**Berliner Hochschule für Technik**

Fachbereich III: Geoinformation

Schwerpunkt: Geoinformatik und Geomedien

Masterarbeit

# Automatisierte Erkennung und Objektsegmentierung in Karten


Sven Endler

Matrikelnummer: 924786

Abgabe: 17.11.2025

Betreuer: Prof. Dr. Florian Hruby

# I. Vorwort

In Zeiten, in denen KI-Modelle immer mehr Aufmerksamkeit erhalten und Diskussionen über ihre Anwendungen kaum noch aus den Nachrichten wegzudenken sind, habe ich mich im Laufe meines Studiums immer mehr mit dem Thema beschäftigt, inwieweit dieser Trend auch in der Geoinformation und speziell in der Kartografie Nutzen bringen kann. In Zusammenarbeit mit meinem Dozenten Prof. Dr. Florian Hruby wurde schließlich ein Thema gefunden, das genau diesem Trend entspricht. Daher möchte ich mich für die gute Zusammenarbeit und für die Begleitung der vielen Jahre meines Studiums seit dem 2. Bachelorsemester bedanken. Darüber hinaus bedanke ich mich beim Geomedienlabor und insbesondere bei Herrn Martin Vigerske für die Bereitstellung der topografischen Karten des ersten Datensatzes.

# II. Abstract

Diese Masterarbeit beschäftigt sich mit der automatisierten Erkennung und Segmentierung von Symbolen und Piktogrammen in Karten. Dafür wurde ein System entwickelt, dass verschiedene Objekterkennungsverfahren nutzt und diese mit einem geeigneten Segmentierungsmodell kombiniert. Die Modelle YOLOv5, YOLOv8, Faster R-CNN, SSD und RetinaNet wurden vergleichend eingesetzt und auf zwei unterschiedlichen Datensätzen trainiert. Für die Segmentierung wird SAM verwendet. Der Datensatz für die einfachen Symbole erzielte insgesamt einen mAP50 Wert von 0,85 sowie eine mAP50-95 von 0,51. Der zweite Datensatz für die Piktogramme konnte mit einer mAP50 von 0,56 und einer mAP50-95 von 0,30 eine schwächere Erkennungsleistung erzielen. Die Kombination aus automatischer Erkennung und Segmentierung bietet neue Möglichkeiten für die Kartografie in Hinblick auf die Analyse und Erstellung von Karten.

# III. Inhaltsverzeichnis

# IV. Abbildungsverzeichnis

# V. Tabellenverzeichnis

# VI. Formelverzeichnis

# VII. Abkürzungsverzeichnis

AHD ..... Average Hausdorff Distance
ANN ..... Artifical Neural Network
AP ..... Average Precision
aRNN ..... Artifical Rekurrent Neural Network
bRNN ..... Biological Rekurrent Neural Network
Caffe ..... Convolutional Architecture for Fast Feature Embedding
CBAM ..... Convolutional Block Attention Module
CNN ..... Convolutional Neural Network
COCO ..... Common Objects in Context
DFL ..... Distribution Focal Loss
DPM ..... Deformable Part-based Model
DSC ..... Dice Similarity Coefficient
FCN ..... Fully Convolutional Network
FFCNN ..... Feed Forward Convolutional Neural Network
FN ..... False Negative
FP ..... False Positive
FPN ..... Feature Pyramid Network
GIS ..... Geoinformationssystem
HD ..... Hausdorff Distance
HOG ..... Histogram of Oriented Gradients
IoU ..... Intersection over Union
Kap ..... Cohens Kappa
mAP ..... mean Average Precision
mAP@0.50 ..... mean Average Precision mit Schwellenwert 0.5
mAP@0.5-0.95 ..... mean Average Precision mit Schwellenwert 0.5 bis 0,95
NMS ..... Non-Maximum Suppression
P ..... Precision
PA ..... Pixel Accuracy
PASCAL ..... Pattern Analysis, Statistical Modelling and Computational Learning
POI ..... Points of interest

R ..... Recall

R-CNN ..... Region-based Convolutional Network

RNN ..... Rekurrent Neural Network

SA ..... Segment Anything

SAM ..... Segment Anything Model

SGD ..... Stochastic Gradient Descent

SIFT ..... Scale-Invariant Feature Transform

SSD ..... Single Shot Multibox Detector

SVM ..... Support Vector Machines

TN ..... True Negative

TP ..... True Positive

ViT ..... Vision Transformer

VOC ..... Visual Object Classes

YOLO ..... You Only Look Once

# 1 Einleitung

## 1.1 Motivation

Die Erkennung und Extraktion von Objekten aus Karten ist ein zunehmend relevantes Forschungsfeld innerhalb der Geoinformation. Mit dem Fortschritt leistungsfähiger Deep-Learning-Algorithmen, die in zahlreichen Anwendungsbereichen erfolgreich eingesetzt werden, eröffnen sich nun auch in der Kartografie und Fernerkundung neue Möglichkeiten. Karten- oder Luftbildinhalte können automatisch analysiert und relevante Objekte identifiziert werden.

Karten stellen geografische Informationen verständlich dar, doch die Interpretation und Digitalisierung ist oft zeitaufwendig, fehleranfällig und nicht skalierbar. Des Weiteren nimmt die Verfügbarkeit digitaler Karten und Kartenarchive sowie Satellitenbilder immer weiter zu. Die Automatisierung der Objekterkennung bietet daher die Möglichkeit, diese Daten effizienter zu nutzen. Beispielsweise können unterschiedliche Objekte wie Straßen, Gebäude oder Flüsse extrahiert und für andere Anwendungen wie etwa GIS oder neue Karten nutzbar gemacht werden. Vor allem kleine und vielfältige Objekte wie Symbole oder Piktogramme stellen eine besondere Herausforderung dar. Im Folgenden wird die zentrale Problemstellung dieser Masterarbeit vorgestellt.

## 1.2 Problemstellung

Trotz großer Fortschritte im Bereich der künstlichen Intelligenz und der Verfügbarkeit von großen Datenmengen stellt die automatische Erkennung und Segmentierung von Objekten auf Karten eine Herausforderung dar. Karten haben unterschiedliche Themen, Maßstäbe, Symboliken und Qualitäten, wodurch die Entwicklung universeller Methoden erschwert wird. Ebenso können Objekte wie Straßen, Beschriftungen oder Symbole je nach Kartenstil unterschiedlich gestaltet sein und sich teilweise überlagern. Dadurch ist es schwer eine präzise automatische Analyse durchzuführen.

Viele der bisherigen Ansätze konzentrieren sich meist auf die Erkennung und Segementierung von Flächenobjekten. Die Forschung zu kleineren Objekten wie Symbolen oder komplexen Piktogrammen steht noch am Anfang und bietet viel Potenzial

für zukünftige Entwicklungen in der Kartografie. Diese spielen allerdings oft eine entscheidende Rolle in der Interpretation und Analyse von Karten. Sie vermitteln Informationen kompakt und intuitiv, wodurch eine schnelle Einordnung in den Kontext ermöglicht wird. Ihre Vielfalt in Form, Farbe und Stil erschwert jedoch eine automatisierte Erkennung. Zudem werden sie meist aufwendig und individuell erstellt. Deren Erkennung ermöglicht es unter anderem, Karten thematisch einzuordnen und dadurch die Auffindbarkeit von bestimmten Karten nach Thema zu verbessern. Darüber hinaus ermöglicht die Segmentierung Symbole und Piktogramme in weiteren Anwendungen wiederzuverwenden.

Das Paper von CAO et al. (2025) stellte einen Ansatz vor, bei dem ein Objekterkennungsverfahren mit einem Segementierungsverfahren kombiniert wird, um das bestmögliche Ergebnis zu erzielen. Dabei wird sich auf Piktogramme in touristischen Karten konzentriert. Darauf aufbauend untersucht diese Arbeit verschiedene Objekterkennungsverfahren und wendet im Anschluss ein geeignetes Segmentierungsverfahren an. Die Ergebnisse werden anschließend vergleichend ausgewertet. Als Datengrundlage dienen Symbole aus topografischen Karten und Piktogramme aus historischen Themenkarten. Das Ziel ist es, herauszufinden, welches Verfahren für diesen Ansatz die besten Ergebnisse liefert und inwieweit sich die Verfahren hinsichtlich Leistungsfähigkeit und Genauigkeit unterscheiden. Es wird also untersucht, wie gut Symbole und Piktogramme in diesen Karten automatisiert erkannt und segmentiert werden können. So trägt die Arbeit zur Weiterentwicklung automatisierter Verfahren in der Kartografie bei.

