt der Zugewinn bei der mAP50–95-Metrik bei etwa +1,1 Prozentpunkten gegenüber YOLOv11, während die Inferenzzeiten lediglich um durchschnittlich +0,2 ms höher ausfallen. Beispielsweise verbessert sich das x-Modell von 54,7% auf 55,2%, bei einer Laufzeitsteigerung von 11,3 ms auf 11,8 ms. Auch das kompakte n-Modell erzielt mit 40,6% gegenüber 39,5% eine Leistungssteigerung bei minimal höherer Latenz (1,64 ms gegenüber 1,5 ms). RT-DETR erreicht mit 53,0% (l) bzw. 54,8% (x) mAP50–95 nahezu die gleichen Präzisionswerte wie YOLOv12 l (53,7%) und x (55,2%), zeigt jedoch eine um etwa 30–40 % längere Inferenzzeit (8,8 ms vs. 6,8 ms bzw. 13,5 ms vs. 11,8 ms).

Tabelle 4.9: Vergleich der Architekturen und Modellgrößen auf dem COCO-Datensatz (nach Jocher (2025)

Architektur	Größe	Bildgröße (Pixel)	mAP50-95	Geschwindigkeit T4 TensorRT (ms)	Parameter (M)
YOLOv11	n	640	39.5	1.5	2.6
	s	640	47.0	2.5	9.4
	m	640	51.5	4.7	20.1
	l	640	53.4	6.2	25.3
	x	640	54.7	11.3	56.9
YOLOv12	n	640	40.6	1.64	2.6
	s	640	48.0	2.61	9.3
	m	640	52.5	4.86	20.2
	l	640	53.7	6.77	26.4
	x	640	55.2	11.79	59.1
RT-DETR	l	640	53.0	8.8	42.3
	x	640	54.8	13.5	76.5

4.2.2 Trainingskonfiguration und Evaluierungsrahmen

Um die Leistungsfähigkeit der verschiedenen Objekterkennungs-Architekturen und dazu gehörigen Parametergrößen objektiv miteinander vergleichen zu können, wurden sämtliche Trainings- und Evaluierungsläufe unter identischen Rahmenbedingungen durchgeführt. Dies betrifft sowohl die Wahl der Hyper- als auch der Augmentierungsparameter, sodass Unterschiede in den Ergebnissen ausschließlich auf architekturbedingte Einflüsse oder auf die im Methodik-Kapitel vorgestellte Labelfilterungsmethode zurückzuführen sind.
Für alle Experimente wurde die Anfangslernrate (lr0) auf 0.0001 festgelegt, während die Endlernrate (lrf) auf ein Zehntel gesetzt wurde. Als Optimierer kam „AdamW“ zum Einsatz, da dieser eine verbesserte Regularisierung gegenüber dem klassischen Adam-Optimierer ermöglicht und insbesondere bei kleineren Batch-Größen eine stabile Konvergenz gewährleistet. Die Batch-Größe wurde einheitlich auf 8 gesetzt. Die Trainingsdauer betrug 200 Epochen, wobei kein *patience*-Parameter zur automatischen Früherkennung von Konvergenz genutzt wurde, um eine vollständige Ausschöpfung des Lernprozesses sicherzustellen. Alle Modelle wurden mit vortrainierten COCO-Gewichten (von Ultralytics) initialisiert, um eine schnellere Konvergenz durch Transfer Learning zu ermöglichen. Die Eingabebilder wurden entsprechend den Ultralytics-Standardeinstellungen mit einer Kantenlänge von 640px übergeben. Dieses Format entspricht zugleich der zuvor beschriebenen Kachelgröße der vorbereiteten Datensätze, welche in 640-Pixel-Kacheln mit einer Überlappung von 25% unterteilt wurden. Damit wurde gewährleistet, dass sowohl die Trainingsdaten als auch die Evaluierungsszenen denselben räumlichen Zuschnitt aufweisen und keine Skalierung verwendet werden musste. Sämtliche übrigen Trainingsparameter entsprachen den Ultralytics-Standardeinstellungen (Tabelle 4.10). Die ausgewählten Hyperparameter

wurden anhand mehrerer vorangegangener Validierungsläufe bestimmt.

Tabelle 4.10: Übersicht der einheitlich verwendeten Trainingsparameter für alle durchgeführten Modell-Vergleiche.

Parameter	**Bezeichnung**	**Wert**
lr0	Anfangslernrate	0.0001
lrf	Endlernrate (Faktor)	0.01
optimizer	Optimierungsalgorithmus	AdamW
batch	Batch-Größe	8
epochs	Anzahl der Epochen	200
patience	Early-Stopping	0 (deaktiviert)
imgsz	Eingabebildgröße	640 px
pretrained	Vortrainierte Gewichte	COCO
device	Trainingsgerät	GPU (CUDA)
momentum	Momentum (AdamW Standard)	0.937
weight_decay	Gewichtsregularisierung	0.0005
warmup_epochs	Anzahl Warmup-Epochen	3.0
warmup_bias_lr	Warmup Bias-Lernrate	0.1
box	Box-Loss-Gewicht	7.5
cls	Klassifikations-Loss-Gewicht	0.5
dfl	Distribution Focal Loss Gewicht	1.5

Für die Datenaugmentierung wurden ausschließlich die standardmäßig in der Ultralytics-Implementierung aktivierten Augmentierungsparameter verwendet (vgl. Tabelle 4.11). Diese umfassen farbliche und geometrische Transformationen, welche die Variabilität der Trainingsdaten erhöhen und damit die Generalisierungsfähigkeit der Modelle fördern. Die Augmentierungsparameter wurden automatisch von der Trainings-Funktion verwendet, wenn man diese nicht manuell überschreibt.

Tabelle 4.11: Standard Augmentierungsparameter des Ultrayltics-Frameworks(Ultralytics, 2025).

Parameter	**Abkürzung**	**Standardwert**
Farbwert (hue)	hsv_h	0.015 (±zuf. Var. bis Wert)
Sättigung (saturation)	hsv_s	0.7 (±zuf. Var. bis Wert)
Helligkeit (value)	hsv_v	0.4 (±zuf. Var. bis Wert)
Translation	translate	0.1 (±zuf. Var. bis Wert)
Skalierung	scale	0.5 (±zuf. Var. bis Wert)
Horizontales Spiegeln	fliplr	0.5 (Wahrscheinlichkeit)
Mosaic (4 Bilder kombinieren)	mosaic	1.0 (Wahrscheinlichkeit)

Sämtliche Experimente wurden auf einem identischen Hardware-Setup ausgeführt, bestehend aus einer Intel Xeon W-2223 CPU (4 Cores / 8 Threads, 3,60 GHz), 64 GB

DDR4-RAM und einer NVIDIA RTX A4000 GPU mit 16 GB GDDR6-VRAM.

4.2.3 Training und Validierung

Um die tatsächliche Leistungsfähigkeit der verschiedenen Architekturen und Modellgrößen im Anwendungsfall der Baumdetektion zu beurteilen, wurden alle Varianten auf dem zuvor erstellten Gesamtdatensatz trainiert und validiert. Der Datensatz wurde dafür in 70% Trainings- und 30% Validierungsdaten aufgeteilt. Auf einen separaten Test-Split wurde verzichtet, da keine Anpassungen oder Optimierungen anhand des Validierungssplits vorgenommen wurden. Vor der Aufteilung wurden die Daten zufällig durchmischt. Eine gezielte Kontrolle der Klassenverteilung in den Splits erfolgte nicht, da alle Modelle auf identischen Datensätzen trainiert und evaluiert wurden, wodurch eine faire Vergleichbarkeit gewährleistet ist. Das Training erfolgte für jede Modellvariante mit den in Kapitel 4.2.2 beschriebenen Hyperparametern. Zur Erfassung und Dokumentation der Leistungsmetriken wurden die im *Ultralytics*-Framework integrierten Funktionen verwendet. Die Validierung erfolgte auf bereits zugeschnittenen und konvertierten Orthophotos sowie den dazugehörigen Label-Dateien. Um eine möglichst unverzerrte Messung der tatsächlichen Trainingszeit zu gewährleisten, wurde diese unmittelbar vor dem Aufruf der *train*-Funktion von Ultralytics gestartet und nach deren Abschluss gestoppt. Die Inferenzzeiten der Validierung wurden ebenfalls mit einer integrierten Funktion des Frameworks bestimmt. Diese Messung umfasst den vollständigen Vorhersageprozess, einschließlich Preprocessing, Inferenz und Postprocessing. Für jedes Modell wurden zehn vollständige Vorhersagedurchläufe durchgeführt. Die ermittelten Zeiten wurden anschließend gemittelt, um zufällige Schwankungen in der Systemauslastung des Betriebssystems (z.B. durch Hintergrundprozesse unter Windows) zu kompensieren. Die in Tabelle 4.12 dargestellten Ergebnisse zeigen die jeweils maximal erreichten Validierungsleistungen (mAP50-95 und mAP50) sowie die entsprechenden F1-Werte, Trainingszeiten und mittleren Inferenzzeiten aller getesteten Modellarchitekturen und -größen.

Innerhalb jeder Architektur ist ein Anstieg der Validierungsleistung mit zunehmender Modellgröße zu erkennen. Bei YOLOv11 steigt die mAP50-95 von 0.475 beim kleinsten Modell (n) auf 0.760 beim größten Modell (x), während die mAP50 von 0.812 auf 0.938 zunimmt. Während die Leistungssteigerungen in der mAP50-95 mit aufsteigender Größe deutlich ausfallen, sind bei den mAP50-Werten ab der Größe m nur noch geringe Verbesserungen von 0.005 (m zu l) und 0.01 (l zu x) zu erkennen. Der F1-Wert folgt demselben Trend und steigt von 0.764 (n) auf 0.923 (x). Die Trainingszeiten nehmen mit der Modellgröße zu und liegen zwischen 1.71 h (n) und 4.28 h (x). Ebenso steigt die mittlere Inferenzzeit mit zunehmender Modellgröße von 9.6 ms (n) auf 18.5 ms (x). Damit zeigt sich eine klare Korrelation zwischen Modellkomplexität, Leistungsfähigkeit und Rechenaufwand.
Ein vergleichbares Muster zeigt sich bei YOLOv12, wo die mAP50-95 von 0.508 (n) auf 0.757 (x) und die mAP50 von 0.841 auf 0.937 ansteigen. Auch der F1-Wert zeigt eine deutliche Steigerung von 0.792 (n) auf 0.918 (x). Die Trainingszeiten liegen zwischen 2.20 h (n) und 6.21 h (x) und sind somit insgesamt höher als bei YOLOv11. Die mittlere Inferenzzeit steigt von 12.9 ms beim kleinsten Modell auf 22.6 ms beim größten Modell.

Tabelle 4.12: Vergleich der Validierungsleistung (mAP-Validierung, Präzision, Recall, F1-Score) sowie der Trainings- und mittleren Inferenzzeit der Modelle auf dem Validierungsdatensatz.

Modell	**mAP50-95**	**mAP50**	**F1**	**Training [h]**	**∅ Inferenz [ms]**
YOLO11x	0.760	0.938	0.923	4.28	18.5
YOLO11l	0.697	0.926	0.899	2.94	14.5
YOLO11m	0.675	0.921	0.888	2.35	10.4
YOLO11s	0.583	0.884	0.844	1.74	9.6
YOLO11n	0.475	0.812	0.764	1.71	9.6
YOLO12x	0.757	0.937	0.918	6.21	22.6
YOLO12l	0.722	0.931	0.910	4.33	19.1
YOLO12m	0.709	0.928	0.905	3.16	12.5
YOLO12s	0.627	0.902	0.869	2.37	12.9
YOLO12n	0.508	0.841	0.792	2.20	12.9
RT-DETRx	0.644	0.909	0.884	7.18	27.8
RT-DETRl	0.601	0.894	0.864	5.65	21.7

Trotz dieser leichten Zunahme der Inferenzzeit bleibt die Effizienz der YOLOv12-Modelle im Verhältnis zur Leistungssteigerung erhalten. Besonders hervorzuheben ist YOLO12m, das eine geringere durchschnittliche Inferenzzeit aufweist als das kleinere s und n-Modell. Vergleicht man die Inferenzzeiten mit den YOLOv11-Modellen, so übertrifft YOLO12m in allen Leistungsmetriken die Modelle l und m der Vorgängerversion. Die Trainingszeit liegt über den Werten der YOLO11m- und YOLO11l-Varianten, die Inferenzzeit postioniert sich zwischen YOLOv11m und -l.
Die RT-DETR-Modelle erreichen mAP50-95-Werte von 0.601 (l) und 0.644 (x) sowie mAP50-Werte von 0.894 und 0.909. Auch hier kann mit steigender Modellgröße eine Leistungssteigerung festgestellt werden. Der F1-Wert nimmt leicht zu, von 0.864 (l) auf 0.884 (x), bleibt jedoch insgesamt unter den entsprechenden Werten (bezogen auf die angegebenen Größen-Buchstaben l und x) der YOLOv11- und YOLOv12-Modelle. Die Trainingszeiten liegen bei 5.65 h (l) und 7.18 h (x) und sind damit die höchsten im Vergleich. Ebenso zeigen die Modelle die längsten mittleren Inferenzzeiten mit 21.7 ms (l) und 27.8 ms (x). Abbildung 4.4 zeigt zusammenfassend die erreichten Leistungen in Abhängigkeit von der jeweiligen Parameteranzahl des verwendeten Modells.

Zur besseren Einordnung der Effizienz der getesteten Modelle wurde die erreichte Validierungsleistung in Relation zur jeweiligen Trainingszeit dargestellt (Abbildung 4.5). Auf der x-Achse ist die Trainingszeit in Stunden angetragen, während die y-Achse die erzielte Validierungsleistung (mAP50–95) zeigt. Diese Darstellung ermöglicht eine direkte Bewertung des Verhältnisses zwischen Trainingsaufwand und erzielter Genauigkeit und kann als Grundlage für die Auswahl eines geeigneten Modells für den weiteren Einsatz dienen.

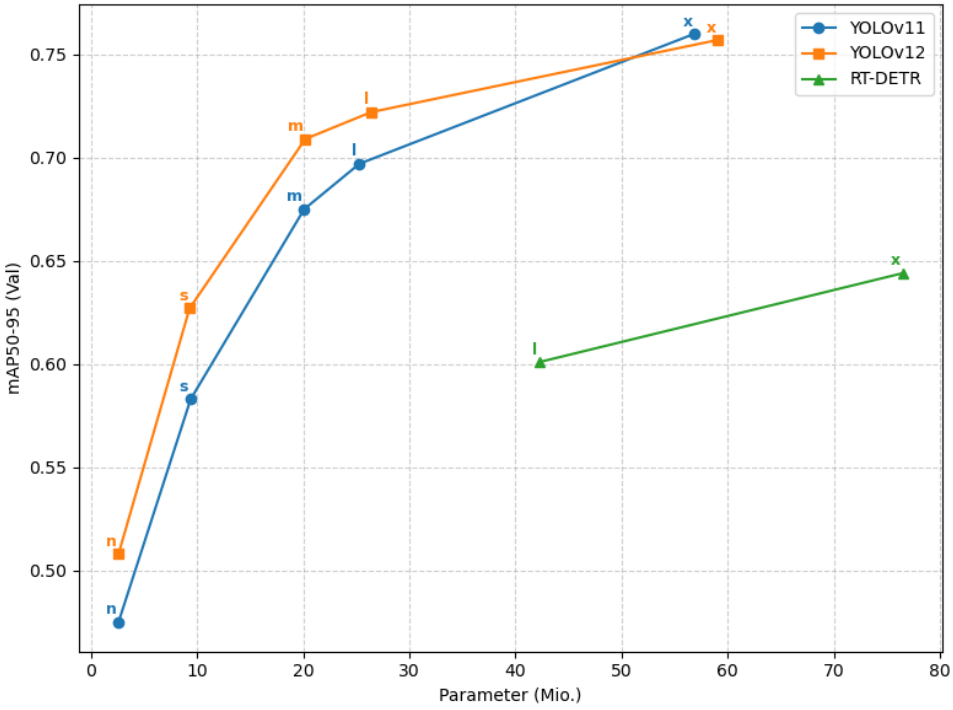

Abbildung 4.4: Darstellung der Validierungsleistung (mAP50-95) in Relation zur Parameterzahl für alle Architekturen und deren Modellgrößen.

Die Auswertung zeigt, dass die Modelle der YOLOv11-Reihe im Verhältnis zur Trainingszeit insgesamt eine höhere Effizienz aufweisen als die Modelle der YOLOv12-Serie. Während die YOLOv12-Modelle im Mittel leicht höhere mAP-Werte erzielen, fällt der Leistungszuwachs im Verhältnis zur längeren Trainingszeit gering aus. Besonders deutlich wird dies bei den größten Modellen: YOLOv11x erreicht mit einer mAP50–95 von 0.760 nahezu die gleiche Genauigkeit wie YOLOv12x (0.757), benötigt dafür jedoch nur 4.28 h statt 6.21 h Trainingszeit.