## 1.3 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich inhaltlich in 8 Kapitel. Zuerst erfolgt die Einleitung mit Motivation, Problemstellung und Aufbau der Arbeit. Danach werden theoretische Grundlagen vorgestellt, die für das Verständnis der Arbeit benötigt werden. Hier werden grundlegend die Funktionsweisen und Architekturen bekannter neuronaler Netze erläutert. Anschließend werden die wichtigsten Objekterkennungsalgorithmen und deren Bewertungsmetriken erklärt. Ebenso wird im nächsten Unterkapitel auf die Segmentierung eingegangen. Im 3. Kapitel werden aktuelle Forschungen zur Erkennung

und Extraktion in der Kartografie sowie deren Herausforderungen aufgezeigt und die Masterarbeit in den Forschungsstand eingeordnet. Im nächsten Kapitel beginnt der praktische Teil der Arbeit. Zunächst werden die Methodik und technische Umsetzung des Ansatzes vorgestellt. Anschließend wird die Implementierung des Systems im Detail erläutert. Im darauffolgenden Kapitel werden die verwendeten Verfahren anhand bestimmter Bewertungsmetriken analysiert und ausgewertet. Danach werden die Ergebnisse im 7. Kapitel diskutiert, worauf ein Fazit und Ausblick zur gesamten Arbeit erfolgt.

# 2 Theoretische Grundlagen

Im folgenden Kapitel werden grundlegende theoretische Prinzipien und Methoden erläutert, die für das Verständnis der Arbeit relevant sind. Zuerst werden die wichtigsten neuronalen Netze erklärt. Darunter Künstliche neuronale Netze[1] (ANN), faltungsneuronale Netze[2] (CNN) und rekurrente neuronale Netze[3] (RNN). Da vor allem CNNs die Basis für die später verwendeten Algorithmen bilden, wird auf diese genauer eingegangen. Anschließend werden bedeutende Objekterkennungsalgorithmen und deren Bewertungsmethoden vorgestellt. Im Fokus stehen hierbei R-CNN, YOLO, SSD und RetinaNet. Zuletzt wird auf das verwendete Segmentierungsverfahren Segment Anything Model (SAM) und dessen Architektur eingegangen.

## 2.1 Neuronale Netze

Neuronale Netze bilden die Grundlage für verschiedenste KI-Methoden und Anwendungen. In diesem Unterkapitel werden die bedeutendsten Typen vorgestellt, ihre Funktionsweisen erläutert und ihr Nutzen in unterschiedlichen Einsatzgebieten aufgezeigt.

### 2.1.1 Künstliche neuronale Netze (ANN)

Künstliche neuronale Netze sind modellbasierte Systeme, welche auf dem Nervensystem des menschlichen Gehirns basieren. Sie werden für maschinelles Lernen oder künstliche Intelligenz eingesetzt (vgl. O'SHEA & NASH, 2015:1; WUTTKE, 2024). Mithilfe von ANNs lassen sich Probleme bewältigen, die für den Menschen unlösbar erscheinen. Häufige Anwendungsprobleme stammen aus den Bereichen Statistik, Informatik und Wirtschaft. Sie spielen unter anderem eine zentrale Rolle in Frühwarnsystemen, der Bilderkennung, Wettervorhersagen sowie medizinischen Analysen (vgl. WUTTKE, 2024). Eine große Anzahl an Rechnerknoten bilden den Hauptbestandteil von ANNs. Sie können mit den Neuronen im menschlichen Gehirn verglichen werden. Durch verteiltes Arbeiten lernen sie gemeinsam aus den Eingaben und können somit die endgültige Aussage optimieren.

[1] Artificial Neural Network (ANN)
[2] Convolutional Neural Network (CNN)
[3] Rekurrent Neural Network (RNN)

In der Regel kommen mehrdimensionale Vektoren zum Einsatz, welche in die Eingabeschicht hineingeladen und anschließend auf die verborgene Schicht verteilt werden. Die verborgenen Schichten verarbeiten die Ausgabe aus der vorherigen Schicht weiter und bewerten, inwiefern strukturelle Veränderungen innerhalb ihrer eigenen Architektur die finale Ausgabe verschlechtern oder verbessern. Im wissenschaftlichen Kontext wird dieser Vorgang als Lernprozess bezeichnet. Werden mehrere verborgene Schichten übereinandergestapelt nennt man den Prozess Deep Learning (vgl. O'SHEA & NASH, 2015:1). Dabei lernt das Netzwerk eine mehrstufige Hierarchie, die es ermöglicht, komplexe Konzepte aus der Erfahrung zu erfassen, ohne dass ein Mensch diese Vorgabe machen muss (vgl. GOODFELLOW et al., 2016) Am Ende der Verarbeitungskette steht die Ausgabeschicht, welche das endgültige Ergebnis des Netzwerks liefert (vgl. O'SHEA & NASH, 2015:2).

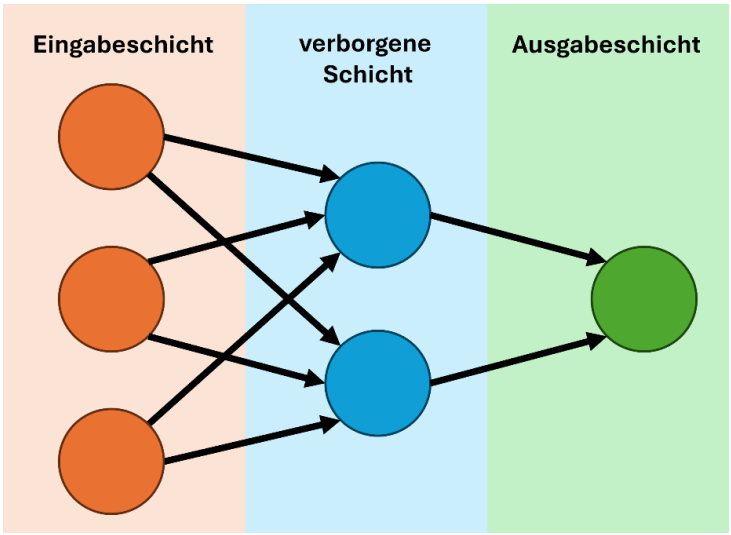



Abbildung 1: schematische Darstellung eines künstlichen neuronalen Netzes (in Anlehnung an O'SHEA & NASH, 2015 sowie WUTTKE, 2024)

### 2.1.2 Faltungsneuronale Netze (CNN)

Faltungsneuronale Netze ähneln den herkömmlichen ANNs, da auch sie aus Neuronen bestehen, welche sich selbst optimieren (vgl. O'SHEA & NASH, 2015:2). CNNs können relevante Merkmale direkt aus den Eingabedaten lernen, sodass eine manuelle Merkmalsextraktion entfällt. Charakteristisch für CNNs ist ihre höhere Effizienz im Hinblick auf Speicherbedarf und Modellkomplexität. Typische Anwendungsbeispiele sind unter anderem Bildverarbeitung, Bildklassifizierung, Spracherkennung oder

Textklassifizierung (vgl. UPRETI, 2022). Allerdings stellt die Verarbeitung von Bilddaten aufgrund ihrer Komplexität und des damit verbundenen Rechenaufwands eine der größten Herausforderungen für ANNs dar. Der wesentliche Unterschied liegt darin, dass CNNs speziell für die Mustererkennung in Bildern ausgelegt sind. Bildspezifische Merkmale können in die Netzwerkarchitektur integriert werden, wodurch das Netzwerk sehr gut für bildorientierte Aufgaben funktioniert (vgl. O'SHEA & NASH, 2015:2).

## Modellarchitektur

Auch CNNs bestehen aus einer Eingabeschicht, mehreren verborgenen Schichten und einer Ausgabeschicht (Siehe Abbildung 1). Der größte Unterschied in der Architektur besteht darin, dass die Schichten in drei Dimensionen organisiert sind. Die Eingabe umfasst zum einen die räumlichen Dimensionen Höhe und Breite, zum anderen die Tiefe. Im Gegensatz zu ANNs verbinden sich die Neuronen von CNNs nur mit einem kleinen, lokalen Bereich der vorhergehenden Schicht. Bei der Verarbeitung werden die Dimensionen Höhe und Breite, durch die Faltungs-[4] und Pooling-Schichten[5] schrittweise reduziert, wodurch die enthaltenden Informationen verdichtet werden. Danach folgen die vollständig verbundenen Schichten, welche die extrahierten Merkmale zur Klassifikation nutzen. Die genaue Funktionsweise der unterschiedlichen Schichttypen wird im weiteren Verlauf erklärt. Die Tiefe entspricht dabei der Anzahl der Klassen. Zuletzt wird ein Score ausgegeben, der für jede Klasse die Wahrscheinlichkeit darstellt (vgl. O'SHEA & NASH, 2015:2).

[4] Faltungsschichten (Convolutional Layer) dienen zur Erkennung von Mustern oder Merkmalen im Eingabebild (vgl. EITCA, 2023)

[5] Pooling Schichten (Pooling Layer) verkleinern die Merkmalskarten, um den Rechenaufwand zu reduzieren (vgl. EITCA, 2023)

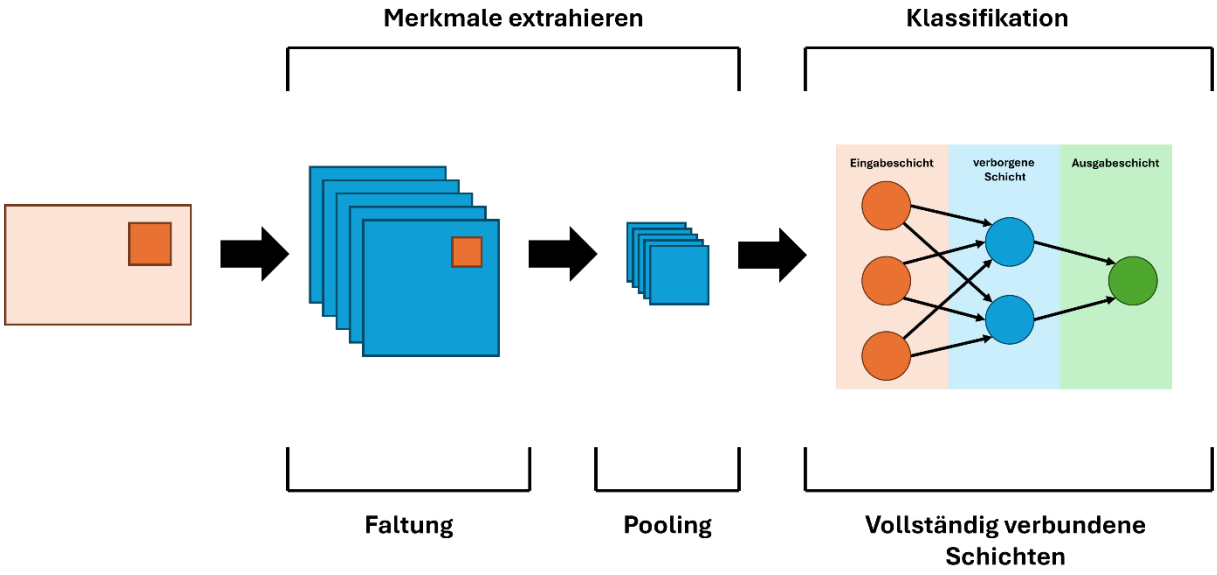



Abbildung 2: schematische Darstellung eines faltungsneuronalen Netzes (in Anlehnung an ISMAIL et al., 2023)

## Faltungsschichten

Die namensgebenden Faltungsschichten sind entscheidend für die Funktionsweise von CNNs. Sie bestehen aus lernbaren Filtern und sind meistens räumlich klein, aber reichen durch die gesamte Tiefe der Eingabedaten. Sobald diese eine Faltungsschicht durchlaufen, wird jeder Filter über die räumlichen Dimensionen bewegt und erzeugt dabei eine Merkmalskarte. Es wird das Skalarprodukt für jeden Wert im Filter berechnet. Dadurch kann das Netzwerk lernen Filter zu entwickeln, die ein bestimmtes Merkmal an einer bestimmten Position in der Eingabe erkennen. Jeder Filter besitzt somit eine Merkmalskarte, die dann in der Tiefendimension übereinandergestapelt werden.

Die Parameter Tiefe, Schrittweite[6] und Padding können die Struktur und Komplexität des Modells erheblich beeinflussen:

Die Tiefe eines faltungsneuronalen Netzes wird durch die Anzahl der Neuronen beeinflusst. Je weniger Neuronen verwendet werden, desto kleiner wird das Netzwerk. Dadurch verringert sich die benötigte Rechenleistung, allerdings kann das Netzwerk weniger komplexe Muster erkennen.

[6] (Stride)

Die Schrittweite definiert, wie weit sich die Filter bei der Faltung über das Eingabebild bewegen. Eine kleine Schrittweite führt beispielsweise zu starken Überlappungen der rezeptiven Felder[7], dies resultiert in einer höheren Auflösung und hohem Rechenaufwand. Eine größere Schrittweite hingegen verringert die Überlappung, reduziert die Anzahl der Ausgaben und senkt den Rechenaufwand (vgl. O'SHEA & NASH, 2015:2). Die Schrittweite wird horizontal und vertikal angewendet, dabei wird beispielsweise jede zweite oder dritte Position übersprungen (vgl. UPRETI, 2022).

Padding bezeichnet das Hinzufügen von Rändern um das Eingabebild. Wird kein Padding angewendet, kann es zu Informationsverlusten kommen, da die Randbereiche bei der Faltung nicht vollständig berücksichtigt werden (vgl. UPRETI, 2022). Eine häufig verwendete Variante ist das Zero-Padding. Hier werden die Ränder des Eingabebildes mit Nullen aufgefüllt. Insgesamt wird Padding verwendet, um die räumlichen Dimensionen der Ausgabedaten zu kontrollieren (vgl. O'SHEA & NASH, 2015:2).

#### Pooling-Schichten

Pooling-Schichten werden genutzt, um die Größe der Daten schrittweise zu verkleinern. Dadurch werden weniger Parameter benötigt und die rechnerische Komplexität des Modells verringert. Typisch sind 2x2 Filter mit einer Schrittweite von 2, diese werden auch als Max-Pooling bezeichnet. Dabei wird die räumliche Größe der Merkmalskarten um ein Viertel reduziert, während die Tiefe erhalten bleibt (vgl. O'SHEA & NASH, 2015:2).

#### Vollständig verbundene Schichten

In vollständig verbundenen Schichten ist jedes Neuron mit sämtlichen Neuronen der vorherigen sowie der nächsten Schicht verbunden, genau wie in ANNs (Siehe Abbildung 1) (vgl. O'SHEA & NASH, 2015:2).

### 2.1.3 Rekurrente neuronale Netze (RNN)

Ein rekurrentes neuronales Netzwerk besteht aus Neuronen, die miteinander über Rückmeldesignale verbunden sind (vgl. GROSSBERG, 2013). FAUSETT (1994:12, 372) beschreibt ein solches Netzwerk als einen geschlossenen Kreislauf, in dem eine Einheit

[7] Ein rezeptives Feld ist der Bereich des Eingabebildes, den ein Neuron der Faltungsschicht auswerten kann.

ihre eigene Ausgabe wieder als Eingabe erhält, und betont, dass sie darauf ausgelegt sind, sequenziell oder zeitlich variierende Muster zu erkennen. Sie werden in künstliche Netzwerke (aRNNs) und biologische Netzwerke (bRNNs) unterteilt. Dabei umfassen aRNNs alle technischen Anwendungen, während bRNNs Netzwerke bezeichnen, die wichtige Gehirnfunktionen steuern. Diese Rückmeldungen helfen unter anderem bei kognitiven Prozessen wie Erinnern, Entscheiden und Lernen (vgl. GROSSBERG, 2013). RNNs bieten eine Vielzahl an Anwendungsmöglichkeiten in verschiedenen Forschungsfeldern. Beispielhafte Einsatzbereiche sind unter anderem die Kopfverfolgung bei Virtual-Reality-Systemen, Verfahren zur Wasserfilterung, Systeme zur Sprachverarbeitung oder robotische Anwendungen. Es werden verschiedene RNN-Strukturtypen unterschieden. Dazu zählen vollständig vernetzte Netze und teilweise vernetzte Netze. Die vollständig vernetzten Netze besitzen keine klare Trennung von Eingabe- und Ausgabeschicht. Jedes Neuron ist mit allen anderen verbunden und kann Signale auch an sich selbst zurückschicken (Siehe Abbildung 3) (vgl. MEDSKER et al., 2001). Die Ausgabeschicht gibt also Rückmeldungen über ihre vorherigen Zustände an die Eingabeschicht zurück, sodass Informationen zu früheren Zeitpunkten verarbeitet werden können. So ist das Netzwerk in der Lage, über mehrere Zeitschritte hinweg zu analysieren und im zeitlichen Verlauf komplexere und abstraktere Muster in den Eingabedaten zu erkennen (vgl. KATTE, 2018:124).

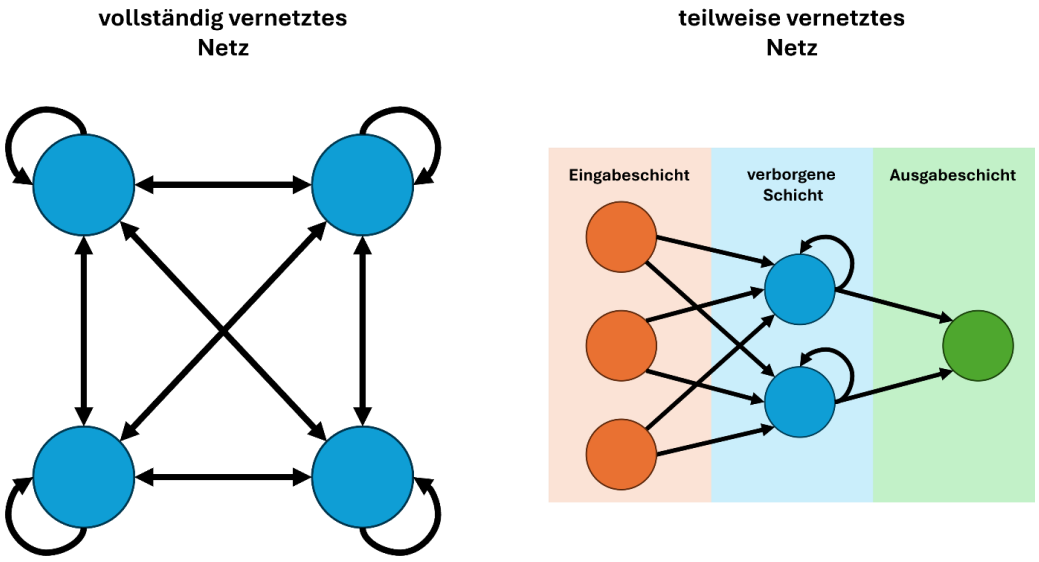



Abbildung 3: schematische Darstellung eines vollständig verbundenen RNN (in Anlehnung an MEDSKER et al. 2001) & Abbildung 4: schematische Darstellung eines teilweise verbundenen RNN (in Anlehnung an WALTERS, 2019)