Abbildung 4.6 veranschaulicht den Zusammenhang zwischen der erzielten Validierungsleistung (mAP50-95) und der mittleren Inferenzzeit pro Bild. Auf der x-Achse ist die durchschnittliche Inferenzzeit in Millisekunden dargestellt, während die y-Achse die erzielte mAP50-95 angibt. Jedes Modell ist entsprechend seiner Architektur farblich markiert, und die jeweilige Modellgröße (n-x) ist als Beschriftung der Punkte angegeben.
Die Ergebnisse zeigen, dass YOLOv11x mit einer mAP50-95 von 0.760 und einer mittleren Inferenzzeit von 18.5ms die höchste Genauigkeit bei zugleich sehr guter Laufzeit erreicht. YOLOv12x erzielt mit 0.757 mAP50-95 nahezu dieselbe Genauigkeit, benötigt mit 22.6ms jedoch rund 22% mehr Zeit pro Bild. Damit bleibt YOLOv11x insgesamt effizienter. Im mittleren Modellbereich zeigt YOLOv12m eine besonders ausgewogene Leistung: Mit 0.709 mAP50-95 bei einer Inferenzzeit von 12.5ms bietet es das beste Verhältnis von Genauigkeit zu Geschwindigkeit innerhalb der YOLOv12-Reihe. Es übertrifft dabei die Modelle YOLOv11m (0.675 mAP50-95, 10.4ms) und YOLOv11l (0.697 mAP50-95, 14.5ms) sowohl in der Genauigkeit als auch in der Effizienz, da der Leistungszuwachs nur mit

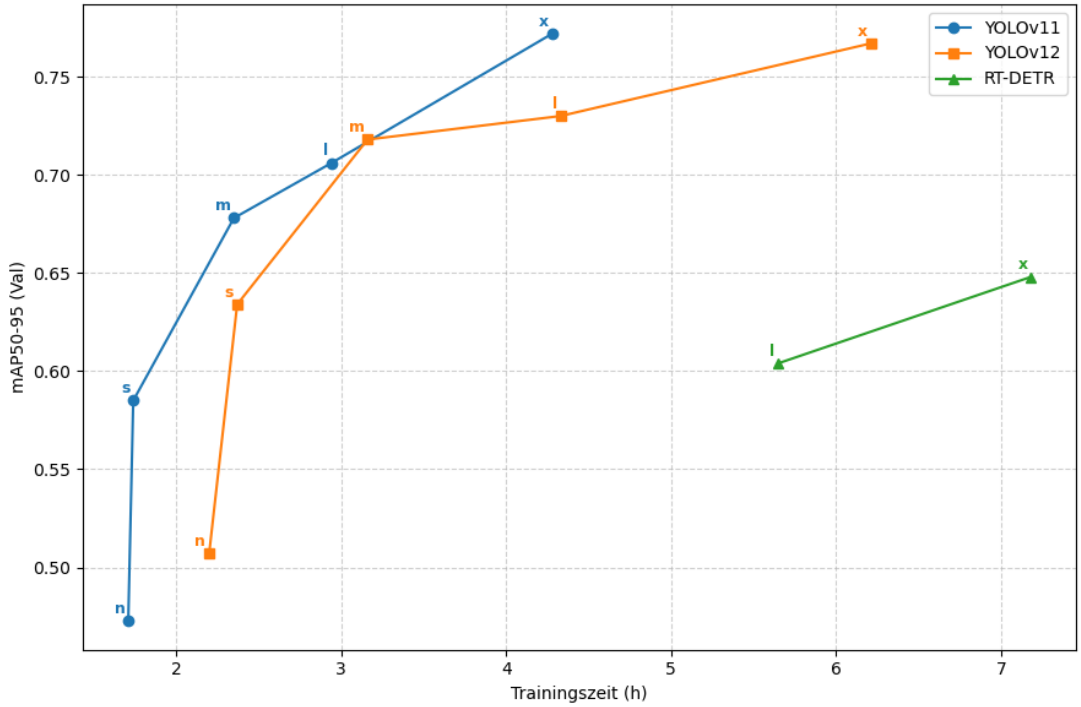

Abbildung 4.5: Darstellung der Validierungsleistung (mAP50-95) in Relation zur Trainingszeit für alle Architekturen und deren Modellgrößen.

geringfügig höherer Inferenzzeit erreicht wird. Die kleineren Modelle (s und n) beider Serien zeigen, wie erwartet, deutlich geringere Inferenzzeiten (zwischen 9.6ms und 12.9ms), jedoch mit spürbarem Verlust an Erkennungsleistung. So liegt YOLOv11n bei 0.475 mAP50-95, während YOLOv12n mit 0.508 mAP50-95 etwas besser abschneidet, jedoch langsamer ist. Im Vergleich dazu erreichen die RT-DETR-Modelle mit 0.601 mAP50-95 (RT-DETRl, 21.7ms) bzw. 0.644 mAP50-95 (RT-DETRx, 27.8ms) deutlich niedrigere Genauigkeiten bei zugleich höheren Inferenzzeiten.
Insgesamt lässt sich festhalten, dass die YOLOv12-Serie tendenziell eine etwas höhere Genauigkeit erzielt, aber längere Inferenzzeiten aufweist. Die YOLOv11-Modelle bleiben daher in Bezug auf die Effizienz (Leistung pro Inferenzzeit) leicht im Vorteil. Insgesamt bietet YOLOv12m den besten Kompromiss zwischen Genauigkeit und Geschwindigkeit bietet.

In den Abbildungen 4.7 bis 4.9 sind die Trainingsverläufe der Modelle YOLOv11, YOLOv12 und RT-DETR über 200 Epochen dargestellt. Gezeigt werden jeweils die mAP50–95 (oben) und die mAP50 (unten) der unterschiedlichen Modellgrößen. Bei YOLOv11 ist eine Konvergenz der mAP50 zwischen etwa 75 und 100 Epochen zu erkennen, während YOLOv12 erst zwischen 100 und 125 Epochen eine vergleichbare Stabilisierung erreicht. Das Modell RT-DETR zeigt eine Konvergenz der mAP50 ab etwa 150 Epochen. Die mAP50–95 steigt bei allen drei Modellen über den gesamten Trainingsverlauf hinweg kontinuierlich an und zeigt keine vollständige Sättigung innerhalb der 200 Epochen. In den letzten zehn Epochen ist bei allen Modellen ein leichter Abfall der mAP50- und mAP50–95-

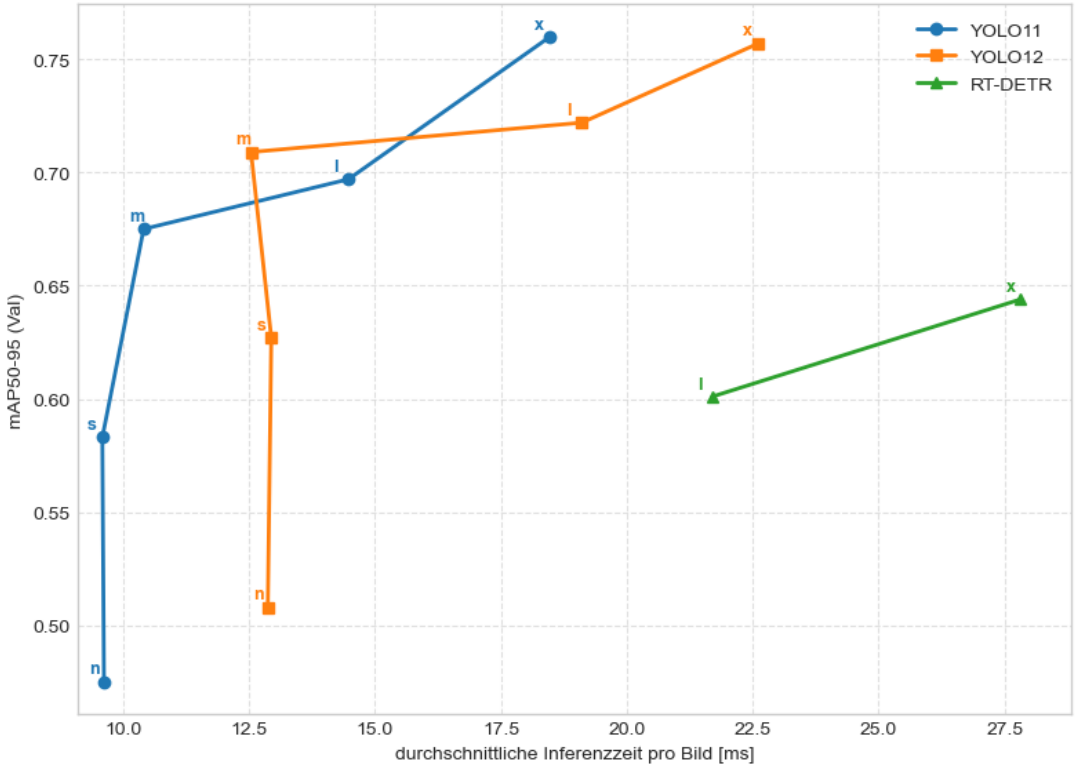

Abbildung 4.6: Darstellung der Validierungsleistung (mAP50-95) in Relation zur durchschnittlichen Inferenzzeit für alle Architekturen und deren Modellgrößen.

Werte zu beobachten. Dieser Effekt ist auf die Deaktivierung des Mosaic-Loaders im späten Trainingsstadium zurückzuführen, diese Einstellung wurde erst nach Abschluss der Trainingsläufe aufgedeckt. Bereits in den frühen Trainingsphasen (etwa 0–25 Epochen) treten Unterschiede in der Performance der Modellvarianten auf. Die Differenzen vergrößern sich mit zunehmender Epochenzahl, insbesondere bei der mAP50–95. Bei der mAP50 hingegen bleiben sie nach Erreichen der Konvergenz konstant. In Abbildung 4.10 werden die Trainingsverläufe der jeweils besten Modelle gegenübergestellt.

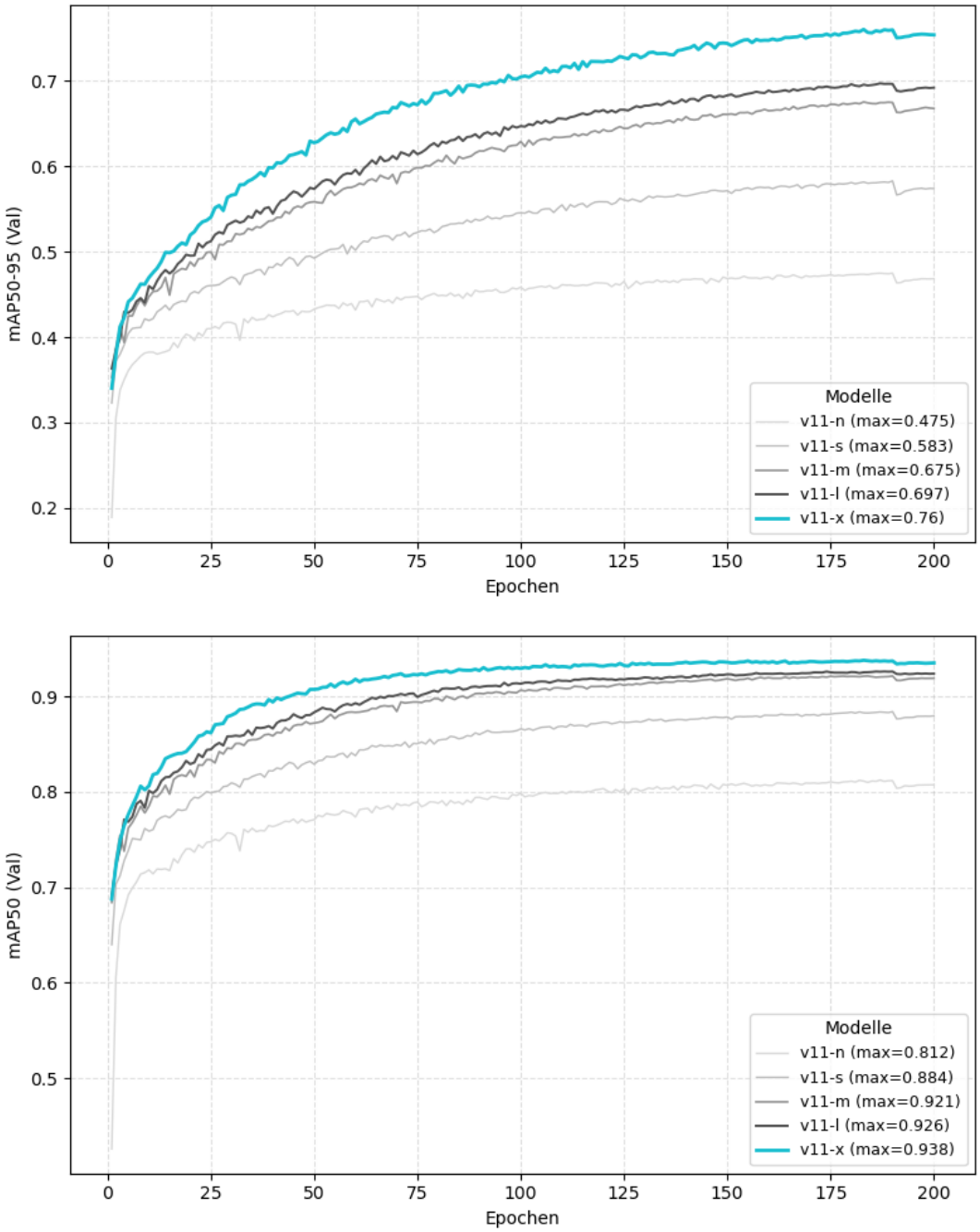

Abbildung 4.7: YOLOv11 – Vergleich der Validierungsergebnisse anhand der mAP50–95 (oben) und mAP50 (unten) über den Trainingsverlauf von 200 Epochen. Die Modellgröße mit der höchsten Leistung ist jeweils in blau dargestellt.

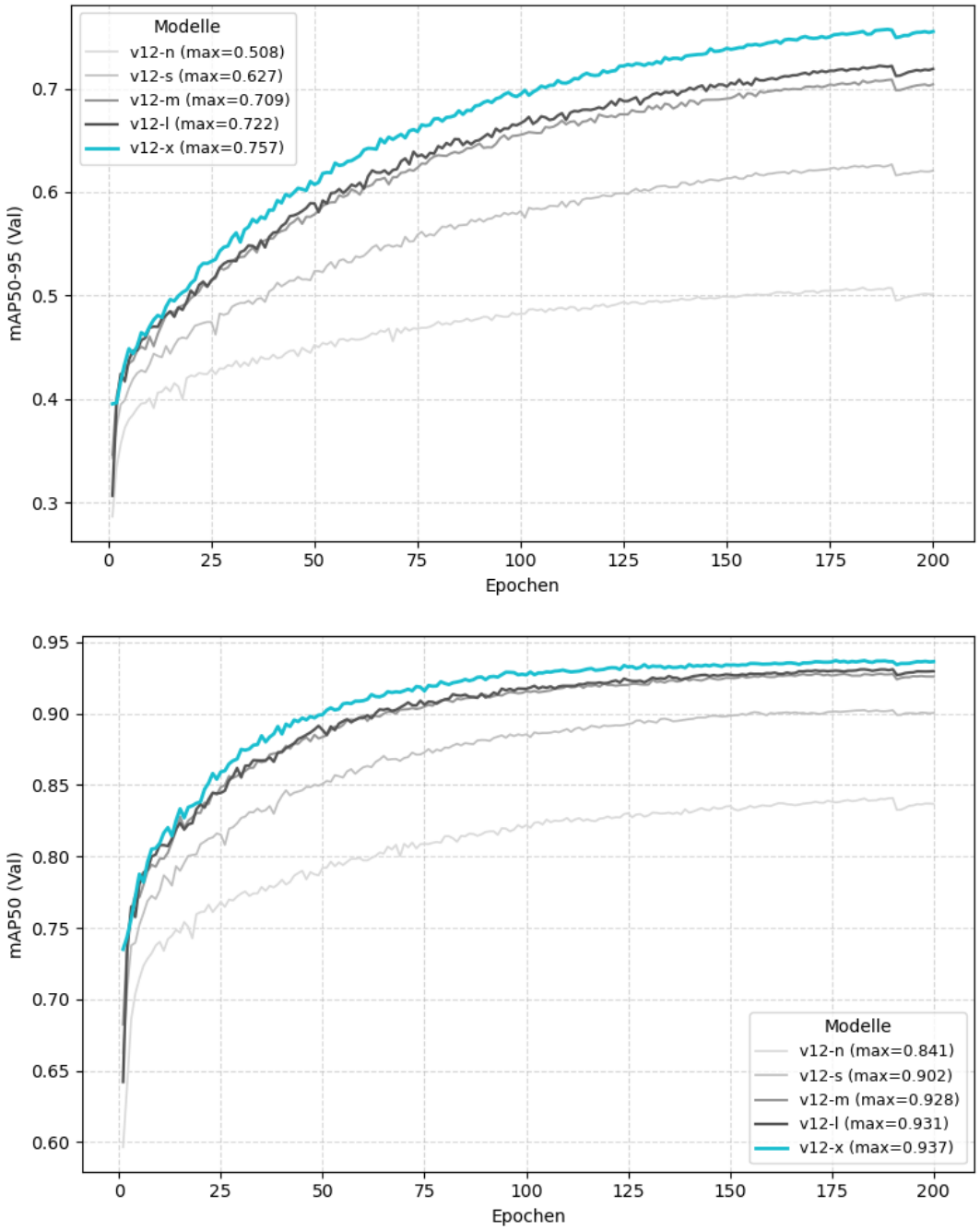

Abbildung 4.8: YOLOv12 – Vergleich der Validierungsergebnisse anhand der mAP50–95 (oben) und mAP50 (unten) über den Trainingsverlauf von 200 Epochen. Die Modellgröße mit der höchsten Leistung ist jeweils in blau dargestellt.