## 2.2 Objekterkennungsalgorithmen

In diesem Unterkapitel werden verschiedene Objekterkennungsalgorithmen vorgestellt, darunter YOLO, SSD und R-CNN. Um einen fundierten Vergleich dieser Algorithmen zu ermöglichen, werden zuvor allgemein anerkannte Bewertungsmethoden erläutert. Dazu zählen die Grundgrößen Precision (P), Recall (R) sowie Intersection over Union (IoU). Darauf aufbauend folgen die zusammengesetzten Größen F1-Score, Average Precision (AP) und Mean Average Precision (mAP).

### 2.2.1 Bewertungsmetriken

In der Objekterkennung spielen verschiedene Bewertungsmetriken eine wichtige Rolle. Sie dienen dazu Modellleistungen einheitlich zu bewerten und zu vergleichen. Bevor die wichtigsten Kennzahlen erläutert werden, ist es zunächst notwendig zentrale Begriffe zu definieren, die als Grundlage dienen. Danach werden die Metriken erläutert.

#### Grundbegriffe

Ein True Positive (TP) liegt vor, wenn ein Modell ein wirkliches Objekt richtig erkannt hat. Es wurde also eine sogenannte Ground Truth Bounding Box richtig detektiert. Ein False Positive (FP) hingegen beschreibt eine fehlerhafte Erkennung. Es wird also ein nicht existierendes Objekt erkannt oder die Position eines vorhandenen Objekts falsch bestimmt. Wenn ein tatsächlich existierendes Objekt vom Modell nicht erkannt wird, wird dies als False Negative (FN) bezeichnet. Ein True Negative (TN) wird nicht gezählt, da es in jedem Bild unendlich viele potenzielle Bounding-Boxen gibt. (vgl. PADILLA et al., 2020:238).

#### Intersection over Union (IoU)

Das Erkennungsmaß Intersection over Union basiert auf dem Jaccard-Index J (JACCARD, 1901), einem aus der Botanik stammenden „*coefficient de communauté florale*“[8]. Dieser beschreibt das Verhältnis der Anzahl gemeinsamer Elemente zweier Mengen zur Gesamtanzahl der Elemente beider Mengen und wird mathematisch wie folgt definiert:

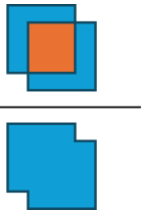
[8] Koeffizient der floralen Gemeinschaft

$$J = \frac{|\, A \cap B \,|}{|\, A \cup B \,|}$$

Formel 1: Jaccard-Index (nach JACCARD, 1901)

In der Objekterkennung wird IoU eingesetzt, um festzulegen, wann eine Erkennung korrekt oder falsch ist. Dabei vergleicht die IoU, wie sehr sich die vorhergesagte Bounding Box ($B_p$) und die tatsächliche Bounding Box ($B_{gt}$)[9] überlappen. Die IoU wird berechnet, indem die Fläche der Überlappung durch die gesamte Fläche der beiden Boxen geteilt wird[10] (vgl. PADILLA et al., 2020:238).

$$IoU = \frac{Fläche\ (B_p \cap B_{gt})}{Fläche\ (B_p \cup B_{gt})} = \frac{Fläche\ der\ Überlappung}{Fläche\ der\ Vereinigung} =$$



Formel 2: Darstellung der IoU Berechnung (PADILLA et al., 2020:238)

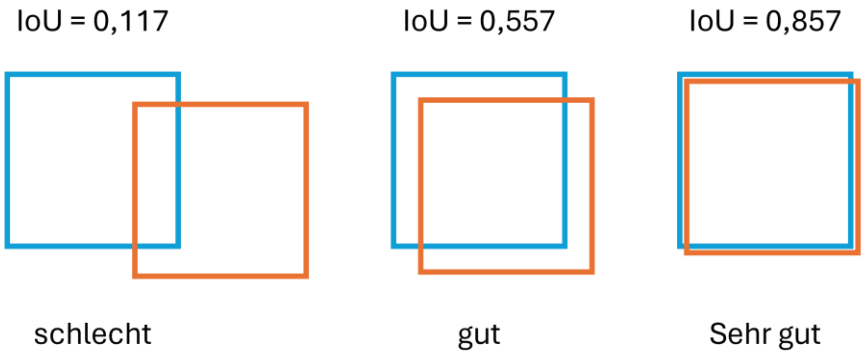



Abbildung 5: schematische Darstellung der IoU (in Anlehnung an TERVEN et al., 2023:1683)

## Precision

Precision gibt an, wie gut ein Modell ausschließlich relevante Objekte erkennt und wird durch den prozentualen Anteil der korrekt positiven Vorhersagen angegeben (vgl. PADILLA et al., 2020:238). Anschaulich beschreibt SIMONELLI (2020) Precision mit folgender

[9] Ground Truth Bounding Boxes
[10] Vereinigung

Frage: "*Out of all the times the model said the input was positive, what percentage were correct?*"

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections}$$

Formel 3: Berechnung der Precision (PADILLA et al., 2020:238)

## Recall

Recall beschreibt, wie gut das Modell alle tatsächlich vorhandenen Objekte erkennt, also alle Ground Truth Bounding Boxes. Es wird angegeben, welcher Anteil der tatsächlich existierenden Objekte auch wirklich vom Modell gefunden werden konnte (vgl. PADILLA et al., 2020:238). Anschaulich formuliert SIMONELLI (2020) Recall mit dieser Frage: "*Out of all the times there was a positive, what percentage were found by the model?"*

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truth}$$

Formel 4: Berechnung der Precision (PADILLA et al., 2020:238)

## Precision-Recall-Kurve

Die Precision-Recall-Kurve zeigt, wie sich Precision und Recall je nach Konfidenzwert verhalten. Eine hohe Konfidenzschwelle bedeutet wenige FP, also eine hohe Precision und umgekehrt. In diesem Fall kann es aber sein, dass viele positive Fälle übersehen werden. Dies führt zu einer hohen Anzahl an FN und damit zu einem niedrigen Recall. Ein sehr gutes Modell würde es somit schaffen, alle Objekte zu finden und nur die relevanten zu bestimmen. Eine perfekte Precision-Recall-Kurve würde konstant bei 100% Precision und 100% Recall liegen, unabhängig davon, wie der Konfidenzwert verändert wird (Siehe Abbildung 5, Kurve A). In der Realität verläuft die Kurve jedoch unterhalb des Maximalwerts und bildet einen Kompromiss aus Precision und Recall (Siehe Abbildung 5, Kurve B) (vgl. PADILLA et al., 2020:238).

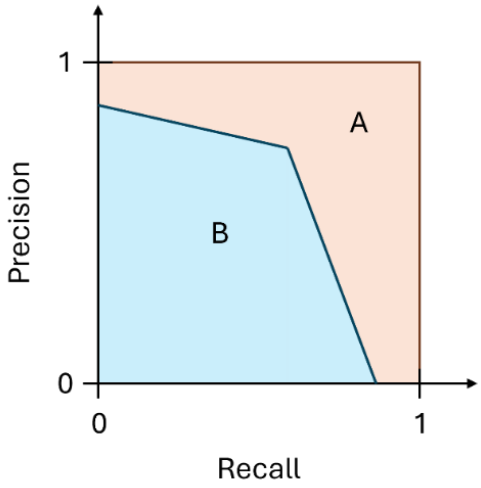



Abbildung 6: schematische Darstellung einer Precision-Recall-Kurve (A: perfekt, B: realistisch) (eigene Darstellung)

## Average Precision (AP)

Die meistverwendete Metrik zur Bewertung der Genauigkeit von Erkennungsmodellen ist die Average Precision. Sie wird für jede Klasse des Modells berechnet (vgl. PADILLA et al., 2020:238). Insgesamt bietet sie eine einzelne Zahl als Vergleichswert und bildet sich aus der Fläche unter der Precision-Recall-Kurve (vgl. ZHU et al., 2020:3).

## mean Average Precision (mAP)

Der Mittelwert der Average Precision über alle Klassen hinweg bildet das Referenzmaß mean Average Precision. Während die AP für jede Klasse berechnet wird, fasst die mAP alle Ergebnisse der Klassen zu einem Wert zusammen. Somit liefert mAP einen Gesamtgenauigkeitswert für das Modell (vgl. PADILLA et al., 2020:238).

Wird mAP als Bewertungsmaß herangezogen, werden in der Regel unterschiedliche IoU Schwellenwerte berücksichtigt. Die gängigsten Metriken sind mAP@0.50 und mAP@0.5-0.95[11]. Dabei misst mAP@0.5 wie gut ein Modell grobe Treffer erkennen kann. Überlappt also eine vorhergesagte Bounding Box die Box des Objekts mit bis zu 50% gilt die Vorhersage als korrekt. mAP@0.5-0.95 hingegen mittelt über 10 verschiedene IoU-Schwellenwerte zwischen 0.5 und 0.95. Es entsteht ein Mittelwert über verschiedene

[11] Im späteren Verlauf mAP50 und mAP50-95; In der Literatur werden verschiedene Bezeichnungen verwendet, beide beschreiben die selbe Bewertungsmetrik.

Genauigkeitsstufen. Daher liefert mAP@0.5-0.95 ein präziseres Maß für die Objekterkennung und wird vor allem bei COCO[12] verwendet (vgl. SIMONELLI, 2020).

## $F_1$-Score

Der $F_1$-Score kombiniert Precision und Recall zu einem einzigen Wert. Dieser wird als harmonisches Mittel der beiden Werte berechnet. Im Gegensatz zum arithmetischen Mittel ist der $F_1$-Score nur dann hoch, wenn sowohl Precision als auch Recall hohe Werte aufweisen (vgl. YUKATA, 2007:1-2). Es wird folgende Formel verwendet:

$$F_1 = \frac{2PR}{P + R}$$

Formel 5: Berechnung des F1 Scores (YUKATA, 2007:1)

## PASCAL Visual Object Classes (VOC) Challenge

Die PASCAL[13] Visual Object Classes (VOC) Challenge wird seit 2005 jährlich als Wettbewerb durchgeführt und bietet als Ergebnis einen standardisierten Datensatz mit annotierten Bildern und Evaluationsverfahren. Das Ziel der VOC-Challenge ist es die Leistungsfähigkeit von Erkennungsmethoden anhand vielfältiger natürlicher Bilder zu untersuchen und den Vergleich von Algorithmen zu erleichtern. Ebenfalls soll durch den Wettbewerb jährlich der Stand der Technik gemessen werden. Die VOC-Challenge gliedert sich in folgende zwei Hauptaufgaben:

Klassifikation: Enthält das Bild ein bestimmtes Objekt?

Detektion: Wo befindet sich das Objekt im Bild?

Darüber hinaus gibt es die beiden Nebenaufgaben Segmentierung auf Pixelebene und Person Layout. Bei ersterem wird jedem Pixel eine Klasse zugeordnet. Beim Person Layout werden Kopf, Hände und Füße von Personen lokalisiert. Der daraus entstehende VOC-Datensatz kommt hauptsächlich in den Forschungsbereichen Computer Vision und Machine Learning zum Einsatz. Über die Jahre hat sich dieser als maßgeblicher Benchmark für die Objekterkennung etabliert (vgl. EVERINGHAM et al., 2009:303-305).

[12] Siehe COCO Abschnitt
[13] Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL)

### Microsoft Common Objects in Context (COCO)

Microsoft Common Objects in Context (COCO) beinhaltet 328.000 alltägliche Bilder mit 91 Objektkategorien und 2,5 Millionen annotierten Objekten. COCO wurde dabei speziell für die Erkennung und Segmentierung von Objekten in ihrem natürlichen Umfeld erstellt. Die Anzahl der annotierten Objekte und Klassen ist weit größer als es bei VOC der Fall ist, wodurch das Erlernen kontextueller Informationen unterstützt wird. Insgesamt enthält COCO (7,7) im Durchschnitt deutlich mehr annotierte Objekte pro Bild als VOC (2,3). Im Gegensatz zu Bounding Boxes sind hier Objekte durch eine pixelgenaue Erkennung annotiert. Der COCO-Datensatz ist dadurch weitaus detaillierter und anspruchsvoller, weshalb er als zentraler Benchmark in der Objekterkennung gilt (vgl. LIN et al., 2014:740-748). Insbesondere moderne Ansätze der Objekterkennung nutzen COCO für die Evaluierung.

## 2.2.2 Region-based Convolutional Neural Network (R-CNN)

Region-based Convolutional Neural Network (R-CNN) von GIRSHICK et al. (2014) gilt als erster großer Ansatz der modernen Objekterkennung. Anders als die Vorgänger, die mit SIFT und HOG[14] arbeiten, wurde bei diesem Ansatz zum ersten Mal CNN für die Objekterkennung verwendet. R-CNN hat es sich daher zum Ziel gesetzt, ältere Ansätze mithilfe eines CNN in Hinblick auf Leistung deutlich zu übertreffen (vgl. GIRSHICK et al., 2014:1). Im Folgenden wird die Architektur und Funktionsweise erläutert.

### Einführung

R-CNN basiert auf dem namensgebenden „*Recognition using Region*“ Paradigma, welches für die Objekterkennung und Segmentierung verwendet wird (vgl. GIRSHICK et al., 2014:1-2). Dabei wird das Bild nicht als Ganzes wahrgenommen, sondern es werden zunächst potenziell relevante Bildausschnitte[15] erzeugt (vgl. GU et al., 2009). Anschließend werden diese Bildregionen anhand des CNN verarbeitet und klassifiziert. Somit können Regionen vorgeschlagen werden, die mit hoher Wahrscheinlichkeit ein

[14] SIFT (Scale-Invariant Feature Transform) und HOG (Histogram of Oriented Gradients) sind Verfahren, die vor allem vor der Entwicklung von Deep-Learning-basierten Ansätzen für die Objekterkennung verwendet wurden.
[15] Regionenvorschläge (Region Proposals)

Objekt enthalten. R-CNN kombiniert die Regionenvorschläge mit CNN und ermöglicht dadurch eine höhere Genauigkeit der Objekterkennung im Vergleich zu früheren Ansätzen. Mit einem mAP-Wert von 53,7 % übertrifft R-CNN den zuvor etablierten multi-feature-Ansatz mit nicht linearem Kernal-SVM, der 35,1 % erreichte (vgl. GIRSHICK et al., 2014:1-2).

## Erkennung

Das System zur Objekterkennung besteht insgesamt aus drei aufeinanderfolgenden Modulen, die zunächst kurz erläutert werden. Das erste Modul erzeugt die Regionenvorschläge, diese sind noch keiner Kategorie zugeordnet. Die Vorschläge definieren die Menge an Möglichkeiten, die dem Detektor zur Verfügung stehen. Das zweite Modul besteht aus dem CNN, welches aus den Regionen Merkmalsvektoren mit einer festen Länge extrahiert. Am Ende werden die Merkmalsvektoren mithilfe von klassenspezifischen linearen Support Vector Machines[16] (SVMs) klassifiziert (vgl. GIRSHICK et al., 2014:1).

## Architektur

Für die Erzeugung der Regionenvorschläge können verschiedene Methoden herangezogen werden. R-CNN ist dabei unabhängig von der gewählten Methode. Für eine Vergleichbarkeit mit früheren Arbeiten haben sich GIRSHICK et al. (2014) für die Selective Search[17] Methode entschieden. Die aus Regionenvorschlägen resultierenden Bildausschnitte werden anschließend vom CNN verarbeitet. Aus dem Netzwerk wird ein 4096-dimensionaler Merkmalsvektor extrahiert. Dabei kommt die Caffe-Implementierung[18], des von KRIZHEVSKY et al. (2012) beschriebenen CNN zum Einsatz. Die Merkmale werden berechnet, indem das Bild zuerst auf 227 x 227 skaliert und der Mittelwert der Bildwerte abgezogen wird. Anschließend läuft es durch das Feedforward-CNN, welches aus fünf Faltungsschichten und zwei vollständig verbundenen Schichten

[16] SVMs sind Lernverfahren des Machine Learnings, die entwickelt wurden, um Daten optimal zu trennen und zu klassifizieren. (vgl. MAMMONE, 2009)
[17] Selective Search ist eine Methode, die aus Bildsegmentierungen Regionenvorschläge für Objekte im Bild erzeugt. (vgl. UIJLINGS, 2013)
[18] Convolutional Architecture for Fast Feature Embedding (Caffe) ist ein Framework zum schnellen trainieren und Anwenden von Deep-Learning-Modellen. (vgl. JIA et al., 2014)

besteht. Die so gewonnen Merkmalsvektoren werden zum Schluss mithilfe der klassenspezifischen linearen SVMs klassifiziert (vgl. GIRSHICK et al., 2014:2-3).