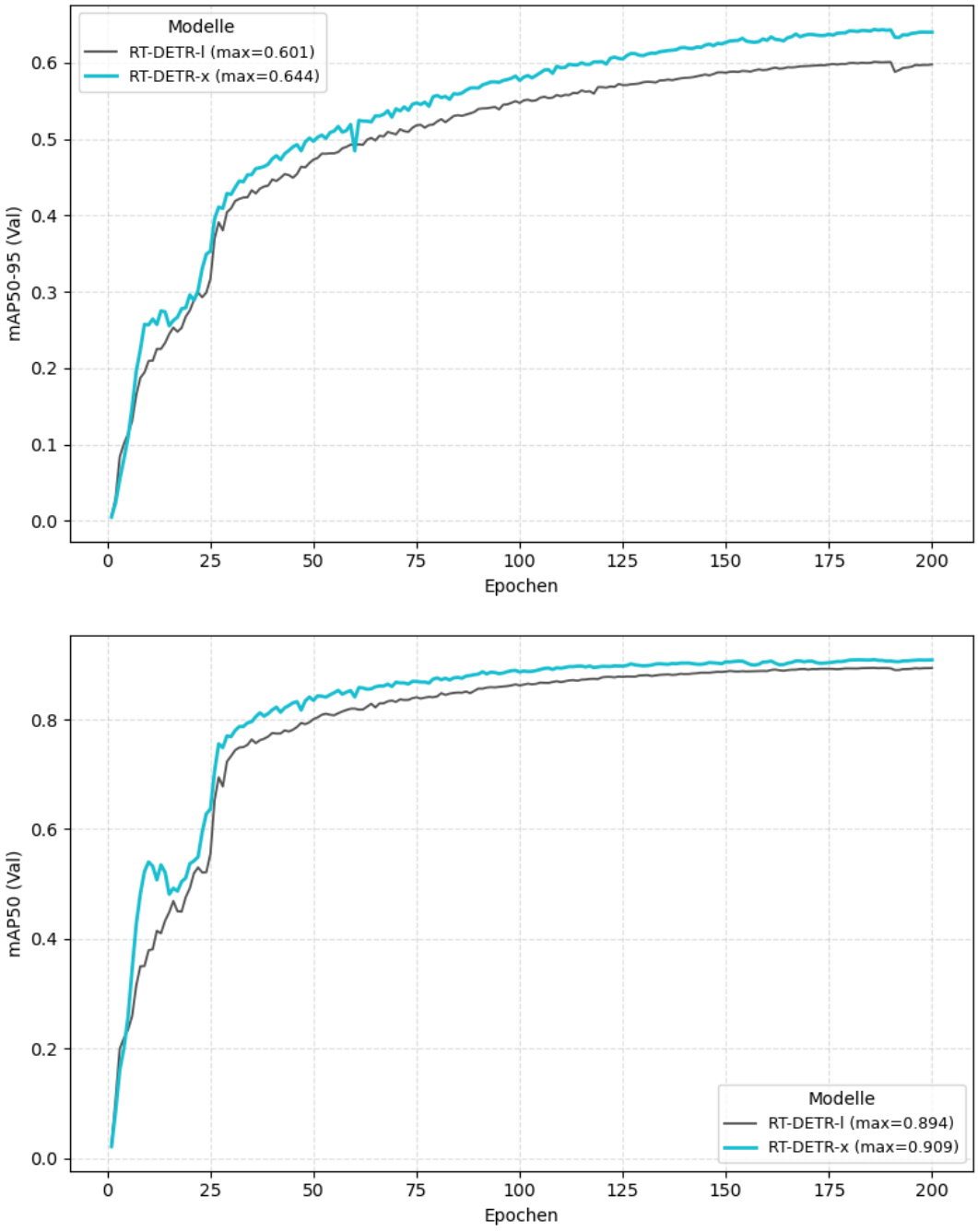

Abbildung 4.9: RT-DETR – Vergleich der Validierungsergebnisse aller Modellgrößen anhand der mAP50–95 (oben) und mAP50 (unten) über den Trainingsverlauf von 200 Epochen. Die Modellgröße mit der höchsten Leistung ist jeweils in blau dargestellt.

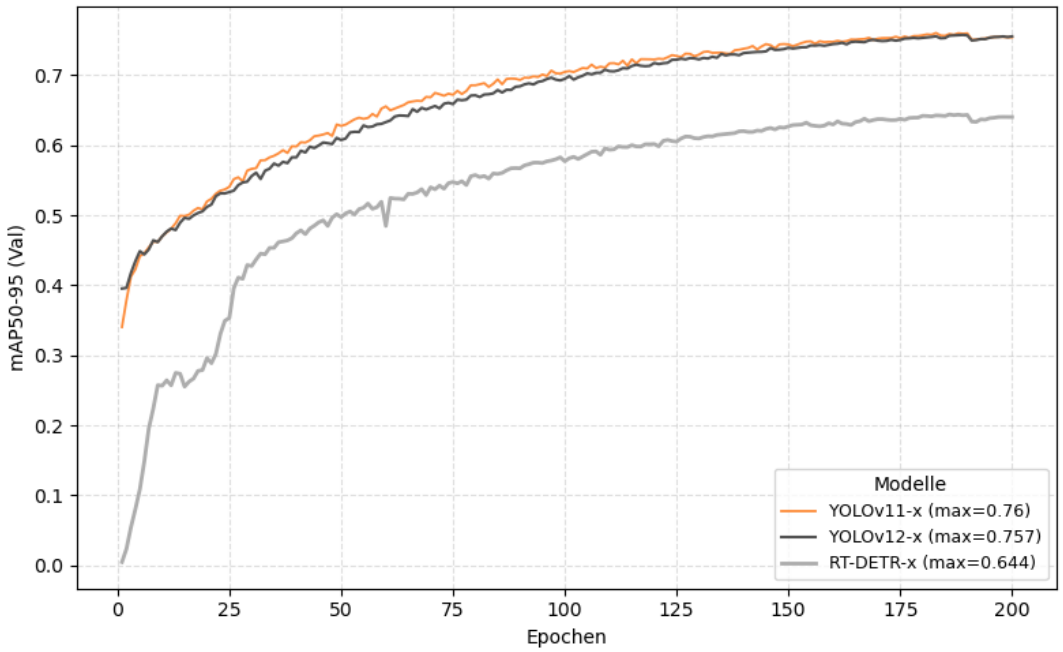

Abbildung 4.10: Vergleich der Validierungsergebnisse des besten Modells je getesteter Architektur anhand der mAP50-95-Leistung über den Trainingsverlauf von 200 Epochen. Die Modellgröße mit der höchsten Leistung ist in orange dargestellt.

4.2.4 Filterung von abgeschnittenen Labels

Da beim Training Orthophotos verwendet werden, die größer sind als die Eingangsgröße der verschiedenen Modelle, müssen diese in Ausschnitte unterteilt werden. Damit alle Trainingsbeispiele mindesten einmal im Training vorhanden sind, wurde eine Überlappung von 25% gewählt. Durch das Ausschneiden ergeben sich Labelfragmente, welche unvollständige Beispiele repräsentieren. Um den Einfluss abgeschnittener Labels auf die Lokalisierungsleistung zu untersuchen, wurden die besten Objekterkennungsmodelle aus den vorherigen Tests herangezogen. Die verwendeten Modelle sind YOLOv11x, YOLOv12x und RT-DETRx. Für die Filterung wurden Sichtbarkeitsschwellen von 0 % bis 50 % in 10 %-Schritten definiert. Der jeweilige Schwellwert gibt an, welcher Restflächenanteil eines Labels im Bildausschnitt sichtbar sein muss, damit es beim Training berücksichtigt wird.

Training und Test

Wie beim vorherigen Vergleich der Architekturen entsprechen die Trainingsparameter den in Kapitel 4.2.2 beschriebenen Einstellungen. Es wurden auch wieder die Daten des gesamten Datensatzes verwendet. Um den Effekt der Label-Filterung zu quantifizieren, wurde für jedes Modell zusätzlich eine Referenz ohne Filterung trainiert.
Die Validierungsergebnisse (mAP50–95 und mAP50) sind in Tabelle 4.13 zusammengefasst. Das YOLO11x Modell zeigt mit zunehmender Filterung bis zu einem Sichtbarkeitsschwellwert von 30% eine stetige Leistungssteigerung. Der mAP50–95 verbessert sich von 0.760 auf 0.790, und auch der mAP50 steigt leicht von 0.938 auf 0.960. Ab einem Schwellwert von über 30% stagniert die Leistung jedoch. Ein ähnlicher Trend lässt sich bei YOLOv12x beobachten. Bis zu einem Schwellwert von 40% steigt der mAP50–95 von 0.757 auf 0.787. Der mAP50 erreicht mit 0.958 sein Maximum bei 30%. Danach sinken die Werte minimal. Auch bei RT-DETRx ist eine Leistungssteigerung zu verzeichnen. Der mAP50–95 steigt von 0.644 auf 0.679 bei 40%, während der mAP50 bei 30% von 0.909 auf 0.942 zunimmt. Ab 40% verschlechtert sich die Leistung auf das Niveau von 10%.

Betrachtet man die Trainingsverläufe in Abbildung 4.11 bis 4.13, so werden die Leistungsunterschiede am Ende der Trainingsphasen aller drei Architekturen deutlich sichtbar. Bei YOLOv11 zeigen sich in den ersten 75 Epochen jedoch starke Schwankungen, wodurch zunächst keine klare Unterscheidung zwischen den getesteten Schwellwerten möglich ist. Ein ähnlicher Effekt ist bei YOLOv12 zu beobachten, wobei sich die Kurve hier etwas früher stabilisiert. Insgesamt wirkt der Verlauf bei YOLOv12 glatter und gleichmäßiger als bei YOLOv11. Bei RT-DETR ist bereits nach etwa 50 Epochen deutlich erkennbar, wie die verschiedenen Schwellwerte einzuordnen sind. Im Vergleich zu YOLOv11 und YOLOv12 zeigt RT-DETR einen insgesamt deutlich gleichmäßigeren und stabileren Kurvenverlauf. Abbildung 4.14 zeigt eine zusammenfassende Gegenüberstellung der erzielten mAP50–95-Werte über alle Sichtbarkeitsschwellen.

Tabelle 4.13: Vergleich der Modellleistungen (mAP50–95, mAP50, F1) in Abhängigkeit des angewendeten Sichtbarkeitsschwellwerts zur Filterung abgeschnittener Labels.

Modell	**Filterungsschwellwert** [%]	**mAP50-95**	**mAP50**	**F1**
YOLO11x	0	0.760	0.938	0,923
	10	0.757	0.946	0,922
	20	0.779	0.958	0,915
	30	0.790	0.960	0,919
	40	0.786	0.955	0,924
	50	0.784	0.955	0,925
YOLO12x	0	0.757	0.937	0,918
	10	0.753	0.945	0,921
	20	0.777	0.956	0,917
	30	0.782	0.958	0,919
	40	0.787	0.956	0,925
	50	0.781	0.956	0,925
RT-DETRx	0	0.644	0.909	0,884
	10	0.661	0.930	0,904
	20	0.674	0.937	0,892
	30	0.676	0.942	0,896
	40	0.679	0.937	0,902
	50	0.666	0.935	0,898

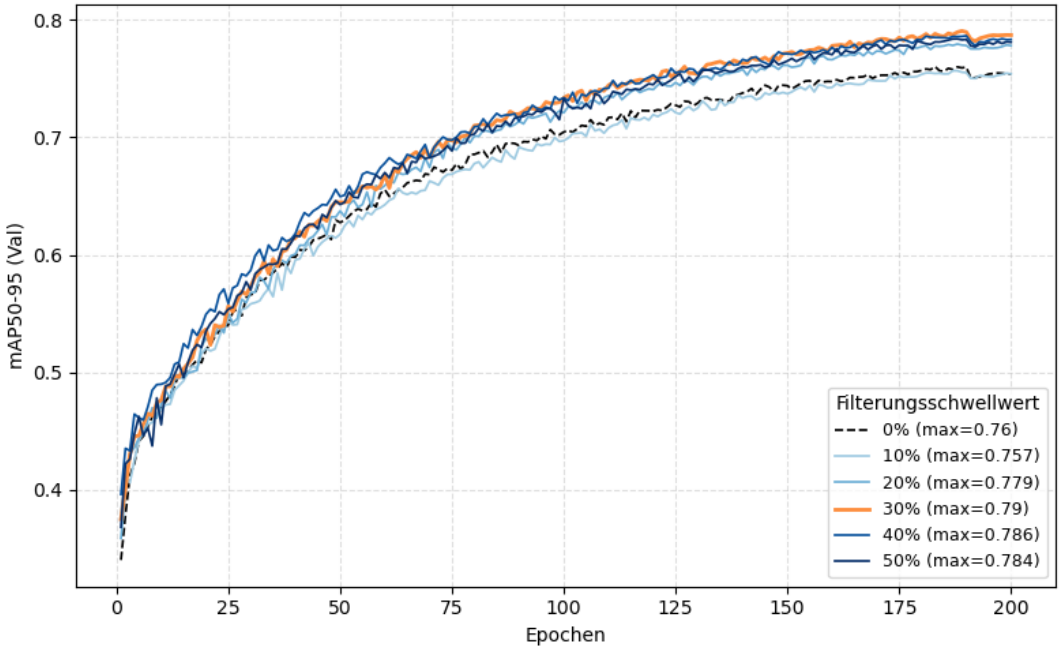

Abbildung 4.11: YOLOv11x – Vergleich der Validierungsergebnisse aller Filterungsschwellwerte anhand der mAP50-95-Leistung über den Trainingsverlauf von 200 Epochen. Referenz ohne Filterung (0%) wird gestrichelt dargestellt.

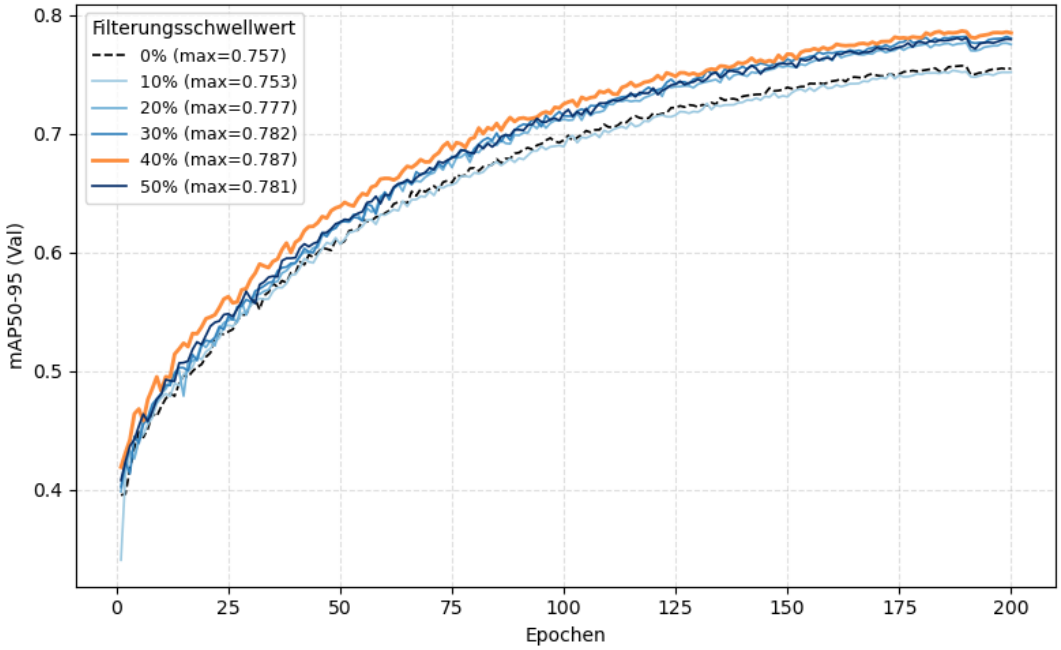

Abbildung 4.12: YOLOv12x – Vergleich der Validierungsergebnisse aller Filterungsschwellwerte anhand der mAP50-Leistung über den Trainingsverlauf von 200 Epochen. Referenz ohne Filterung (0%) wird gestrichelt dargestellt.

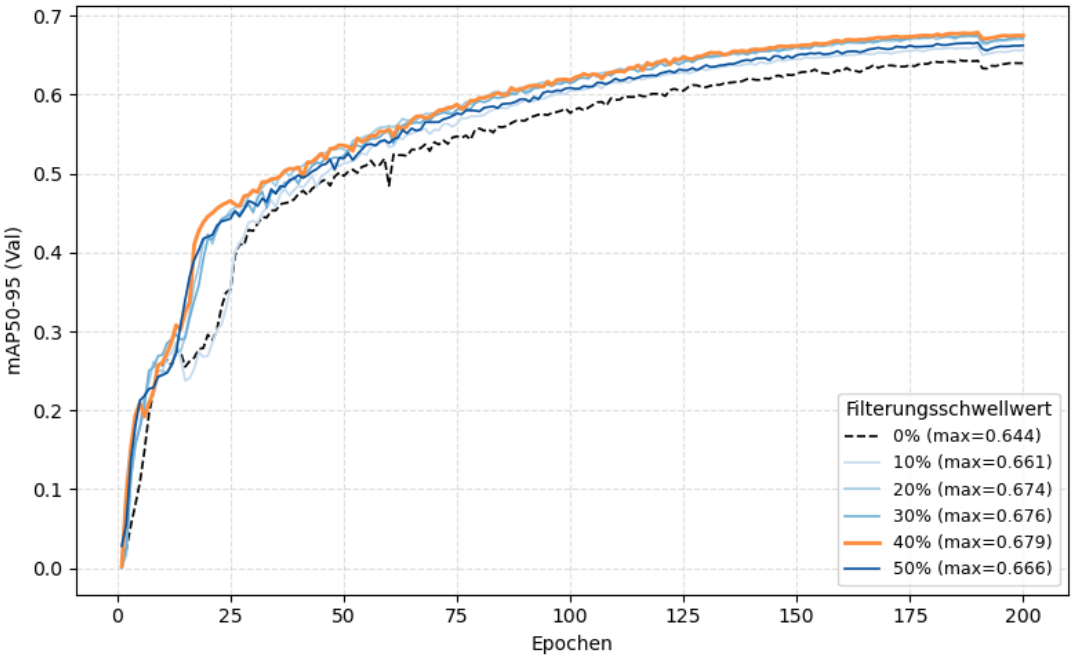

Abbildung 4.13: RT-DETRx – Vergleich der Validierungsergebnisse aller Filterungsschwellwerte anhand der mAP50-95-Leistung über den Trainingsverlauf von 200 Epochen. Referenz ohne Filterung (0%) wird gestrichelt dargestellt.

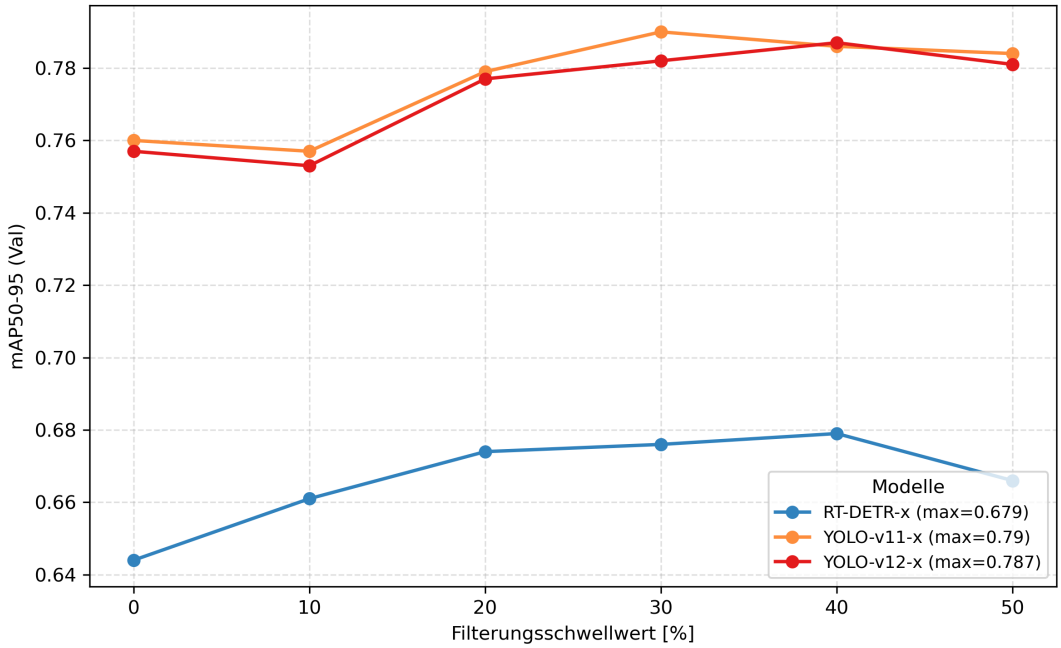

Abbildung 4.14: Visualisierung der Validierungsergebnisse über die mAP50-95 in Bezug auf die Schwellwerte von 0% bis 50% bei der Labelfilterung für die besten Modell aus dem Architekturvergleich.