### Weiterentwicklung

Da nach jedem Regionen Vorschlag CNN angewendet werden muss, ist der Rechenaufwand für das Verfahren hoch. Daher wurde R-CNN im Laufe der Zeit weiterentwickelt, mit dem Ziel schneller und effizienter zu werden. Jede neue Version sorgte dabei nicht nur für Leistungssteigerungen, sondern auch für methodische Verbesserungen. Neue Innovationen waren zum Beispiel optimierte Verarbeitungen oder Segmentierungen. In der folgenden Tabelle sind die wichtigsten Weiterentwicklungen aufgeführt.

| Version | Entwickler | Jahr | Innovation | mAP50 | mAP50-95 |
|---|---|---|---|---|---|
| R-CNN | GIRSHICK et al. | 2014 | Region Proposals, Selective Search, CNN-Features, SVM | 53,3 % (VOC) | - |
| Fast R-CNN | GIRSHICK | 2015 | RoI-Pooling, end-to-end training, single pass through the CNN | 68,1 % (VOC) | - |
| Faster R-CNN | REN et al. | 2016 | Region Proposal Network | 69,9 % (VOC), 41,5 % (COCO) | 21,2 % |
| Mask R-CNN | HE et al. | 2017 | RoIAlign, Instance Segmentation | 60 % (COCO) | - |

Tabelle 1: R-CNN Weiterentwicklungen im Überblick (eigene Darstellung)

## 2.2.3 You Only Look Once (YOLO)

In den letzten Jahren wurden zahlreiche Ansätze zur Objekterkennung entwickelt. Viele davon enthielten jedoch komplexe Verfahren oder benötigten hohe Rechenressourcen. Mit der Entwicklung von YOLO oder auch „*You Only Look Once*“ wurde ein neuer Ansatz eingeführt, der die Objekterkennung deutlich vereinfacht und beschleunigt. Das Verfahren gilt als eines der bekanntesten Modelle für die Echtzeit-Objekterkennung und hat durch seine einheitliche Architektur maßgeblich zur Weiterentwicklung der

Objekterkennung beigetragen. Im folgenden Kapitel werden das Konzept und die Funktionsweise der YOLO-Architektur näher erläutert.

## Einführung

YOLO basiert auf einem simplen Prinzip: Ein CNN sagt in einem Durchlauf gleichzeitig mehrere Bounding Boxes und die zugehörigen Klassenwahrscheinlichkeiten vorher. Das Verfahren wird dabei mit vollständigen Bildern trainiert, wodurch sich die Erkennungsleistung unmittelbar verbessert. Im Gegensatz zu anderen Objekterkennungsmethoden ist YOLO deutlich schneller, da keine aufwendige Verarbeitungspipeline vorausgesetzt wird. Gleichzeitig erzielte bereits die erste Version eine gute Genauigkeit im Verhältnis zur Geschwindigkeit und war schon damals der Konkurrenz weit voraus. Seitdem wurde YOLO mehrfach weiterentwickelt, um bestimmte Eigenschaften noch weiter zu verbessern. Diese Versionen werden im weiteren Verlauf des Kapitels erläutert. Der zweite große Vorteil besteht darin, dass YOLO ein Bild immer im Gesamtkontext analysiert. Dadurch werden zusammenhängende Informationen über Klassen und deren Aussehen erfasst, wodurch weniger Hintergrundfehler gemacht werden als bei anderen Methoden. Ebenfalls lernt YOLO generalisierbare Objektrepräsentationen. Dies sorgt dafür, dass das Modell auf neuen Datenbereichen weniger fehleranfällig ist. Jedoch hatten vor allem ältere YOLO-Versionen Probleme, kleine Objekte präzise zu erkennen (vgl. REDMON et al., 2016:779-780).

## Erkennung

Bei YOLO werden die verschiedenen Schritte der Objekterkennung in einem CNN zusammengefasst. Das gesamte Bild wird berücksichtigt, um alle Objekte und deren Position zu erfassen. Dadurch wird die Erkennung schneller, das Modell einfacher zu trainieren und die durchschnittliche Genauigkeit bleibt hoch. Bei der Erkennung teilt YOLO das Eingabebild in ein Raster auf und wenn das Zentrum eines Objekts in eine Rasterzelle fällt, ist diese zuständig für die Erkennung. Dabei sagt jede Rasterzelle die Bounding Boxes und die Konfidenzwerte der enthaltenen Objekte vorher. Die Konfidenzwerte geben an, wie sicher sich das Modell ist, dass die Bounding Boxes an der richtigen Stelle gesetzt wurden. Befindet sich kein Objekt in der Zelle sollte die Konfidenz bei Null liegen. Existiert jedoch ein Objekt entspricht der Konfidenzwert der IoU zwischen

der vorhergesagten Box und der Ground Truth Box. Jede Bounding Box besteht aus den folgenden Komponenten: den x- und y-Koordinaten des Boxzentrums, Breite (w) und Höhe (h) relativ zum Gesamtbild sowie einem Konfidenzwert. Außerdem sagt jede Rasterzelle eine Klassenwahrscheinlichkeit vorher, insofern ein Objekt in der Zelle enthalten ist. Werden die Klassenwahrscheinlichkeiten mit den Konfidenzwerten der einzelnen Boxen multipliziert entsteht ein klassenspezifischer Konfidenzwert. Dieser gibt an, wie wahrscheinlich es ist, dass die Klasse in einer Box vorkommt und wie gut die Box das Objekt abdeckt (vgl. REDMON et al., 2016:780).

## Architektur

Das YOLO-Modell ist in einem CNN implementiert. Dieses basiert auf GoogleLeNet, einem bekannten CNN für Bildklassifikationen. Im Gegensatz zu GoogleLeNet werden vereinfachte Faltungsschichten verwendet. Die ersten Faltungsschichten extrahieren Bildmerkmale, während die vollständig verbundenen Schichten die Wahrscheinlichkeit und Koordinaten der Objekte vorhersagen. Das ursprüngliche Modell besteht aus 24 Faltungsschichten und 2 vollständig verbundenen Layern. Anfangs wurde dazu auch eine Fast YOLO-Variante mit 9 Faltungsschichten bereitgestellt (vgl. REDMON et al., 2016:780). Da das Modell oft mehrere überlappende Bounding Boxes für dasselbe Objekt vorhersagt, wird in der Nachbereitung Non-Maximum Suppression (NMS) verwendet. Dieser Verarbeitungsschritt wird angewendet, um die Anzahl an sich überlappenden Bounding Boxes zu reduzieren und somit die Gesamtqualität der Erkennung zu erhöhen. Dabei werden irrelevante Bounding Boxes entfernt und nur die genausten behalten (vgl. TERVEN et al., 2023:1683).

## Weiterentwicklung

Zuvor wurde das Grundkonzept der ersten YOLO-Version erläutert. Basierend darauf wurde YOLO über die letzten Jahre kontinuierlich weiterentwickelt. Verbesserungen in Genauigkeit, Effizienz und Vielseitigkeit haben YOLO-Modelle leistungs- und anpassungsfähiger für unterschiedlichste Aufgaben gemacht. Die einzelnen Versionen zeichnen sich durch unterschiedliche architektonische Veränderungen aus und bieten neue Innovationen. Während sich frühere Versionen vor allem auf die Steigerung von Genauigkeit und die Integration von neuen Mechanismen wie Anchorboxes oder Multi-

Scale konzentrierten, veränderte sich der Fokus von späteren Generationen zu effizienteren Backbones, verbesserten Trainingsstrategien und geringerem Rechenaufwand. Ebenfalls wurde die Modularität der Modelle ausgebaut, sodass verschiedene Varianten für unterschiedliche Hardwareumgebungen entstanden sind[19]. YOLO hat sich somit von einem reinen Echtzeitdetektor zu einer flexiblen Modellreihe mit unterschiedlichen Anwendungsmöglichkeiten entwickelt.

Tabelle 1 zeigt die Entwicklung der YOLO-Versionen von der ersten Veröffentlichung bis hin zur aktuellen Variante. Aufgeführt werden die Entwickler, das Erscheinungsjahr, zentrale Innovationen und getestete mittlere Genauigkeiten je nach Schwellenwert. Da vor allem spätere Varianten unterschiedliche Modellgrößen bereitstellen, wurde immer der Wert für die größte Modellvariante angegeben, dieser repräsentiert damit den höchstmöglichen Wert. Spätere Versionen wurden meistens nicht mehr mit mAP50 verglichen, denn mit der Einführung von COCO wurde mAP50-95 als Standardbewertungsmaß verwendet. Dieses Maß lieferte ein aussagekräftigeres Bild als mAP50. Aus diesem Grund sind teilweise keine mAP50-Werte für neuere Versionen auffindbar. Alle mAP50-95 Werte beziehen sich auf den Standarddatensatz COCO.

| Version | Entwickler | Jahr | Innovation | mAP50 | mAP50-95 |
|---|---|---|---|---|---|
| YOLO | REDMON et al. | 2015 | realtime one-stage detection | 63,4 % (VOC) | – |
| YOLOv2 | REDMON & FARHADI | 2016 | Anchor Boxes, Multi-Scale Training, Darknet Backbone | 76,8 % (VOC), 44 % (COCO) | 21,6 % |
| YOLOv3 | REDMON & ALI FARHADI | 2018 | Multi-Scale Prediction, Residual-Backbone (Darknet-53) | 57,9 % (COCO) | – |
| YOLOv4 | BOCHKOVSKIY et al. | 2020 | CSPDarknet-53 Backbone, SAT, Data Augmentation, CIoU loss | 65,7 % (COCO) | 43,5 % |
| YOLOv5 | JOCHER et al. | 2020 | PyTorch-based framework, DLA training strategy, exportability | 72,7 % (COCO) | 56,8 % |