4.3 Praktische Anwendung der automatisierten Baumdetektion

Nach der reinen Leistungsbestimmung der verschiedenen Modellarchitekturen und -größen erfolgte im nächsten Schritt die Auswahl eines geeigneten Modells für die praktische Anwendung. Während in der vorhergehenden Untersuchung ausschließlich die Detektionsleistung unter identischen Bedingungen betrachtet worden war, stand in dieser Phase zusätzlich die Übertragbarkeit der Modelle auf neue Datensätze sowie der Einfluss der verfügbaren Datenmenge im Fokus. Zu diesem Zweck wurden Training und Validierung auf unterschiedlichen Städten durchgeführt, um die Generalisierungsfähigkeit der Modelle zu bewerten.
Dieses Kapitel beschreibt die Schritte zur Auswahl und Optimierung eines geeigneten Modells für die automatisierte Baumdetektion auf unbekannten Gebieten. Die Arbeitsschritte umfassen die Auswahl des am besten geeigneten Modells aus einer Reihe von Architekturen und Größen, die Optimierung der Trainingsparameter für eine optimale Generalisierung und die abschließende Anwendung auf unabhängigen Testdatensätzen.
Das Setup für die praktische Anwendung basiert auf einem leistungsstarken Desktop-System mit einem Intel Core i7-12700K Prozessor, 32 GB DDR4-Arbeitsspeicher und einer NVIDIA GeForce RTX 5070 Grafikkarte.

4.3.1 Auswahl eines geeigneten Modells

Die Modelle wurden mit den Daten aus München und Augsburg trainiert, während Regensburg als unabhängiger Validierungsdatensatz diente. Für das Training wurden erneut die in Kapitel 4.2.2 beschriebenen Parameter verwendet, jedoch wurde die Epochenanzahl auf 30 reduziert. Diese Anpassung basierte auf den Ergebnissen vorangegangener Tests, die gezeigt hatten, dass bei der gegebenen Datensatzgröße und der beabsichtigten Generalisierung bereits nach wenigen Epochen eine stabile Konvergenz erreicht wurde. Eine höhere Trainingsdauer führte dabei zu keiner eindeutigen Leistungssteigerung, sondern lediglich zu einem erhöhten Rechenaufwand. Bewertet wurde die maximal erzielte Leistung auf dem Validierungsdatensatz, da diese den besten Kompromiss zwischen Erkennungsgenauigkeit und Übertragbarkeit widerspiegelte, ohne die Testdaten zu verwenden. Die Metriken und Trainingszeiten wurden analog zu Kapitel 4.2.3 bestimmt. Die abgeschnittenen Labels in den Trainingsdaten wurden mit dem zuvor detektierten Schwellwert von 30% gefiltert.

Die in Tabelle 4.14 dargestellten Ergebnisse zeigen, dass das YOLOv12m-Modell mit einem mAP50-95 von 0.399 und einem F1-Wert (bei IoU=50) von 0.712 die höchste Gesamtleistung erzielt. Die größere Modellvariante YOLOv12l erreicht mit einem mAP50-95 von 0.397 und einem F1 von 0.715 ein nahezu gleichwertiges Ergebnis, benötigt jedoch mit 15.5 Minuten die längste Trainingszeit.
Das YOLOv11l-Modell liegt mit einem mAP50-95 von 0.396 und einem F1 von 0.709 nur geringfügig darunter. Ähnlich schneiden YOLOv11m und YOLOv12s ab, die beide einen mAP50-95 von 0.391 erzielen und F1-Werte von 0.705 bzw. 0.706 erreichen. Deutlich

Tabelle 4.14: Vergleich der YOLOv11- und YOLOv12-Modelle hinsichtlich Validierungsleistung und Trainingszeit für 30 Epochen. Ergebnisse geordnet nach mAP50-95 in absteigender Reihenfolge.

Modell	**mAP50-95**	**mAP50**	**F1**	**Precision**	**Recall**	**Trainingszeit [min]**
YOLOv12m	0.399	0.751	0.712	0.728	0.697	11.26
YOLOv12l	0.397	0.754	0.715	0.737	0.696	15.54
YOLOv11l	0.396	0.747	0.709	0.720	0.699	11.11
YOLOv11m	0.391	0.743	0.705	0.712	0.698	9.24
YOLOv12s	0.391	0.742	0.706	0.734	0.679	7.60
YOLOv11s	0.379	0.724	0.694	0.712	0.676	6.13
YOLOv12n	0.362	0.705	0.673	0.703	0.645	6.96
YOLOv11n	0.353	0.689	0.656	0.688	0.633	6.12

schwächer performen die kleineren Modelle YOLOv12n (mAP50-95 = 0.362, F1 = 0.673) und YOLOv11n (mAP50-95 = 0.353, F1 = 0.656), die zugleich die kürzesten Trainingszeiten aufweisen.
Insgesamt zeigt sich ein klarer Zusammenhang zwischen Modellgröße und Genauigkeit: größere Modelle liefern höhere mAP- und F1-Werte, erfordern jedoch längere Trainingszeiten. Unter Abwägung von Genauigkeit und Effizienz wurde YOLOv12m als Referenzmodell für die praktische Anwendung ausgewählt. Es bietet mit einer mAP50-95 von 0.399, einem F1 von 0.712 und einer Trainingszeit von 11.1 Minuten das ausgewogenste Verhältnis zwischen Lokalisierungsgenauigkeit, F1-Score und Trainingszeit. Zusätzlich ist noch die Inferenzzeit zu erwähnen die in Kapitel 4.2.3 ermittelt wurde, und unter der der beiden andern Modell liegt (YOLOv12l, YOLOv11l).

4.3.2 Training und Tuning des besten Modells

Nach der Auswahl des Modells YOLOv12m erfolgte im nächsten Schritt die Untersuchung verschiedener Trainingskonfigurationen. Ziel war es, den Einfluss unterschiedlicher Augmentierungsparameter auf die Detektionsleistung zu analysieren und das Modell gezielt für die Baumdetektion zu optimieren. Die Hyperparameter wurden dabei unverändert beibehalten, da sie sich in vorangegangenen Tests bereits als geeignet erwiesen hatten.
Die Konfigurationen folgten einem schrittweisen Vorgehen: Zeigte eine Variante eine bessere Leistung als die vorherige, diente sie als Grundlage für weitere Anpassungen. Ergaben sich keine Verbesserungen, wurde die leistungsstärkste Konfiguration als Ausgangspunkt beibehalten.
Als Konfiguration 0 wurde das Ergebnis aus dem vorherigen Kapitel mit den dort verwendeten Augmentierungen übernommen. In Konfiguration 1 wurde ein Training ohne jegliche Augmentierung durchgeführt, um die Effekte der zufälligen Veränderungen isoliert bewerten zu können. Darauf aufbauend folgte ein Training mit den Standardparametern und einer zusätzlichen zufälligen Drehung im Bereich von $-15°$ bis $+15°$ (Konfiguration 2),

sowie ein weiteres mit einer erweiterten Drehung von −30° bis +30° (Konfiguration 3). Aufbauend auf Konfiguration 3 wurde in Konfiguration 4 zusätzlich eine vertikale Spiegelung mit einer Wahrscheinlichkeit von 0.5 aktiviert. In Konfiguration 5 kam auf derselben Basis die *Copy-Paste*-Funktion mit einer Wahrscheinlichkeit von 0.25 hinzu. Abschließend wurde in Konfiguration 6 die zufällige Variation des *hue*-Werts von 0.015 auf 0.03 erhöht.

Tabelle 4.15: Übersicht der Trainingskonfigurationen für YOLOv12m

Nr.	Augmentierung / Anpassung	Beschreibung
0	Standard	Parameter aus Kapitel 4.2.2
1	Keine Augmentierung	Referenztraining ohne Datenaugmentation
2	Rotation ±15°	Standardparameter + leichte Rotation
3	Rotation ±30°	Standardp. + Erweiterter Rotationsbereich
4	Rot. ±30° + Flip (p=0.5)	Zusätzliche vertikale Spiegelung
5	Rot. ±30° + Copy-Paste (p=0.25)	Einfügen synthetischer Objekte
6	Rot. ±30° + Hue=0.03	Erhöhte Farbvariation

Für alle Konfigurationen wurden die Trainingsdaten der Städte München und Augsburg genutzt, Regensburg diente als unabhängiger Validierungsdatensatz.

Die Ergebnisse für die verschiedenen Konfigurationen sind nach der genannten Reihenfolge geordnet in Tabelle 4.16 zusammengefasst. Ohne Augmentierung erreichte Konfiguration 1 den niedrigsten mAP50-95 mit 0.354 und einen F1-Wert von 0.660. In Konfiguration 2, bei der eine Rotationsaugmentierung im Bereich von −15° bis +15° verwendet wurde, stieg der mAP50-95 auf 0.400 und der F1-Wert auf 0.706. Die Erweiterung des Rotationsbereichs auf −30° bis +30° in Konfiguration 3 führte zu einem mAP50-95 von 0.405 und einem F1-Wert von 0.715. Bei Konfiguration 4, in der zusätzlich eine vertikale Spiegelung mit einer Wahrscheinlichkeit von 0.5 angewendet wurde, lagen die Werte bei 0.398 mAP50-95 und 0.708 (F1). Die Einführung der Copy-Paste-Funktion in Konfiguration 5 (Wahrscheinlichkeit 0.25) ergab nahezu identische Werte mit einem mAP50-95 von 0.399 und einem F1-Wert von 0.712. In Konfiguration 6, bei der die Farbwertvariation (hue) von 0.015 auf 0.03 erhöht wurde, blieb der mAP50-95 mit 0.400 konstant, während der F1-Wert auf 0.716 stieg. Insgesamt erzielte Konfiguration 3 den höchsten mAP50-95, während Konfiguration 6 den höchsten F1-Wert aufwies.

Basierend auf der Trainingskurve in Abbildung 4.15 lässt sich erkennen, dass das Modell aus Konfiguration 3 eine stabile und gleichmäßige Konvergenz über alle betrachteten Metriken aufweist. Die Trainingsverluste (box_loss, cls_loss und dfl_loss) sinken stetig und nähern sich gegen Ende der 30 Epochen einem stabilen Minimum an. Auch die entsprechenden Validierungsverluste zeigen einen parallelen Verlauf ohne Anzeichen von Überanpassung, was auf ein gut generalisierendes Modell hinweist. Die Metriken *Precision* und *Recall* steigen kontinuierlich an und stabilisieren sich gegen Ende des Trainings bei Werten um 0.70, was eine zuverlässige Erkennungsleistung bei gleichzeitig ausgewogenem Verhältnis von Präzision und Trefferquote belegt. Besonders deutlich ist die Konvergenz

Tabelle 4.16: Validierungsergebnisse der verschiedenen Trainingskonfigurationen für YOLOv12m über 30 Epochen. Die Ergebnisse sind nach Konfigurationen aufsteigend sortiert.

Konfig	mAP50–95	mAP50	Precision	Recall	F1
0	0.399	0.751	0.728	0.697	0.712
1	0.354	0.686	0.711	0.615	0.660
2	0.400	0.750	0.710	0.701	0.706
3	0.405	0.757	0.728	0.703	0.715
4	0.398	0.750	0.723	0.694	0.708
5	0.399	0.751	0.728	0.697	0.712
6	0.400	0.754	0.728	0.704	0.716

in den Kennwerten mAP50 und mAP50-95 zu erkennen: Beide Metriken steigen über die ersten 15 Epochen stark an und erreichen anschließend ein Plateau, das eine konsistente Modellleistung über verschiedene IoU-Schwellen hinweg bestätigt. Auf Grundlage dieser stabilen Konvergenzverläufe und der in Tabelle 4.16 dargestellten Ergebnisse wurde das in Konfiguration 3 trainierte Modell für die praktische Anwendung übernommen. Es stellt die leistungsstärkste Variante dar, da es die höchste mAP50-95 erreichte und gleichzeitig eine robuste, gleichmäßig verlaufende Trainingsdynamik zeigte. Damit bildet das resultierende Modell die Grundlage für die weiteren Detektionsschritte im Rahmen der Anwendung.

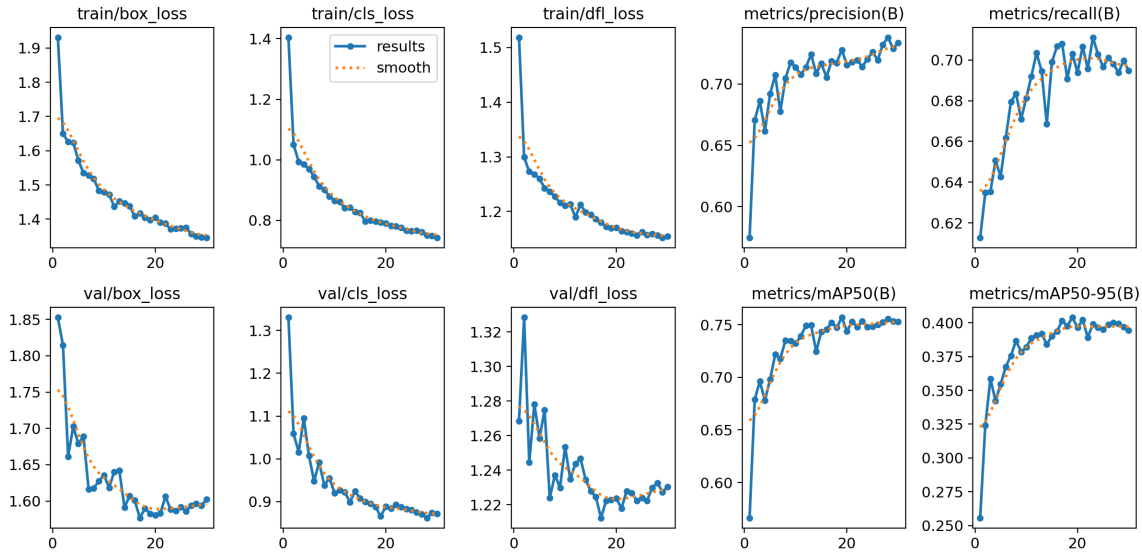

Abbildung 4.15: Trainings- und Validierungsverläufe des YOLOv12m-Modells (Konfiguration 3) mit Darstellung der wichtigsten Verlust- und Leistungsmetriken über den Trainingszeitraum.

4.3.3 Optimierung der Vorhersage Parameter

Nachdem die beste Trainingskonfiguration ermittelt worden war, begann die Optimierung der Parameter für die praktische Vorhersage. Der Vorhersagealgorithmus verfügt über zwei zentrale Stellgrößen: Den Konfidenz-Schwellenwert (direkt in *Ultralytics-Train*-Funktion einzustellen) und den IoU-Schwellenwert der Non-Maximum-Suppression (Filterung der Zusammengefassten Ergebnisse). Ziel war es, die optimale Kombination beider Parameter zu finden, um eine möglichst hohe Erkennungsleistung zu erzielen. Die Optimierung erfolgte auf Kacheln mit einer Größe von 250x250m und einer Überlappung von 25%, was einer Überdeckung von 62.5m entspricht. Diese Konfiguration gewährleistet durch die anschließende Anwendung der NMS eine vollständige Flächenabdeckung. Für das Konfidenz-Tuning wurden Schwellen von 5%, 10%, 20%, 30%, 40% und 50% getestet, während die NMS-Schwellen in Intervallen von 10% im Bereich von 10% bis 60% variiert wurden. Die Bewertung der Parameterkombinationen erfolgte anhand des Validierungsdatensatzes, um eine unabhängige Optimierung gegenüber dem Testdatensatz sicherzustellen. Ziel war die Maximierung des F1-Scores, also ein möglichst ausgewogenes Verhältnis zwischen Präzision und Recall. Dabei wurde eine leicht höhere Präzision bevorzugt, um Fehlklassifikationen zu minimieren. Das Tuning erfolgte ausschließlich auf nach Höhe gefilterten Daten, in Anlehnung an den Baumdatensatz des LDBV, der ebenfalls nur Bäume ab einer Höhe von 5m enthält. Die Ergebnisse wurden abschließend in Form von Heatmaps visualisiert, um die Wechselwirkungen der Parameter anschaulich darzustellen.
Die Heatmap des F1-Scores in Abbildung 4.16 verdeutlicht, dass die besten Ergebnisse bei einem NMS-Schwellenwert von 20% und Konfidenz-Schwellen von 5%, 10% und 20% mit einem F1-Wert von 0.767 erzielt wurden. Diese Kombinationen lieferten identische Werte. Nahezu gleichwertige Resultate wurden zudem bei einem NMS-Schwellenwert von 30% in Verbindung mit denselben Konfidenz-Stufen erreicht (F1 = 0.766) sowie bei einem NMS-Wert von 40% (F1 = 0.765). Die Unterschiede zwischen den besten Konfigurationen lagen lediglich im Bereich der dritten Nachkommastelle (0.001–0.002) und sind damit vernachlässigbar. Das schlechteste Ergebnis wurde bei einer Konfidenz von 50% und einem NMS-Schwellenwert von 10% beobachtet.