[19] n (nano), s (small), m (medium), l (large), x (extra-large) (vgl. JOCHER et al., 2024a)

| | | | | | |
|---|---|---|---|---|---|
| YOLOv6 | LI et al. | 2022 | EfficientRep Backbone, Decoupled head | – | 52,8 % |
| YOLOv7 | WANG et al. | 2022 | E-ELAN, RepConvN., trainable bag-of-freebies | – | 56,8 % |
| YOLOv8 | JOCHER et al. | 2023 | anchor-free head, unified tasks, User-friendly Python Package, multi-task support | – | 53,9 % |
| YOLOv9 | WANG et al. | 2024 | PGI, GELAN, DFL v2, improved label assignment | – | 55,6 % |
| YOLOv10 | WANG et al. | 2024 | anchor-free training & inference, NMS-free training, holistic model design | – | 54,4 % |
| YOLO11 | JOCHER et al. | 2024 | C3k2, C2PSA, improved feature extraction, multi-task versatility | – | 54,7 % |
| YOLO12 | YUNJIE et al. | 2025 | Area Attention, R-ELAN, Optimized Attention Architecture, Comprehensive Task Support | 72 % (COCO) | 55,2 % |

Tabelle 2: YOLO-Versionen im Überblick (eigene Darstellung in Anlehnung an KOTTHAPALLI 2025:11 & JIANG, 2025:3)

### 2.2.4 Single Shot MultiBox Detector (SSD)

Nach der Einführung von YOLO wurde ebenso an weiteren Ansätzen für die automatische Objekterkennung geforscht. Einer dieser neuen Ansätze basiert auf einem ähnlichen Prinzip. Single Shot MultiBox Detector gilt als eines der bekanntesten Verfahren in der automatischen Objekterkennung. Im Folgenden wird die Funktionsweise und Architektur von SSD genauer erklärt.

#### Einführung

Der Single Shot MultiBox Detector (SSD) greift die Idee der direkten Objekterkennung in einem Verarbeitungsschritt auf, erweitert sie jedoch um wichtige Verbesserungen in

Genauigkeit und Geschwindigkeit. Dabei ist SSD der erste Objektdetektor, welcher auf einem tiefen neuronalen Netz basiert. Folgende Verbesserungen wurden im Vergleich zu älteren Verfahren angewendet:

1. Einsatz von kleinen Faltungsfiltern für die Vorhersage von Objektklassen und Bounding Box Verschiebungen.
2. Einführung von separaten Filtern für unterschiedliche Seitenverhältnisse.
3. Anwendung der Filter auf viele Merkmalskarten aus späteren Netzwerkschichten, um Detektion auf mehreren Skalen zu ermöglichen.

Diese Verbesserungen ermöglichen eine hohe Genauigkeit auch bei niedrigen Eingangsauflösungen, wodurch die Detektionsgeschwindigkeit ebenfalls erhöht wird. Im Vergleich zur ersten YOLO-Version mit mAP 63,4 erreicht SSD ein mAP von 74,3, was eine starke Verbesserung darstellt (vgl. LIU et al., 2016:22).

## Erkennung

Für die Erkennung benötigt SSD sowohl das Eingabebild als auch die Ground Truth Bounding Boxes für jedes Objekt. Dabei nutzt SSD Merkmalskarten mit verschiedenen Auflösungen, um Objekte auf unterschiedlichen Skalen zu erfassen. An jeder Position dieser Merkmalskarten werden mehrere vordefinierte Standard-Boxes mit unterschiedlichen Größen und Seitenverhältnissen platziert. Anschließend berechnet das Modell für jede Standard-Box die Anpassungen der Boxposition[20], sowie die Konfidenzwerte für jede Klasse. Damit mehrere überlappende Boxen herausgefiltert werden wird auch hier NMS angewendet. So bleiben nur die relevanten Boxen als Ergebnis erhalten (vgl. LIU et al., 2016:23-24).

## Architektur

SSD basiert auf einem feedforward-CNN[21] (FFCNN). Dieses Netzwerk erzeugt viele Bounding-Boxes mit unterschiedlichen Größen und berechnet für jede Box die Wahrscheinlichkeit, dass sich ein Objekt darin befindet. Insgesamt besteht das Netzwerk aus zwei Teilen, dem Basennetzwerk und den zusätzlichen Schichten. Das

[20] Offset
[21] Ein FFCNN ist die einfachste Form von ANNs, hier können Daten nur in eine Richtung fließen. (TANVEER et al., 2023)

Basennetzwerk wird für die Bildklassifikation verwendet und die anschließenden zusätzlichen Schichten ermöglichen die Objekterkennung. Die zusätzlichen Schichten bestehen aus den folgenden Bauteilen, welche zusammen die Objekterkennung realisieren:

1. Mehrskalen Merkmalskarten: Zusätzliche Faltungsschichten sind direkt nach dem Basennetzwerk eingebaut. Diese Schichten werden schrittweise kleiner, sodass eine Vorhersage von Objekten auf mehreren Skalen gewährleistet wird.
2. Convolutional Predictor: Mithilfe von Faltungsfiltern wird für jede Zelle in der Merkmalskarte die wahrscheinlichste Klasse und die Anpassung der Bounding Box berechnet.
3. Standard-Boxes: Zuletzt werden jeder Zelle einer Merkmalskarte mehrere vordefinierte Boxes mit unterschiedlicher Form und Größe zugeordnet. Diese Boxes liegen immer an festen Positionen relativ zu ihrer Zelle. Sie liefern die Ausgangsposition für die Vorhersagen des Convolutional Predictors.

Durch die Kombination aus diesen Bauteilen können die zusätzlichen Schichten effizient Objekte mit verschiedenen Formen und Größen erkennen (vgl. LIU et al., 2016:23-25).

## Weiterentwicklung

Nachdem das Grundkonzept von SSD erläutert wurde wird im Folgenden auf Weiterentwicklungen des Modells eingegangen. In dem darauffolgenden Jahr nach der Veröffentlichung etablierten sich zwei neue Ansätze, die neue Ideen einbrachten und die Leistung des Basismodells verbessern. Die Entwicklungen zielten darauf ab eine höhere Erkennungsgenauigkeit bei kleinen Objekten, bessere Feature-Extraktion und effizientere Nutzung von Rechenressourcen zu gewährleisten (vgl. FU et al., 2017 & LI et al., 2017). Weitere Ansätze zielten auf spezielle Anwendungsfälle ab und werden im weiteren Verlauf der Arbeit thematisiert. Die nachfolgende Tabelle zeigt die zentralen Weiterentwicklungen und stellt die Grundeigenschaften vergleichend dar.

| Version | Entwickler | Jahr | Innovation | mAP50 | mAP50-95 |
|---|---|---|---|---|---|
| SSD | LIU et al. | 2016 | Base One-Stage Architecture, Multi-Scale-Feature-Maps, Default Boxes | 76,9 % (VOC) | _ |
| DSSD | FU et al. | 2017 | Feedforward connections in deconvolution, new output module | 81,5 % (VOC), 53,3 % (COCO) | 33,2 % |
| FSSD | LI et al. | 2017 | Feature fusion module | 84,2 % (VOC), 52,8 (COCO) | 31,8 % |

Tabelle 3: SSD Weiterentwicklungen im Überblick (eigene Darstellung)

### 2.2.5 RetinaNet

Nach der Vorstellung von SSD wird mit RetinaNet ein weiterer bedeutender Vertreter von einstufigen Detektoren[22] betrachtet. Das Modell wurde entwickelt, um Schwächen früherer Verfahren auszubessern und neue Impulse in der Objekterkennung zu setzen. Daher hat RetinaNet einen entscheidenden Einfluss auf die Weiterentwicklung der Objekterkennung.

#### Einführung

LIN et al. (2017a) haben es sich zur Aufgabe gemacht mit RetinaNet den ersten einstufigen Detektor zu entwickeln, welcher qualitativ ähnliche Ergebnisse liefert, wie die zweistufigen Verfahren. Als Hauptproblem wurde dabei das Klassenungleichgewicht während des Trainings identifiziert. Für die Lösung schlagen sie eine neue Verlustfunktion[23] vor. Bei dieser handelt es sich um einen dynamisch skalierbaren Kreuzentropie-Loss[24], bei welcher der Skalierungsfaktor gegen Null abnimmt, wenn die Sicherheit für eine korrekte Klasse steigt. Dadurch kann die Gewichtung einfacher Beispiele leichter reduziert werden und sich das Modell auf schwierige Beispiele konzentrieren. In Zuge von Experimenten wurde herausgefunden, dass der

[22] One Stage Detector
[23] Die Verlustfunktion wird verwendet, um die Leistung von Machine Learning Modellen zu messen. (vgl. PYKES, 2024)
[24] Kreuzentropie-Loss (Cross-Entropy-Loss) ist eine spezielle Verlustfunktion, welche die Differenz aus vorhergesagter Wahrscheinlichkeit und den tatsächlichen Werten misst. (vgl. PYKES, 2024)

vorgeschlagene Focal-Loss herkömmliche Methoden übertrifft und es daher möglich macht einen genauen einstufigen Detektor zu trainieren. Dieser wird als RetinaNet bezeichnet (LIN et al., 2017a:1-2).