Abbildung 4.17 zeigt, dass die höchste Präzision bei einer Konfidenz-Schwelle von 50% und einer NMS-Schwelle von 10% erzielt wurde. Das schlechteste Ergebnis trat hingegen dreimal auf, jeweils bei einer NMS-Schwelle von 60% in Kombination mit Konfidenz-Schwellen von 5%, 10% und 20%. Damit liegen der beste und die schlechtesten Werte in den diagonal gegenüberliegenden Ecken der Darstellung, wobei ein nahezu gleichmäßiger Übergang der Präzisionswerte entlang dieser Diagonalen erkennbar ist.
Abbildung 4.18 zeigt ein umgekehrtes Bild im Vergleich zur Präzision. Die höchsten Recall-Werte wurden bei einer NMS-Schwelle von 50% und Konfidenz-Schwellen von 5%, 10% und 20% erzielt, während das schlechteste Ergebnis bei einer Konfidenz-Schwelle von 50% und einer NMS-Schwelle von 10% auftrat. Auch hier zeigt sich ein fließender Übergang der Werte über die Parameterkombinationen hinweg.
In Abbildung 4.19 ist erkennbar, dass die höchste Lokalisierungsgenauigkeit (mAP50) bei einer Konfidenz-Schwelle von 5% und einer NMS-Schwelle von 50% erreicht wurde. Die

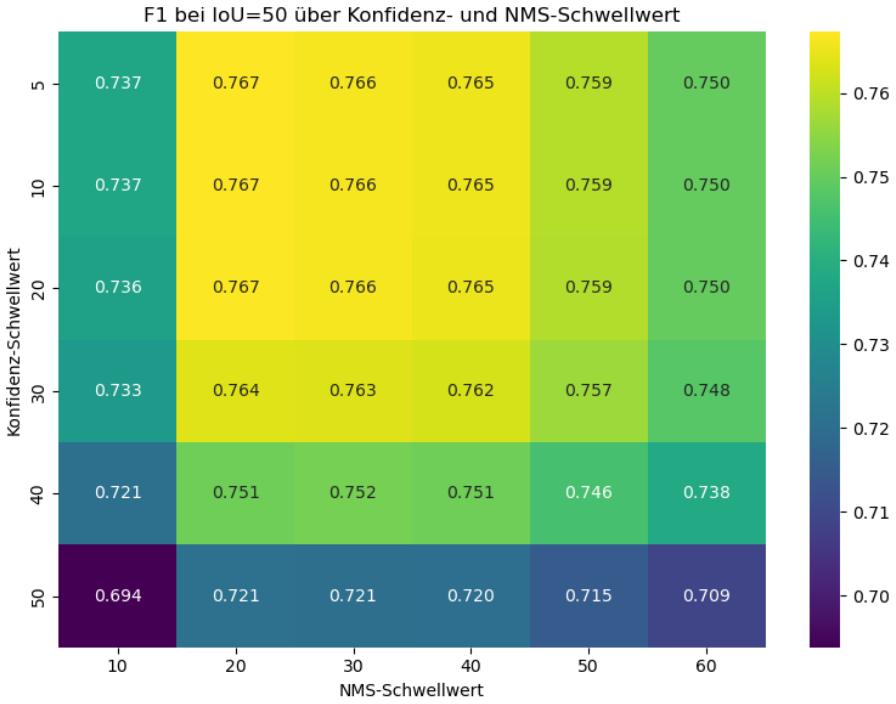

Abbildung 4.16: Heatmap des F1-Scores in Abhängigkeit von Konfidenz- und NMS-Schwellenwerten zur Optimierung der Vorhersageparameter.

Ergebnisse bei gleicher Konfidenz-Schwelle und NMS-Schwellen von 40% (0.758) sowie 60% (0.759) weichen nur minimal ab und sind somit vernachlässigbar schlechter. Die niedrigste mAP50 wurde bei einer Konfidenz von 50% und einer NMS-Schwelle von 10% erzielt. Ein ähnliches Muster zeigt sich in Abbildung 4.20 für die mAP50-95: Der beste Wert wurde bei einer Konfidenz-Schwelle von 5% und einer NMS-Schwelle von 60% mit 0.281 erreicht, während das schlechteste Ergebnis erneut bei einer NMS-Schwelle von 10% und einer Konfidenz von 50% mit 0.213 beobachtet wurde.

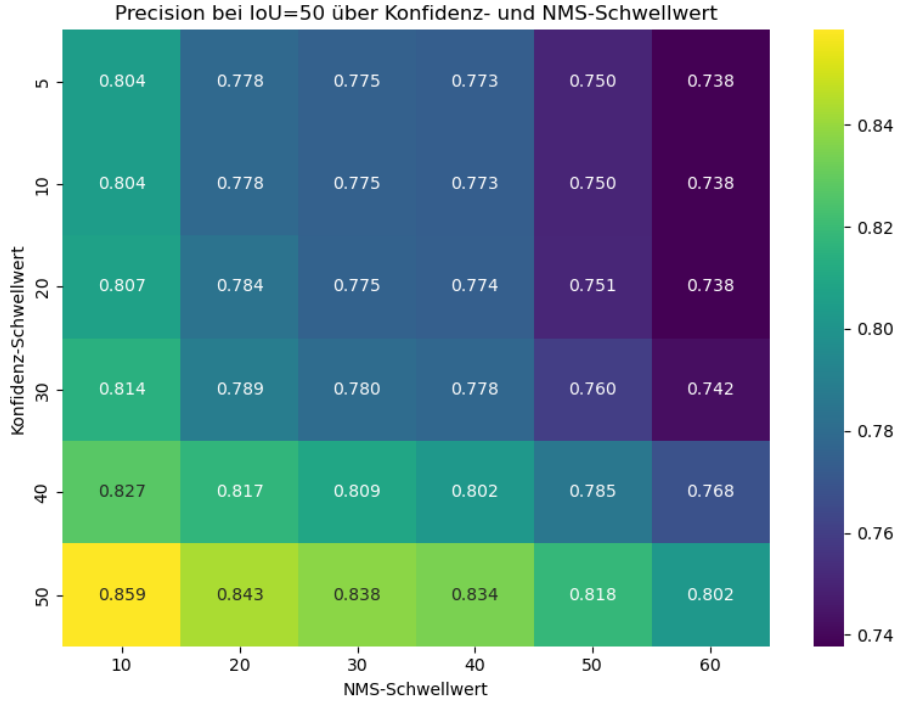

Abbildung 4.17: Heatmap der Präzision (Precision) für verschiedene Kombinationen von Konfidenz- und NMS-Schwellenwerten.

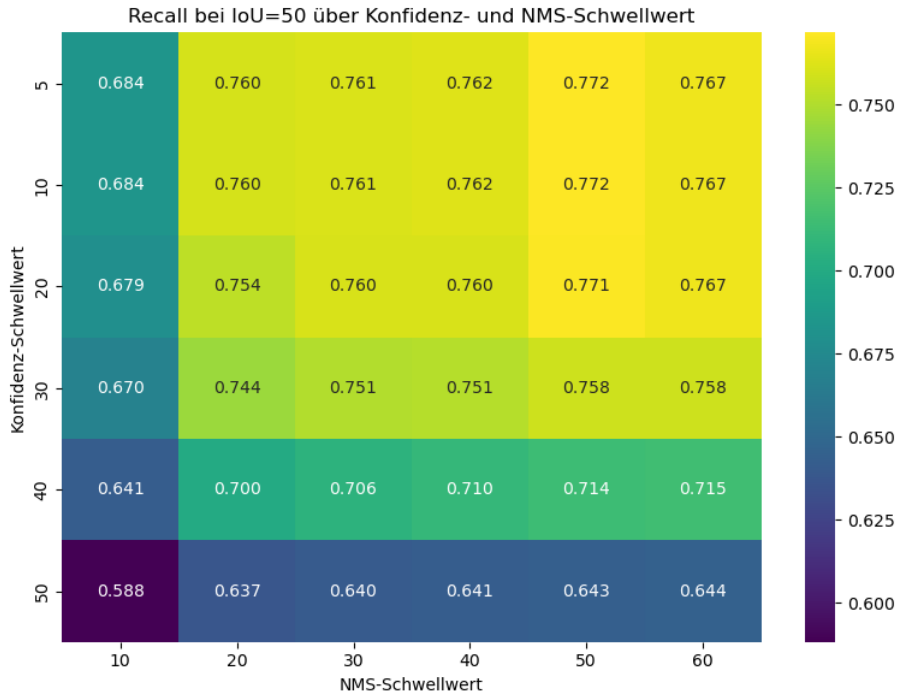

Abbildung 4.18: Heatmap des Recall-Werts in Abhängigkeit der getesteten Konfidenz- und NMS-Schwellenwerte.

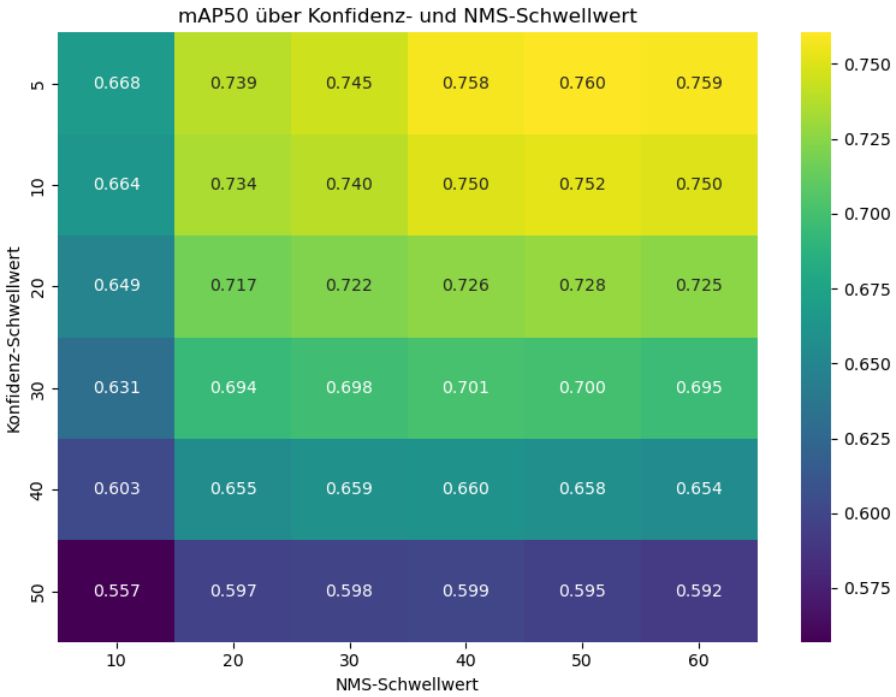

Abbildung 4.19: Heatmap der mittleren Genauigkeit bei 50% IoU (mAP50) für unterschiedliche Vorhersageparameterkombinationen

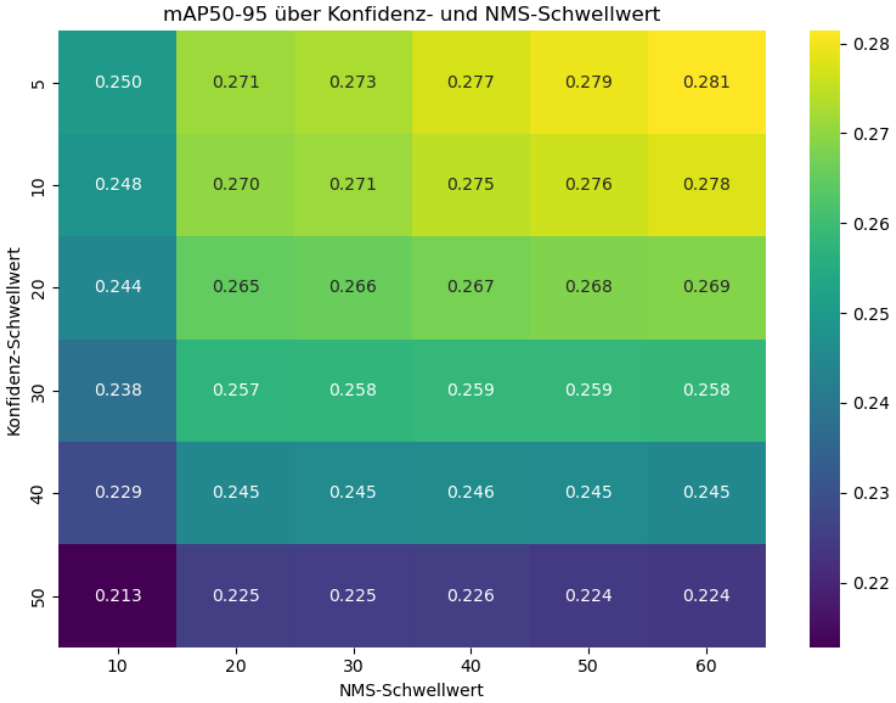

Abbildung 4.20: Heatmap der mittleren Genauigkeit über den IoU-Bereich 50–95% (mAP50–95) zur Bewertung der Gesamtdetektionstreue in Abhängigkeit von Konfidenz- und NMS-Schwellenwerten.

Für die praktische Anwendung wurde die Parameterkombination Konfidenz gleich 5% und NMS gleich 40% gewählt. Diese Kombination stellt einen ausgewogenen Kompromiss zwischen Erkennungsleistung und Box-Lokalisierungsgenauigkeit dar. Mit einem F1-Score von 0.7654 liegt sie nur geringfügig unter dem globalen Maximum von 0.7673, erreicht jedoch gleichzeitig eine mAP50-95 von 0.2771.

4.3.4 Vorhersage auf Testgebieten

Für die abschließenden Experimente auf unabhängigen Testdaten wurde das zuvor trainierte Modell (YOLOv12m - Konfiguration 3) mit den zuvor bestimmten Parametern (Kapitel 4.3.3) verwendet. Da diese Parameter auf Metriken basieren, die ausschließlich auf Bäumen mit einer Höhe von über 5 m berechnet wurden, wurde derselbe Höhenfilter auch hier konsequent angewendet, um eine konsistente Bewertungsgrundlage sicherzustellen. Die Kacheln wurden mit einer Größe von 640 px und einer Überlappung von 25% verarbeitet. Die minimale Bounding-Box-Größe betrug 5px (entspricht 1 m bei einer Bodenauflösung von 0.2m/px); das maximale Seitenverhältnis der Boxen wurde auf 2.5 festgelegt. Diese Einstellungen basieren auf zuvor durchgeführten Untersuchungen und entsprechen den Standardparametern der Detektionspipeline; sie wurden im Rahmen der finalen Versuche nicht weiter getunt. Das Vorgehen war für alle Kategorien und beide Teststädte identisch: Zunächst erfolgte die Vorhersage auf den Testkacheln, anschließend die Filterung der Vorhersagen mittels NMS sowie die nachgelagerte Höhenfilterung, bei der ausschließlich Objekte mit einer ermittelten Höhe von über 5 m beibehalten wurden. Dadurch fließen nur relevante Baumobjekte in die abschließende Bewertung ein.
Die Berechnung der Evaluationsmetriken erfolgte gemäß der in Kapitel 3.2.5 beschriebenen Implementierung. Grundlage bildet der IoU, wobei Objekte mit einem IoU von mindestens 0.5 als korrekt erkannt gelten (True Positives). Precision, Recall und F1 wurden entsprechend als Precision, Recall und F1 bei einem IoU von 0.5 berechnet.
Die Inferenzzeit für beide Städte und alle Kategorien betrug bei einer Eingabemenge von 80 Kacheln insgesamt 38 s. Sie umfasst neben der eigentlichen Modellvorhersage auch die Umwandlung der YOLO-Labels in georeferenzierte Shapefiles. Die anschließende NMS-Filterung, bei der die Vorhersagen innerhalb der Shapefiles bereinigt werden, dauerte 43 s. Für die Höhenfilterung, bei der für jede Kachel das nDOM sowie die zugehörigen Gebäudeumringe geladen werden, wurden 77 s benötigt. Insgesamt ergibt sich somit eine Gesamtverarbeitungszeit von 158s für 80 Kacheln mit einer Fläche von jeweils 250×250m (entsprechend 0,0625 km²). Hochgerechnet entspricht dies einer durchschnittlichen Rechenzeit von 31,6 s pro km² für einzelne Kacheln.

Abbildung 4.21–4.24 zeigen exemplarisch die Ergebnisse der einzelnen Verarbeitungsschritte für eine Kachel der Kategorie 3 aus Ingolstadt. Abbildung 4.21 zeigt, dass durch die Höhenfilterung Vegetationsobjekte unter 5 m entfernt werden, wodurch die Referenzdaten stärker auf baumähnliche Strukturen fokussiert werden. Abbildung 4.22 zeigt, wie durch die NMS-Filterung überlappende Vorhersagen zusammengefasst und redundante Detektionen entfernt werden. Die anschließende Höhenfilterung (Abbildung 4.23) führt zu einer weiteren Bereinigung, insbesondere in Bereichen mit niedrigem Bewuchs. Der finale Vergleich in Abbildung 4.24 zeigt eine Gegenüberstellung der gefilterten Referenz und der finalen Vorhersage. Die Zusammenfassung der Vorhersage Ergebnisse je Stadt werden danach präsentiert.

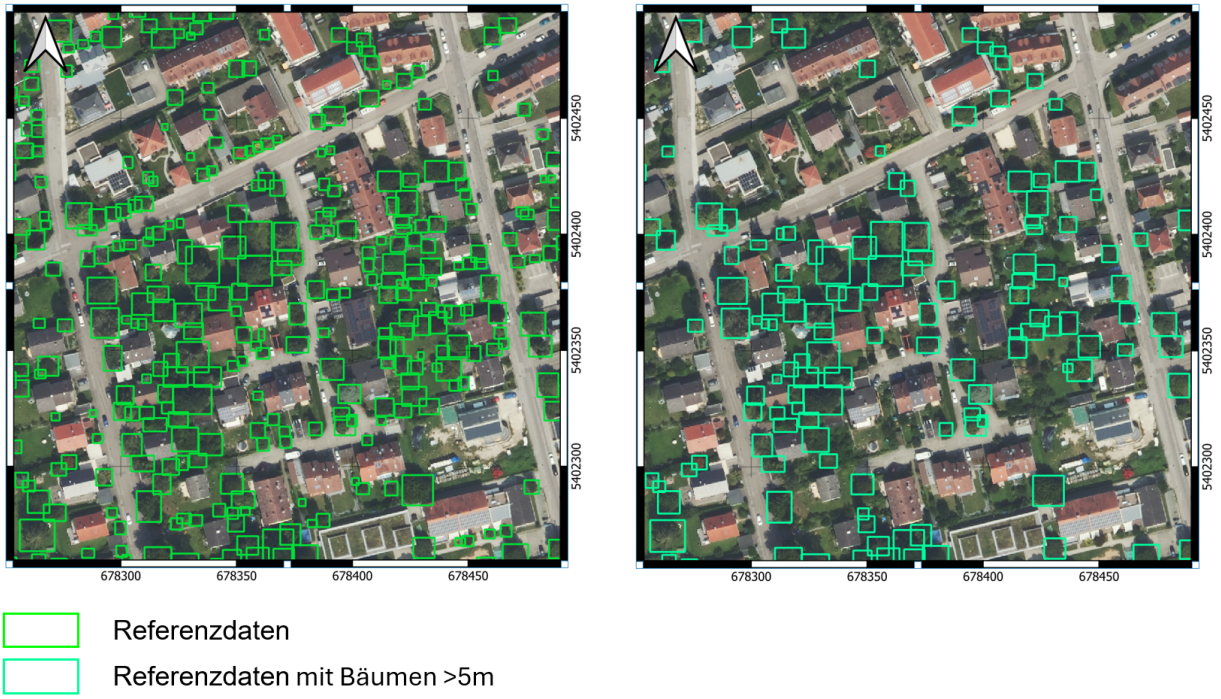

Abbildung 4.21: Gegenüberstellung von ungefilterten Referenzdaten (links) und nach Höhe (>5m) gefilterten Referenzendaten (rechts) aus Ingolstadt, Kategorie 3. Quelle des Orthophotos: LDBV (2025e).

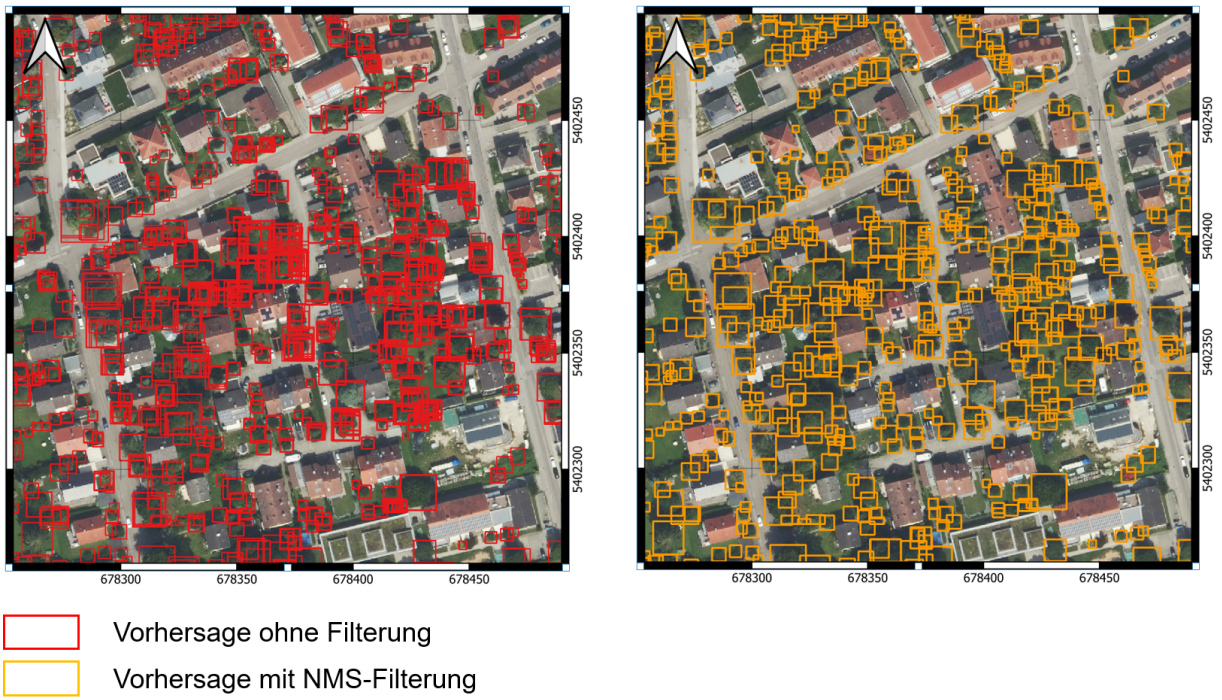

Abbildung 4.22: Gegenüberstellung von ungefilterten Vorhersageergebnissen (links) und mit NMS gefilterten Vorhersagen (rechts) aus Ingolstadt, Kategorie 3. Quelle des Orthophotos: LDBV (2025e).

Abbildung 4.23: Gegenüberstellung von der mit NMS gefilterteten Vorhersage (links) und nach Höhe (>5m) gefilterten Vorhersage (rechts) aus Ingolstadt, Kategorie 3. Quelle des Orthophotos: LDBV (2025e).

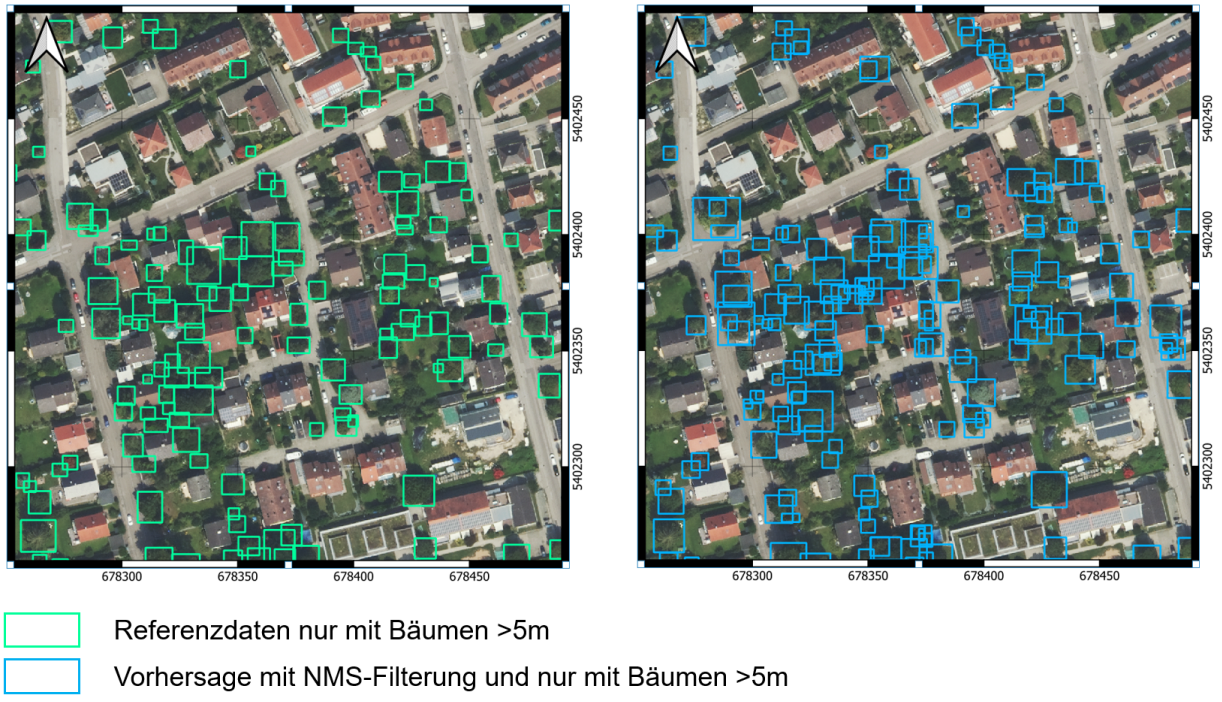

Abbildung 4.24: Gegenüberstellung von nach Höhe (>5m) gefilterten Referenzen (links) und mit NMS und nach Höhe (>5m) gefilterten Vorhersagen (rechts) aus Ingolstadt, Kategorie 3. Quelle des Orthophotos: LDBV (2025e).

Nürnberg

Die in Tabelle 4.17 dargestellten Werte zeigen die mittleren Ergebnisse der vier Testkategorien für die Stadt Nürnberg. Die Werte basieren jeweils auf dem Durchschnitt von zehn Testkacheln pro Kategorie. Die höchste Lokalisierungsgenauigkeit wurde in Kategorie 1 erzielt. Diese weist mit einem mAP50 von 0.800 sowie einem F1-Wert von 0.807 die besten Leistungskennzahlen auf. Die Präzision ist mit 0.794 geringer als der Recall mit 0.830. In Kategorie 2 zeigt sich ein umgekehrtes Verhältnis: Die Präzision (0.698) liegt leicht über dem Recall (0.679). In den Kategorien 3 und 4 sind Präzision und Recall annähernd ausgeglichen (Diff <=1%). In Kategorie 3 beträgt die Präzision 0.736 und der Recall 0.755, während in Kategorie 4 Werte von 0.749 beziehungsweise 0.775 erreicht wurden. Über alle Kategorien hinweg zeigen sich nur geringe Unterschiede zwischen Präzision und Recall.

Tabelle 4.17: Durchschnittliche Leistungsmetriken (mAP50, mAP50–95, Precision, Recall, F1) für die vier Testkategorien in Nürnberg. Die Werte stellen den Mittelwert der Ergebnisse von jeweils zehn Testkacheln pro Kategorie dar.

Kategorie	**mAP50**	**mAP50-95**	**Precision**	**Recall**	**F1**
1	0.800	0.299	0.794	0.830	0.807
2	0.664	0.212	0.698	0.679	0.686
3	0.726	0.240	0.736	0.755	0.745
4	0.743	0.236	0.749	0.775	0.758

In Abbildung 4.25 sind die Vorhersageergebnisse für die Kategorien 1 bis 4 dargestellt. In Kategorie 1, einer dicht bebauten städtischen Umgebung mit Straßenbäumen, wurden freistehende Bäume zuverlässig erkannt. Besonders im nördlichen, offener bebauten Bildbereich mit größeren Baumstrukturen erfolgten präzise Detektionen, sowie auch innerhalb von Innenhöfen. Bei eng aneinanderstehenden Baumgruppen treten mehrfach überlappende Bounding Boxes auf, wobei meist eine größere Box den gesamten Baum umfasst. Fehlklassifikationen traten damit vor allem bei optisch schwer trennbaren Baumgruppen auf. In Kategorie 2, einem Wohngebiet mit sehr dichter Bebauung und hoher Bodenversiegelung, stimmen die freistehenden Bäume ebenfalls gut mit den Referenzdaten überein, wobei die meisten Detektionen in Gärten liegen. In dichter bewachsenen Bereichen, insbesondere entlang von Grundstücksgrenzen, zeigt das Modell eine geringere Trennschärfe bei der Abgrenzung einzelner Bauminstanzen. Die Objektrahmen der Referenzen und Vorhersagen stimmen bei freistehenden Bäumen weitgehend überein. Kategorie 3 umfasst großflächige, in Reihen angeordnete Wohnblocks mit größeren Abständen zwischen den Gebäuden. Freistehende Bäume wurden zuverlässig erkannt, während bei dichter stehenden Bäumen überlappende Vorhersagen auftreten, ähnlich wie in den Kategorien 1 und 2. Kategorie 4 zeigt ein Gebiet mit industriell geprägten Flächen, durchmischt mit Wohnbebauung, in dem ein Großteil der Vegetation in privaten Gärten liegt. Für alle Kategorien konnten keine Fehlklassifikationen auf Gebäuden festgestellt werden.

Abbildung 4.25: Überlagerung der nach Höhe (> 5 m) gefilterten Referenzen (türkis) und der mit NMS sowie nach Höhe (> 5 m) gefilterten Vorhersagen (blau) aus Nürnberg. Oben links: Kategorie 1, oben rechts: Kategorie 2, unten links: Kategorie 3, unten rechts: Kategorie 4.
Quelle des Orthophotos: LDBV (2025e).

Ingolstadt

Die in Tabelle 4.18 dargestellten Werte zeigen die mittleren Ergebnisse der vier Testkategorien für die Stadt Ingolstadt. Die Werte basieren jeweils auf dem Durchschnitt von zehn Testkacheln pro Kategorie. Die höchste Lokalisierungsgenauigkeit wurde in Kategorie 1 erreicht. Diese weist mit einem mAP50 von 0.838 sowie einem F1-Wert von 0.843 die besten Leistungskennzahlen auf. Präzision (0.845) und Recall (0.843) liegen hier nahezu gleichauf und zeigen eine sehr geringe Abweichung. In Kategorie 2 liegt der Recall (0.759) leicht über der Präzision (0.731). In Kategorie 3 ist das Verhältnis umgekehrt: Die Präzision (0.752) übersteigt den Recall (0.718). In Kategorie 4 zeigen beide Werte wieder ein ähnliches Verhältnis mit leicht höherer Präzision (0.797) im Vergleich zum Recall (0.768). Über alle Kategorien hinweg liegen die Unterschiede zwischen Präzision und Recall im geringen Bereich. In den meisten Fällen sind beide Kennwerte ausgewogen, wobei einzelne Kategorien leichte Verschiebungen zugunsten einer der beiden Metriken aufweisen. Insgesamt zeigen die Ergebnisse ein stabiles Verhältnis zwischen Präzision und Recall über alle vier Kategorien. In Abbildung 4.26 sind die Ergebnisse für die Kategorien 1 bis 4 der Stadt Ingolstadt

Tabelle 4.18: Durchschnittliche Leistungsmetriken (mAP50, mAP50–95, Precision, Recall, F1) für die vier Testkategorien in Ingolstadt. Die Werte stellen den Mittelwert der Ergebnisse von jeweils zehn Testkacheln pro Kategorie dar.

Kategorie	mAP50	mAP50-95	Precision	Recall	F1
1	0.838	0.338	0.845	0.843	0.843
2	0.718	0.274	0.731	0.759	0.742
3	0.679	0.225	0.752	0.718	0.733
4	0.753	0.284	0.797	0.768	0.780

dargestellt. In Kategorie 1 ist eine hohe Bebauungsdichte erkennbar. Freistehende Bäume werden zuverlässig detektiert, während bei dicht beieinanderstehenden Baumgruppen mehrere Vorhersagen pro Baum auftreten, wodurch die Abgrenzung einzelner Instanzen nicht immer eindeutig ist. Die Objektrahmen der Vorhersagen sind tendenziell etwas größer als die der Referenzen. Im südöstlichen Bereich, auf dem dargestellten Parkplatz, stimmen Referenzen und Vorhersagen sehr gut überein. Hier sind hauptsächlich freistehende Bäume zusehen. In Kategorie 2 werden Straßenbäume, auch wenn sie eng beieinanderstehen, zuverlässig erkannt. In dichter bewachsenen Baumgruppen zeigt sich derselbe Effekt wie in Kategorie 1: Mehrfachdetektionen und geringere Trennschärfe zwischen benachbarten Baumkronen. Die Objektrahmen von Referenz und Vorhersage stimmen insgesamt gut überein. Kategorie 3 zeigt ein überwiegend bebautes Wohngebiet, in dem sich dieselbe Systematik beobachten lässt. Freistehende Bäume werden klar detektiert, während in Bereichen dichter Vegetation mehrere überlappende Vorhersagen auftreten. Kategorie 4 zeigt ein Industrieareal mit angrenzendem Parkplatz. Optisch besteht eine hohe Übereinstimmung zwischen Referenz- und Vorhersagedaten. Es finden sich zahlreiche freistehende Bäume, bei welchen Referenzen und Vorhersagen sehr gut übereinstimmen.

Abbildung 4.26: Überlagerung der nach Höhe (> 5 m) gefilterten Referenzen (türkis) und der mit NMS sowie nach Höhe (> 5 m) gefilterten Vorhersagen (blau) aus Ingolstadt. Oben links: Kategorie 1, oben rechts: Kategorie 2, unten links: Kategorie 3, unten rechts: Kategorie 4.
Quelle des Orthophotos: LDBV (2025e).

5 Diskussion

Im Folgendem werden die Ergebnisse der Experimente zusammengefasst, analysiert und bewertet. Desweiteren wird der entwickelte Arbeitsablauf kritisch betrachtet und auf mögliche Verbesserungen hingewiesen. Zuletzt wird ein Ausblick für zukünftige Verbesserungen und Einsatzmöglichkeiten gegeben.

5.1 Datenvorbereitung

Das Szenen-Sampling hat in Anbetracht der Ergebnisse aus Tabelle 4.2 insgesamt gut funktioniert. Die Flächenanteile der vier Hauptkategorien dominieren im gesamten Datensatz und bestätigen damit die korrekte Umsetzung der Sampling-Logik. Die fehlenden Kachelbeispiele für Augsburg in Kategorie 3 weisen jedoch auf eine strukturelle Limitierung des Ansatzes hin: In Städten mit geringer Flächenverfügbarkeit bestimmter Landnutzungsklassen können die gewählten 250×250 m-Kacheln zu groß sein, um ausreichend viele möglichst homogene Ausschnitte zu erfassen. Eine mögliche Lösung wäre die Verwendung kleinerer Kachelgrößen – etwa 125×125 m oder direkt in der Größe des Modell-Inputs (z. B. 640 px). Dadurch ließen sich auch kleinräumigere, heterogene Bereiche einbeziehen und die Flächenrepräsentation gleichmäßiger gestalten. Diese Reduktion der Kachelgröße würde jedoch den Labeling-Aufwand erheblich erhöhen, da bei gleicher Gesamtfläche eine deutlich größere Anzahl an Szenen zu verarbeiten wäre. Künftige Arbeiten müssten daher zwischen räumlicher Vielfalt und manuellem Aufwand abwägen. Ein weiterer Aspekt betrifft die gewählte Überlappung der 250 m-Kacheln beim Zuschneiden der Trainingsdaten: Zwar wurde dadurch sichergestellt, dass alle Objekte mindestens einmal im Training vorkommen, gleichzeitig führte dies aber zu einer Mehrfachrepräsentation einzelner Bauminstanzen. Diese können während des Trainings ungewollt eine höhere Gewichtung erhalten und damit die Modelloptimierung leicht verzerren. Alternativ könnten auch Trainingsdatensätze ohne überlappende Kacheln verwendet werden, wobei Bounding Boxes mit zu geringer Fläche oder einem ungünstigen Seitenverhältnis vorab gefiltert werden.

Der Labeling-Prozess gestaltete sich insgesamt als sehr zeitaufwendig. Pro Kachel wurden durchschnittlich zwischen 171 und 247 Bäume annotiert (Tabelle 4.3). Dabei traten häufig Unsicherheiten auf, welche Objekte eindeutig als Bäume zu klassifizieren waren. Besonders in dicht bewachsenen Bereichen war die Abgrenzung zwischen Bäumen und großflächigen Sträuchern schwierig. Der Vergleich des Schattenwurfs von Gebäuden und Vegetationsstrukturen konnte hierbei teilweise unterstützen, erlaubte jedoch keine durchgehend verlässliche Differenzierung. In der Draufsicht ist es grundsätzlich anspruchsvoll,

die vertikale Ausprägung der Vegetation zu beurteilen, wodurch Fehleinschätzungen nicht vollständig zu vermeiden sind. Die Bedeutung des manuellen Labelings ist im gesamten Prozess als sehr hoch einzuschätzen, da diese ersten Beispiele die Grundlage für das nachfolgende halbautomatische Labeling bilden. Werden in der manuellen Phase Bäume übersehen, interpretiert das Modell diese Flächen als Hintergrund, was langfristig zu Fehlklassifikationen führen kann. Zudem ist anzuführen, dass der Labeling-Stil stark von der jeweiligen Person abhängt: Das Modell übernimmt gewissermaßen die individuelle Label-Strategie des Bearbeiters. Der Übergang zum halbautomatischen Labeling konnte den Prozess deutlich beschleunigen. Das bereits trainierte Modell war in der Lage, eine Vielzahl an Objekten korrekt vorzuschlagen und teilweise sogar Bäume zu identifizieren, die dem menschlichen Bearbeiter entgangen wären. Mit jeder Erweiterung des Datensatzes verbesserte sich die Modellleistung spürbar, was auf den zunehmenden Abdeckungsgrad unterschiedlicher Stadtstrukturen und Beleuchtungssituationen zurückzuführen ist. Dennoch erfordert das halbautomatische Labeling weiterhin eine sorgfältige Kontrolle: Alle vom Modell vorgeschlagenen Labels müssen überprüft und fehlerhafte Annotationen manuell korrigiert werden, um eine systematische Fehlerfortpflanzung zu vermeiden. Die Gesamtanzahl der gelabelten Bäume pro Stadt reflektiert den Einfluss des Szenen-Samplings. Von Nürnberg bis Ingolstadt liegt die Zahl jeweils zwischen etwa 6.000 und 7.000 Bäumen. Kategorie 3 („Discontinuous Urban Fabric 30–50 %“) weist dabei die höchsten Baumzahlen auf, während sie in Kategorien 1 und 4 am geringsten vertreten sind. Ein erwartbares Ergebnis, da stark versiegelte Flächen (Kat. 1) und Industriegebiete (Kat. 4) typischerweise nur wenige Einzelbäume aufweisen. Abschließend lässt sich festhalten, dass der Einsatz bereits vortrainierter Modelle für künftige Labeling-Aufgaben sehr zu empfehlen ist. Auf Grundlage des bestehenden Datensatzes kann ein feingetuntes Modell weitere Bilddaten deutlich effizienter und konsistenter annotieren. Der Datensatz selbst bietet dafür eine solide Basis: Er vereint die strukturelle und radiometrische Variabilität von fünf Städten mit unterschiedlichen urbanen und architektonischen Gegebenheiten und bildet damit ein breites Spektrum städtischer Umweltsituationen ab.

5.2 Vergleich aktueller Echtzeit-Objekterkennungsarchitekturen

Der Vergleich der Testergebnisse auf dem Baumdatensatz bestätigt im Wesentlichen die bekannten Muster aus dem COCO-Datensatz: Mit zunehmender Modellgröße steigt die Lokalisierungsgenauigkeit, gleichzeitig nehmen die Inferenzzeiten aller getesteten Architekturen (YOLOv11, YOLOv12 und RT-DETR) zu. Neu und bisher undokumentiert ist hingegen das Verhalten der Trainingszeiten, da hierzu von Ultralytics keine offiziellen Angaben vorliegen. Diese Information ist entscheidend, um die erzielten Ergebnisse vollständig einordnen zu können.
Die Trainingskurven aller Architekturen und Modellgrößen (Abb. 4.7, 4.9) zeigen, dass die gewählte Epochenzahl von 200 ausreichend ist, um eine stabile Konvergenz der mAP50 zu erreichen. YOLOv11 konvergiert dabei am schnellsten, gefolgt von YOLOv12 und schließlich RT-DETR, das erst gegen Ende der 200 Epochen stabil wird. Die Verläufe der mAP50–95 deuten jedoch darauf hin, dass für eine vollständige Konvergenz etwas

mehr Trainingszeit erforderlich wäre. Für den Architekturvergleich sind die 200 Epochen dennoch ausreichend, da sich die Leistungsunterschiede deutlich abzeichnen.
Die erzielten Metriken auf dem Validierungsdatensatz übertreffen erwartungsgemäß die Werte der COCO-Referenzmodelle, was vor allem auf den Ein-Klassen-Charakter der Aufgabe zurückzuführen ist. Die Trainingszeiten variieren deutlich zwischen den Architekturen: Für YOLOv11 liegen sie zwischen 1.7 und 4.3 Stunden, für YOLOv12 zwischen 2.2 und 6.2 Stunden. Damit benötigt YOLOv12, das erste Transformer-Elemente integriert, konsistent mehr Zeit als YOLOv11. Bei den Leistungsmetriken schneidet YOLOv12 insgesamt etwas besser ab, mit Ausnahme der größten Modellvariante. So übertrifft YOLOv11x die YOLOv12x-Variante sowohl bei Lokalisierungsgenauigkeit als auch beim F1-Score, während es mit 4.28 Stunden Trainingszeit und 18.5 ms Inferenzzeit deutlich effizienter ist als YOLOv12x (6.21 Stunden, 22.6 ms).
Für Szenarien, in denen höchste Genauigkeit Priorität hat und längere Trainings- und Inferenzzeiten akzeptabel sind, ist YOLOv12x daher die geeignete Wahl. Stehen hingegen schnelle Inferenzzeiten und kürzere Trainingsläufe im Vordergrund – etwa für iteratives Hyperparameter-Tuning oder energieeffiziente Workflows – bietet YOLOv11x den besseren Kompromiss. In den kleineren Modellklassen ist eine differenzierte Abwägung sinnvoll: Während YOLOv11l bei leicht kürzerer Trainingszeit ähnliche Werte liefert, erzielt YOLOv12m eine höhere Lokalisierungsgenauigkeit und einen besseren F1-Score bei gleichzeitig geringerer Inferenzzeit.
Besonders hervorzuheben ist die Leistung von YOLOv12m. Dieses Modell zeigt über mehrere Wiederholungen eine Inferenzzeit, die schneller ist als bei den Nano- und Small-Varianten, erreicht dabei aber deutlich höhere Genauigkeiten. Es erweist sich somit als besonders geeignet für die in dieser Arbeit untersuchte Anwendungsaufgabe. Die mAP50–95 liegt zwar rund 3–4% unter den besten Modellen, der F1-Score unterscheidet sich jedoch um weniger als 1%. Damit bietet YOLOv12m bei etwa halbierter Inferenzzeit nahezu identische Erkennungsleistung und so einen optimalen Kompromiss zwischen Präzision, Geschwindigkeit und Rechenaufwand.
Unter identischen Trainingsbedingungen über 200 Epochen bleibt RT-DETR hinter den YOLO-Modellen zurück. Dies lässt sich vermutlich auf die grundlegend unterschiedliche Architektur mit Transformer-Mechanismen zurückführen, die andere Hyperparameter erfordern würde. Die getestete RT-DETR-l-Variante besitzt bei vergleichbarer Größenbezeichnung bereits rund 16 Millionen zusätzliche Parameter gegenüber den YOLO-Modellen und benötigt entsprechend längere Trainingszeiten – über 1.3 Stunden mehr als das äquivalente YOLOv12-Modell. Durch die hohe Parameteranzahl scheint ein ressourcenintensiver Trainingsprozess zu resultieren, was das Modell zeitlich ineffizienter macht.
Die Filterung abgeschnittener Labels diente dazu, den Einfluss unvollständiger Objekte auf die Lokalisierungsgenauigkeit zu untersuchen. Da die Trainingsdaten überlappende Kacheln enthalten, traten zahlreiche abgeschnittene Bounding Boxes auf, die sich für diese Analyse besonders eigneten. Bei allen drei Architekturen zeigte sich ein positiver Effekt: Die Lokalisierungsgenauigkeit (mAP50–95) stieg um bis zu 3% bei YOLOv11x und YOLOv12x sowie um 3.5% bei RT-DETR. Die besten Ergebnisse wurden bei einer Sichtbarkeitsschwelle von 30–40% erzielt. Für den F1-Score konnten keine eindeutigen Verbesserungen festgestellt werden; die Unterschiede blieben unter 1%. Insgesamt lässt sich

festhalten, dass die Filterung unvollständiger Labels die Lokalisierungsleistung messbar verbessert, ohne die Präzision und den Recall wesentlich zu beeinflussen.

5.3 Praktische Anwendung der automatisierten Baumdetektion

Bei der praktischen Anwendung des Modells lag der Fokus nicht allein auf einem Performance-Vergleich, sondern insbesondere auf Generalisierungsfähigkeit, Anwendbarkeit und Übertragbarkeit. Um diese Aspekte gezielt zu untersuchen, wurde der Datensatz nach Städten aufgeteilt: München und Augsburg dienten als Trainingsdaten, Regensburg als Validierungsdatensatz, während Nürnberg und Ingolstadt als unabhängige Teststädte verwendet wurden. Diese Aufteilung ermöglicht eine realistische Bewertung der Generalisierung auf radiometrisch unbekannte Daten. Gleichzeitig konnten auf Basis der Validierung in Regensburg Augmentierungsparameter angepasst werden, um das Modell robuster gegenüber variierenden Farb-, Beleuchtungs- und Aufnahmebedingungen zu machen.
Für den Anwendungsaspekt wurden alle Modelle der YOLOv11- und YOLOv12-Familie, mit Ausnahme der X-Modelle (aufgrund zu hoher Hardwareanforderungen), auf diesem Datensplit trainiert und validiert. Frühere Tests hatten gezeigt, dass bei radiometrisch unbekannten Daten bereits 30 Epochen ausreichen, um eine Konvergenz auf dem Validierungsdatensatz zu erreichen; entsprechend wurde die Trainingsdauer reduziert. Die besten Ergebnisse erzielten die Modelle YOLOv12m, YOLOv12l und YOLOv11l (in absteigender Reihenfolge). Da YOLOv12m die höchste mAP50–95 erreichte und in den übrigen Metriken mit YOLOv11l vergleichbar war, wurde dieses Modell für die Anwendung ausgewählt. Es hatte bereits im Architekturvergleich durch ein günstiges Verhältnis aus Genauigkeit und Effizienz überzeugt.
Im Anschluss wurde YOLOv12m mit unterschiedlichen Kombinationen von Augmentierungsparametern trainiert. Die besten Resultate erzielten die Standard-Augmentierungen von Ultralytics in Kombination mit einer zufälligen Rotation im Intervall von $\pm 30°$. Diese Rotation trug vermutlich dazu bei, die Sensitivität gegenüber Schattenrichtungen und Beleuchtung zu verringern, wodurch das Modell robuster gegenüber variablen Aufnahmebedingungen wurde.
Vergleicht man die kurze Trainingsdauer von 30 Epochen mit den 200 Epochen des Architekturvergleichs, zeigt sich, dass die geringere Epochenzahl hier ausreichend war, um auf radiometrisch neuen Daten zu konvergieren. Bei einer Erweiterung des Datensatzes um weitere Städte oder größere radiometrische Varianz müsste geprüft werden, ob die Epochenzahl erhöht werden muss. In den vorangegangenen Tests waren die Modelle mit der Radiometrie der Validierungsdaten bereits vertraut, weshalb dort die Metriken grundsätzlich höher ausfielen und durch Augmentierung weiter gesteigert werden konnten. Das Modell sieht so immer weitere Varianten, was die Leistung verbessert. Dies verdeutlicht, dass beim Training großer Datensätze genau auf die Balance zwischen Trainingsdauer, Datenvielfalt und Augmentierung geachtet werden muss, um eine robuste Generalisierung auf unbekannte Szenen zu erreichen.
Zur Optimierung der Vorhersageparameter wurde für jede Metrik eine Heatmap erstellt, in der Konfidenz- und NMS-Schwellen systematisch variiert wurden. Die besten F1-Werte

lagen im Bereich einer Konfidenzschwelle von 5–20% und einer NMS-Schwelle von 20–40%. Für die Precision wurden die höchsten Werte bei 50% Konfidenz und 10% NMS erzielt – also bei einer sehr strengen Filterung, die nur besonders sichere Vorhersagen zulässt und selbst geringe Überlappungen unterdrückt. Beim Recall zeigte sich das entgegengesetzte Verhalten: niedrige Konfidenz (5%) und hohe NMS-Schwelle (60%) führten zu den besten Ergebnissen. Ähnliche Muster traten bei der Lokalisierungsgenauigkeit (mAP50–95, mAP50) auf, die ebenfalls im Bereich niedriger Konfidenzen und mittlerer NMS-Werte ihre Maxima erreichte. Für den abschließenden Anwendungstest wurde daher eine Konfidenzschwelle von 5% und eine NMS-Schwelle von 40% gewählt, da diese Kombination den besten Kompromiss zwischen hoher Erkennungsrate und präziser Lokalisierung bot.
Die Ergebnisse der praktischen Anwendung bestätigen die Funktionsfähigkeit der implementierten Pipeline (Abb. 4.21–4.24). Die Höhenfilterung ab 5 m arbeitet zuverlässig, wie der Vergleich der gefilterten Referenzdaten in Abb. 4.21 und 4.23 zeigt. Ebenso reduziert die NMS-Filterung (Abb. 4.22) effektiv redundante Bounding Boxes und eliminiert doppelte Vorhersagen.
Die Tests in Nürnberg und Ingolstadt verdeutlichen, dass das Modell in beiden Fällen die besten Ergebnisse in Kategorie 1 („Continuous Urban Fabric > 80%“) erzielt. Diese stark versiegelten, innerstädtischen Flächen sind häufig von Straßen- und Einzelbäumen geprägt, die räumlich klar voneinander getrennt sind. Entsprechend hoch fallen die Ergebnisse aus: mAP50 = 0.838 und F1 = 0.843 in Ingolstadt sowie mAP50 = 0.800 und F1 = 0.807 in Nürnberg.
Die niedrigsten Werte wurden für Nürnberg in Kategorie 2 erreicht (mAP50 = 0.664, F1 = 0.686). Diese Kategorie umfasst dicht bebaute Wohngebiete mit 50–80% Versiegelung, in denen Bäume häufig entlang von Grundstücksgrenzen oder in Gruppen vorkommen. Das Beispiel in Abb. 4.25 zeigt, dass das Modell Schwierigkeiten hat, eng beieinanderstehende Bäume korrekt zu trennen. Für Ingolstadt traten die geringsten Werte in Kategorie 3 auf (mAP50 = 0.679, F1 = 0.733), was ebenfalls auf die hohe Baumdichte und Überlappung der Kronen zurückzuführen ist (Abb. 4.26).
Ein visueller Vergleich der RGB-Orthophotos legt zudem nahe, dass die geringere radiometrische Qualität und der niedrigere Kontrast der Nürnberger Orthophotos zur insgesamt etwas schwächeren Modellleistung beigetragen haben könnten. In Ingolstadt, wo die Daten kontrastreicher und klarer strukturiert sind, erzielte das Modell insgesamt bessere Ergebnisse.

5.4 Ausblick

Die Ergebnisse dieser Arbeit zeigen, dass die automatisierte Baumdetektion auf Basis aktueller Echtzeit-Objekterkennungsmodelle bereits robuste und übertragbare Ergebnisse liefern kann. Für das Training wurden lediglich Daten aus zwei Städten verwendet. Dennoch bestehen mehrere Ansatzpunkte, um die Methodik und den Datensatz künftig weiterzuentwickeln.
Eine weitere Verbesserung wäre es, auch ältere oder neuere Orthophotos in den Datensatz aufzunehmen. Diese könnten die bereits markierten Bäume beinhalten, wodurch die

Methode auch mit Daten aus verschiedenen Zeitpunkten verwendet werden könnte. Es wäre allerdings wichtig, zu prüfen, ob die Positionen und Größen der Bäume auf den neuen Bildern mit den vorhandenen Annotierungen übereinstimmen. Eine automatisierte Qualitätsprüfung der Labelgeometrien wäre hierfür ein vielversprechender Ansatz.
Methodisch könnte der Trainingsprozess durch die Verwendung kleinerer, nicht überlappender Eingangskacheln weiter optimiert werden. Insbesondere Kacheln in der Größe des Model-Inputs würden redundante Überlappungen vermeiden und die Datenbalance verbessern, da jedes Objekt nur einmal ins Training eingeht. Gleichzeitig ließe sich -bei gleichbleibender Flächenabdeckung - dadurch die Vielfalt der Trainingsbeispiele erhöhen, da kleinere Szenen mehr lokale Variationen innerhalb der Stadtstruktur abbilden können.
Darüber hinaus eröffnet die Detektion einzelner Bäume neue Anwendungsfelder. Auf Basis der vorhergesagten Baumpositionen könnten Analysen der Baumdichte pro Quadratmeter in Relation zum Versiegelungsgrad durchgeführt werden, um stadtplanerisch relevante Informationen abzuleiten. Solche Analysen könnten dazu beitragen, potenzielle Pflanzstandorte zu identifizieren und die Verteilung urbaner Bäume objektiv zu bewerten.
Ein noch offener technischer Aspekt betrifft die Vorhersage auf großflächigen Orthophoto-Kacheln. In der Praxis liegen viele städtische Datensätze in Einheiten von 1 km² vor. Bei der Verarbeitung solcher Szenen würden Objekte an den Kachelrändern teilweise abgeschnitten und dadurch fehlerhaft erkannt werden. Für eine flächendeckende Anwendung ist daher eine Strategie zur Kachelüberlappung oder Kantenfusion erforderlich, die benachbarte Vorhersagen effizient zusammenführt, ohne den Rechenaufwand wesentlich zu erhöhen. Eine solche Post-Processing-Methode könnte wesentlich zur nahtlosen Baumkartierung großer Stadtgebiete beitragen.
Schließlich wäre die Verwendung noch höher aufgelöster Orthophotos ein vielversprechender Ansatz, um die Abgrenzung einzelner Baumkronen zu verbessern. Eine feinere räumliche Auflösung könnte die Instanztrennung innerhalb dichter Baumgruppen erleichtern und die bisherigen Schwächen verringern.
Insgesamt bietet die entwickelte Methodik eine solide Grundlage, um die automatisierte Baumdetektion künftig in großmaßstäbige urbane Analysen und Planungsprozesse zu integrieren. Die vergleichsweise geringe Inferenzzeit des gewählten Modells ermöglicht eine zeit- und ressourceneffiziente Bestimmung von Baumstandorten auch auf großen Flächen und macht den Ansatz damit für operative Anwendungen, wie die regelmäßige Aktualisierung kommunaler Baumkataster, besonders attraktiv. Durch gezielte Erweiterungen des Datensatzes, methodische Verfeinerungen und eine stärkere inhaltliche Nutzung der Ergebnisse kann die Methode zu einem wertvollen Instrument in der datenbasierten Stadt- und Umweltplanung werden.

6 Zusammenfassung

Diese Arbeit befasst sich mit der automatisierten Erkennung von Bäumen in städtischen Gebieten anhand hochauflösender Orthophotos. Ziel war es, ein Verfahren zu entwickeln, das mithilfe moderner Objekterkennungsmodelle Bäume zuverlässig identifiziert und dabei schnell genug ist, um auch große Stadtgebiete effizient auszuwerten.

Als Datengrundlage dienten Orthophotos von den fünf größten bayerischen Städten aus dem Open-Data-Angebot des Bayerischen Landesamts für Digitalisierung, Breitband und Vermessung. Zur Sicherstellung einer ausgewogenen Repräsentation verschiedener urbaner Strukturen erfolgte ein gezieltes Szenen-Sampling auf Basis des europäischen Urban Atlas, dessen detaillierte Landnutzungsklassifikation in vier übergeordnete Hauptkategorien zusammengefasst wurde. Dadurch konnten typische urbane Landschaftsszenen, wie etwa dicht bebaute Innenstädte, Wohngebiete mittlerer Dichte oder Industrie- und Gewerbeflächen, systematisch in den Datensatz integriert werden. Die anschließende Annotation der Baumobjekte erfolgte teils manuell, teils halbautomatisch mithilfe bereits trainierter Modelle, wodurch eine abwechslungsreiche Trainingsbasis mit reduziertem Arbeitsaufwand geschaffen wurde.

Für den Modellvergleich wurden die aktuellen Echtzeit-Architekturen YOLOv11, YOLOv12 und RT-DETR getestet. Es zeigte sich, dass größere Modelle zwar präziser sind, jedoch mehr Rechenzeit benötigen. Unter allen Varianten erzielte YOLOv12 insgesamt die besten Ergebnisse, während YOLOv11 bei kürzeren Trainings- und Validierungszeiten effizienter arbeitete. In der praktischen Anwendung überzeugte besonders das Modell YOLOv12m, das bei halbierter Inferenzzeit nahezu die gleiche Erkennungsleistung wie die größten Modelle erreichte und damit den besten Kompromiss zwischen Genauigkeit und Geschwindigkeit bot. Eine moderate Filterung unvollständiger Trainingsobjekte an Randbereichen führte zusätzlich zu einer spürbaren Leistungssteigerung.

In der praktischen Anwendung wurde das Modell YOLOv12m auf zwei Städten trainiert, auf einer Stadt validiert und auf zwei unbekannten Städten angewendet. Zufällige Bildrotationen im Training erwiesen sich als besonders hilfreich. Sie erhöhten die Robustheit gegenüber unterschiedlichen Schatten- und Beleuchtungssituationen. Bei der Anwendung des Modells auf den unbekannten Städten zeigte es eine hohe Generalisierungsfähigkeit. Es erzielte in stark versiegelten innerstädtischen Bereichen entlang von Straßen sehr hohe Erkennungsraten. Bäume in dichten Gruppen wurden dagegen nur unzureichend erkannt, und das Modell lieferte dort deutlich schwächere Ergebnisse.

Die entwickelte Methodik ermöglicht eine schnelle und präzise Erfassung von Bäumen über große Stadtflächen hinweg. Durch die Erweiterung des Datensatzes auf weitere Städte und Aufnahmejahre, den Einsatz kleinerer, nicht überlappender Kacheln und die Verwendung

hochaufgelöster Orthophotos kann die Genauigkeit künftig weiter verbessert werden. Aufgrund der kurzen Inferenzzeit eignet sich das Verfahren besonders für traditionell sehr aufwendige Anwendungen, wie beispielsweise die Aktualisierung kommunaler Baumkataster oder die Ermittlung potenzieller Pflanzstandorte. Damit leistet diese Arbeit einen Beitrag zur datenbasierten Stadt- und Umweltplanung und zeigt, wie KI-gestützte Bildanalyse den Umgang mit urbanem Grün deutlich vereinfachen kann.

Literaturverzeichnis

[Beloiu u. a. 2023] Beloiu, Mirela ; Heinzmann, Lucca ; Rehush, Nataliia ; Gessler, Arthur ; Griess, Verena C.: Individual Tree-Crown Detection and Species Identification in Heterogeneous Forests Using Aerial RGB Imagery and Deep Learning. In: *Remote Sensing* 15 (2023), Nr. 5, S. 1463. – URL https://doi.org/10.3390/rs15051463

[Bochkovskiy u. a. 2020] Bochkovskiy, Alexey ; Wang, Chien-Yao ; Liao, Hong-Yuan M.: *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020

[Carion u. a. 2020] Carion, Nicolas ; Massa, Francisco ; Synnaeve, Gabriel ; Usunier, Nicolas ; Kirillov, Alexander ; Zagoruyko, Sergey: End-to-End Object Detection with Transformers. In: *CoRR* abs/2005.12872 (2020). – URL https://arxiv.org/abs/2005.12872

[da Costa-Luis 2019] Costa-Luis, Casper O. da: tqdm: A Fast, Extensible Progress Meter for Python and CLI. In: *Journal of Open Source Software* 4 (2019), Nr. 37, S. 1277. – URL https://www.researchgate.net/publication/333181652_tqdm_A_Fast_Extensible_Progress_Meter_for_Python_and_CLI

[EEA 2018] EEA: *Urban Atlas 2012-2018 Mapping Guide*. Version 6.3. Copenhagen: Copernicus Land Monitoring Service (Veranst.), 2018. – URL https://land.copernicus.eu/en/technical-library/urban_atlas_2012_2018_mapping_guide. – Abgerufen am 17.09.2025

[EU-Copernicus Programme 2025] EU-Copernicus Programme: *Copernicus: Access to Data*. 2025. – URL https://www.copernicus.eu/en/access-data. – Abgerufen am 16.09.2025

[Garcia-Garcia u. a. 2017] Garcia-Garcia, Alberto ; Orts-Escolano, Sergio ; Oprea, Sergiu ; Villena-Martinez, Victor ; Garcia-Rodriguez, Jose: *A Review on Deep Learning Techniques Applied to Semantic Segmentation*. 2017. – URL https://arxiv.org/abs/1704.06857

[Gillies u. a. 2019] Gillies, Sean u. a.: *Rasterio: Access to geospatial raster data*. 2019. – URL https://github.com/rasterio/rasterio

[Girshick 2015] Girshick, Ross: Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015

[Harris u. a. 2020] Harris, Charles R. ; Millman, K. J. ; Walt, Stéfan J. van der u. a.: Array programming with NumPy. In: *Nature* 585 (2020), S. 357–362. – URL https://numpy.org

[He u. a. 2015] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: *Deep Residual Learning for Image Recognition*. 2015. – URL https://arxiv.org/abs/1512.03385

[He u. a. 2025] He, Lh. ; Zhou, Yz. ; Liu, L. u. a.: Research on object detection and recognition in remote sensing images based on YOLOv11. In: *Scientific Reports* 15 (2025), S. 14032. – URL https://doi.org/10.1038/s41598-025-96314-x

[Hunter 2007] Hunter, John D.: Matplotlib: A 2D Graphics Environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95. – URL https://matplotlib.org

[Jegham u. a. 2025] Jegham, Nidhal ; Koh, Chan Y. ; Abdelatti, Marwan ; Hendawi, Abdeltawab: *YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions*. 2025. – URL https://arxiv.org/abs/2411.00201

[Jocher und Qiu 2024] Jocher, Glenn ; Qiu, Jing: *Ultralytics YOLO11*. 2024. – URL https://github.com/ultralytics/ultralytics

[Jocher 2025] Jocher, Glenn und M.: *Ultralytics: AI Vision and Machine Learning Toolkit*. 2025. – URL https://github.com/ultralytics/ultralytics

[Jordahl u. a. 2020] Jordahl, Kelsey ; Bossche, Joris V. den ; Fleischmann, Martin ; Wasserman, Jacob ; McBride, James ; Gerard, Jeffrey ; Tratner, Jeff ; Perry, Matthew ; Badaracco, Adrian G. ; Farmer, Carson ; Hjelle, Geir A. ; Snow, Alan D. ; Cochran, Micah ; Gillies, Sean ; Culbertson, Lucas ; Bartos, Matt ; Eubank, Nick ; maxalbert ; Bilogur, Aleksey ; Rey, Sergio ; Ren, Christopher ; Arribas-Bel, Dani ; Wasser, Leah ; Wolf, Levi J. ; Journois, Martin ; Wilson, Joshua ; Greenhall, Adam ; Holdgraf, Chris ; Filipe ; Leblanc, François: *geopandas/geopandas: v0.8.1*. 2020. – URL https://doi.org/10.5281/zenodo.3946761

[Khanam und Hussain 2024] Khanam, Rahima ; Hussain, Muhammad: YOLOv11: An Overview of the Key Architectural Enhancements. In: *arXiv preprint arXiv:2410.17725* (2024)

[Krizhevsky u. a. 2012] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems* Bd. 25, 2012

[LDBV 2025a] LDBV: *Digitales Geländemodell (DGM1 / DGM5)*. 2025. – URL https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=dgm1. – Abgerufen am 16.09.2025

[LDBV 2025b] LDBV: *Digitales Oberflächenmodell (DOM20)*. 2025. – URL https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=dom20. – Abgerufen am 16.09.2025

[LDBV 2025c] LDBV: *OpenData: Verwaltung*. 2025. – URL https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=verwaltung. – Abgerufen am 17.09.2025

[LDBV 2025d] LDBV: *OpenGeoData Bayern*. 2025. – URL `https://geodaten.bayern.de/opengeodata/`. – Abgerufen am 16.09.2025

[LDBV 2025e] LDBV: *Orthophotos – Produktbeschreibung*. 2025. – URL `https://www.ldbv.bayern.de/produkte/luftbilder/orthophotos.html`. – Abgerufen am 16.09.2025

[Lecun u. a. 1998] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[García de León u. a. 2025] León, Andrea S. García de ; Leichtle, Tobias ; Ullmann, Tobias ; Castañeda-Gómez, Antonio ; Rötzer, Thomas ; Karges, Nils ; Martin, Klaus ; Taubenböck, Hannes: The Relation of Land Surface Temperature and Trees across Different Urban Land Use Classes based on Remote Sensing. In: *Joint Urban Remote Sensing Event (JURSE)*. Munich, Germany, 2025

[Li u. a. 2022] Li, Feng ; Zhang, Hao ; Liu, Shilong ; Guo, Jian ; Ni, Lionel M. ; Zhang, Lei: DN-DETR: Accelerate DETR Training by Introducing Query Denoising. In: *CVPR*, 2022

[Lin u. a. 2025] Lin, Tsung-Yi u. a.: *pycocotools: Python API for COCO dataset and evaluation*. 2025. – URL `https://github.com/ppwwyy/cocoapi`

[Lin u. a. 2017] Lin, Tsung-Yi ; Dollar, Piotr ; Girshick, Ross ; He, Kaiming ; Hariharan, Bharath ; Belongie, Serge: Feature Pyramid Networks for Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017

[Lin u. a. 2018] Lin, Tsung-Yi ; Goyal, Priya ; Girshick, Ross ; He, Kaiming ; Dollár, Piotr: *Focal Loss for Dense Object Detection*. 2018. – URL `https://arxiv.org/abs/1708.02002`

[Lin u. a. 2014] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Dollár, Piotr ; Zitnick, C L.: Microsoft COCO: Common Objects in Context. In: *ECCV*, 2014, S. 740–755

[Liu u. a. 2022] Liu, Shilong ; Li, Feng ; Zhang, Hao ; Yang, Xiao ; Qi, Xianbiao ; Su, Hang ; Zhu, Jun ; Zhang, Lei: DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR. In: *International Conference on Learning Representations*, 2022

[Loshchilov und Hutter 2019] Loshchilov, Ilya ; Hutter, Frank: *Decoupled Weight Decay Regularization*. 2019. – URL `https://arxiv.org/abs/1711.05101`

[McKinney 2010] McKinney, Wes: Data Structures for Statistical Computing in Python. In: *Proceedings of the 9th Python in Science Conference*, URL `https://pandas.pydata.org`, 2010, S. 51–56

[Nair und Hinton 2010] Nair, Vinod ; Hinton, Geoffrey E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, S. 807–814

[OpenAI 2025] OpenAI: *ChatGPT (Modelle GPT-4.5 und GPT-5)*. `https://chat.openai.com`. 2025. – Abgerufen am 01.11.2025

[Pauleit u. a. 2019] Pauleit, Stephan ; Zölch, Teresa ; Reischl, Astrid ; Rahman, Mohammad ; Rötzer, Thomas: Cool durch grüne Infrastruktur: Die Potenziale des Stadtgrüns zur städtischen Klimawandelanpassung. In: *Transforming Cities* (2019), Nr. 3, S. 60–65

[QGIS Entwicklungsteam 2024] QGIS Entwicklungsteam: *QGIS Geographic Information System. Open Source Geospatial Foundation Project*. `https://qgis.org`. 2024. – Version 3.34, Abgerufen am 01.11.2025

[Redmon u. a. 2016] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, URL `https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf`, 2016, S. 779–788

[Ren u. a. 2017] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (2017), Nr. 6, S. 1137–1149

[Ribeiro u. a. 2021] Ribeiro, Andreza P. ; Bollmann, Harry A. ; Oliveira, Anderson de ; Rakauskas, Felipe ; Cortese, Tatiana Tucunduva P. ; Rodrigues, Maria Santiellas C. ; Quaresma, Cristiano C. ; Ferreira, Maurício L.: The Role of Tree Landscape to Reduce Effects of Urban Heat Islands: A Study in Two Brazilian Cities. In: *Trees* 35 (2021), Nr. 6, S. 1869–1889

[Rumelhart u. a. 1986] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), S. 533–536. – URL `https://api.semanticscholar.org/CorpusID:205001834`

[StaBuL 2024] StaBuL: *Tabelle 12411-01-01-4: Bevölkerung der Gemeinden*. 2024. – URL `https://www.regionalstatistik.de/genesis/online?operation=table&code=12411-01-01-4&bypass=true&levelindex=1&levelid=1760339473828`. – Abgerufen am 13.10.2025

[Stark u. a. 2025] Stark, Thomas ; Pardo, Julia ; Wurm, Michael ; Leichtle, Tobias ; Martin, Klaus ; Taubenböck, Hannes: Urban Tree Detection: Comparing YOLOv8 and DeepForest for Accurate Single-Tree Identification from Aerial Imagery. In: *2025 Joint Urban Remote Sensing Event (JURSE)* Bd. CFP25RSD-ART, 2025, S. 1–4

[Tian u. a. 2025] Tian, Yunjie ; Ye, Qixiang ; Doermann, David: YOLOv12: Attention-Centric Real-Time Object Detectors. In: *arXiv preprint arXiv:2502.12524* (2025)

[Ultralytics 2025] Ultralytics: *Hyperparameter Tuning Guide*. https://docs.ultralytics.com/de/guides/hyperparameter-tuning/. 2025. – Abgerufen am 08.11.2025

[Ultralytics 2025] Ultralytics: *Ultralytics YOLO Documentation – Data Augmentation and Training Parameters*. 2025. – URL https://docs.ultralytics.com/modes/train/. – Abgerufen am 13.10.2025

[Velasquez-Camacho u. a. 2023] Velasquez-Camacho, Luisa ; Etxegarai, Maddi ; Miguel, Sergio de: Implementing Deep Learning algorithms for urban tree detection and geolocation with high-resolution aerial, satellite, and ground-level images. In: *Computers, Environment and Urban Systems* 105 (2023), S. 102025. – URL https://doi.org/10.1016/j.compenvurbsys.2023.102025

[Wang u. a. 2019] Wang, Chien-Yao ; Liao, Hong-Yuan M. ; Yeh, I-Hau ; Wu, Yueh-Hua ; Chen, Ping-Yang ; Hsieh, Jun-Wei: *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. – URL https://arxiv.org/abs/1911.11929

[Waskom u. a. 2021] Waskom, Michael L. u. a.: *Seaborn: Statistical Data Visualization*. 2021. – URL https://seaborn.pydata.org

[Zhao u. a. 2024] Zhao, Yian ; Lv, Wenyu ; Xu, Shangliang ; Wei, Jinman ; Wang, Guanzhong ; Dang, Qingqing ; Liu, Yi ; Chen, Jie: DETRs Beat YOLOs on Real-time Object Detection. In: *arXiv preprint arXiv:2304.08069* (2024)

[Zhao u. a. 2019] Zhao, Zhong-Qiu ; Zheng, Peng ; Xu, Shou-Tao ; Wu, Xindong: Object Detection With Deep Learning: A Review. In: *IEEE Transactions on Neural Networks and Learning Systems* 30 (2019), Nr. 11, S. 3212–3232

[Zhou u. a. 2019] Zhou, Xingyi ; Wang, Dequan ; Krähenbühl, Philipp: *Objects as Points*. 2019. – URL https://arxiv.org/abs/1904.07850

[Zhu u. a. 2020] Zhu, Xizhou ; Su, Weijie ; Lu, Lewei ; Li, Bin ; Wang, Xiaogang ; Dai, Jifeng: Deformable DETR: Deformable Transformers for End-to-End Object Detection. In: *International Conference on Learning Representations*, 2020

[Zou u. a. 2023] Zou, Zhengxia ; Chen, Keyan ; Shi, Zhenwei ; Guo, Yuhong ; Ye, Jieping: *Object Detection in 20 Years: A Survey*. 2023. – URL https://arxiv.org/abs/1905.05055