## Erkennung

Die Objekterkennung mit RetinaNet läuft wie folgt ab. Zuerst wird das Eingabebild durch den Backbone geleitet, welches daraus mehrskalige Merkmalskarten erstellt. Auf diesen Karten werden dann Anker mit verschiedener Seitenlänge und Größe platziert. Sie dienen als Ausgangspunkt für mögliche Objekte. Anschließend ordnet das Klassifikations-Subnetz jedem Anker die zugehörige Objektklasse zu. Das Regressions-Subnetz berechnet Größe und Form der Bounding Boxen. Im letzten Schritt kommt auch hier NMS zum Einsatz, um überflüssige Boxen herauszufiltern. So liefert RetinaNet für jedes Bild eine Menge erkannter Objekte mit passender Klasse und Begrenzungsrahmen (vgl. LIN et al., 2017a:4-5). Im nächsten Unterkapitel wird genauer auf die Bauteile eingegangen.

## Architektur

RetinaNet ist ein einheitliches Netzwerk, welches aus einem Backbone-Netzwerk und zwei Subnetzwerken mit speziellen Aufgaben besteht. Der Backbone ist für die Berechnung der Faltungsmerkmalskarten verantwortlich und basiert auf einem vortrainierten CNN. Das erste Subnetz übernimmt die Objekterkennung auf der Ausgabe des Backbones, während das zweite Subnetz die Regressionsberechnung der Bounding Boxes durchführt. Das Design beider Subnetze ist dabei speziell auf ein einstufiges System ausgelegt. RetinaNet besteht aus folgenden Bauteilen:

Feature Pyramid Network (FPN) Backbone: FPN ergänzt ein normales CNN um einen Top-Down-Pfad und laterale Verbindungen. So kann das Netzwerk effizient eine mehrskalige Merkmalspyramide aus einem Eingabebild mit fester Auflösung erstellen. Dabei ist jede Ebene dafür zuständig, Objekte von verschiedenen Größen zu erkennen.

Anchor Boxes: Jede Pyramidenebene verwendet Anchor Boxes, die Objekte in verschiedenen Größen und Seitenverhältnissen abdecken. Jeder Anker wird dann entweder einem Objekt oder dem Hintergrund zugeordnet. Anker, die keinem Objekt klar zugeordnet werden können, werden ignoriert. Für jeden Anker mit Objekt werden

zusätzlich Offsets berechnet, um die genaue Position der Bounding Box vorherzusagen. Insgesamt besteht das Modell aus 9 Ankern pro Ebene.

Klassifikations-Subnetz: Dieses erste Subnetz sagt für jede Position, jeden Anker und jede Objektklasse die Wahrscheinlichkeit für das Vorhandensein eines Objekts vorher. Es ist ein kleines FCN[25] , dass an jeder Ebene angeschlossen ist. Die Parameter werden über alle Pyramidenebenen verteilt.

Box-Regression-Subnetz: Dieses Subnetz ist ebenfalls ein FCN, welches dafür zuständig ist, für jeden Anker die genaue Verschiebung zur richtigen Bounding Box vorherzusagen. Es ist ähnlich aufgebaut wie das vorherige Subnetz, aber mit 4 Werten pro Anker, welche die Position und Größe der Box bestimmen (vgl. LIN et al., 2017a:4-5).

## Weiterentwicklung

Im Lauf der Zeit sind einige Entwicklungen herausgekommen, welche die Grundidee hinter RetinaNet unterschiedlich ausbauen. Kernthemen dieser Modelle waren unter anderem ankerfreie Ansätze, effizientere Feature-Pyramiden oder neue Verlustfunktionen. Das Ziel aller Ansätze ist es jedoch die Balance zwischen Genauigkeit und Rechenaufwand zu verbessern. In der folgenden Tabelle sind die wichtigsten Weiterentwicklungen zusammengefasst.

| Version | Entwickler | Jahr | Innovation | mAP50 | mAP50-95 |
|---|---|---|---|---|---|
| RetinaNet | LIN et al. | 2017a | Focal Loss, One-Stage, FPN | 59,1 % (COCO) | 39,1 % |
| ATSS | ZHANG et al. | 2020 | Anchor-Free, Adaptive Training Sample Selection | 68,9 % (COCO) | 50,7 % |
| GFL | LI et al. | 2020 | Generalized Focal Loss (GFL, DFL) | 67,4 % (COCO) | 48,2 % |
| VFNet | ZHANG et al. | 2021 | Varifocal Loss, star-shaped bounding box feature representation | 73 % (COCO) | 55,1 % |

Tabelle 4: RetinaNet Weiterentwicklungen im Überblick (eigene Darstellung)

[25] Vollständig konvolutionales Netzwerk (Fully Convolutional Network)

## 2.3 Objektsegmentierung

Die Segmentierung von Objekten bildet einen weiteren großen Aspekt in der Bildverarbeitung und Computer Vision (vgl. MINAEE et al., 2020:1). Im Folgenden werden die Grundlagen erklärt, spezielle Bewertungsmetriken vorgestellt und etablierte Modellansätze erläutert.

### 2.3.1 Grundlagen

Während die Erkennung das Ziel verfolgt Objekte anhand eines Begrenzungsrahmens zu lokalisieren und zu klassifizieren, liegt die Aufgabe bei der Segmentierung darin, Konturen eines Objekts auf Pixelebene zu bestimmen (vgl. VERSCHAE & SOLAR, 2015:1; MINAEE et al., 2020:1). Anwendungsfelder sind beispielsweise medizinische Bildanalysen, robotische Wahrnehmung, Augmented Reality oder Videoüberwachung. Zu den frühen Ansätzen der Segmentierung zählen dabei Verfahren wie Thresholding, Region Growing oder der Watershed-Algorithmus. Später folgten fortgeschrittenere Methoden wie Active Contours, Graph Cuts sowie Conditional und Markov Random Fields. Der Einsatz von Deep Learning führte jedoch zu einer neuen Generation an Bildsegmentierungsverfahren. Diese zeichnen sich durch eine deutliche Leistungssteigerung und hohe Genauigkeit aus. Segmentierung kann dabei in zwei Arten unterteilt werden zum einen die semantische Segmentierung, bei der jedem Pixel eine Kategorie zugeordnet wird, und zum anderen die Instanz Segmentierung, die einzelne Objekte im Bild voneinander abgrenzt (vgl. MINAEE et al., 2020:1).

### 2.3.2 Bewertungsmetriken

Einige Bewertungsmetriken, wie Precision, Recall, Intersection over Union und der F1-Score wurden bereits zuvor im Objekterkennungskapitel erläutert und spielen auch für die Evaluation der Segmentierung eine wichtige Rolle, da sie auf Pixel- oder Maskenebene angewendet werden. Darüber hinaus gibt es aber auch weitere spezifische Metriken, die speziell für die Bewertung der Segmentierungsqualität entwickelt wurden. Im Folgenden Kapitel werden die wichtigsten Metriken vorgestellt.

Die Grundbegriffe True Positive, False Positive, True Negative und False Negative wurden bereits im Kapitel zur Objekterkennung erläutert (Siehe Kapitel 2.2.1). Während sie dort

auf Bounding Boxes angewendet werden, beziehen sie sich in der Segmentierung auf die Klassifikation einzelner Pixel. Da in der Segmentierung jeder Pixel eine Klassenzuordnung erhält, wird anders als bei der Erkennung auch ein False Negative aktiv ausgewertet.

## Pixel Accuracy (PA)

Die Pixel Accuracy (PA), auch bekannt als Rand Index, misst den Anteil an korrekt klassifizierten Pixeln im Bild. Sie gilt als bekannteste Metrik in der Bildsegmentierung und wird definiert als Anzahl der korrekten Vorhersagen, bestehend aus den richtigen Vorhersagen und negativen Vorhersagen, im Verhältnis zur Gesamtzahl der Vorhersage. Für ungleich verteilte Klassen ist sie jedoch wenig aussagekräftig (vgl. MÜLLER et al., 2022:6).

$$PA = \frac{TP + TN}{TP + TN + FN + FP}$$

Formel 6: Berechnung der PA (MÜLLER et al.,2022:6)

## Dice Similarity Coefficient (DSC)

Der Dice Similarity Coeffizient (DSC) gilt als Standardmetrik in der Bildsegmentierung und misst die Überlappung zwischen der vorhergesagten Segmentierungsmaske und der Ground Truth Maske. Mathematisch entspricht er dem harmonischen Mittel und ist dem F1-Score gleichzusetzen. In der Segmentierung wird jedoch der DSC bevorzugt eingesetzt, da dieser die Bewertung auf Pixel- oder Maskenebene ausdrückt (vgl. MÜLLER et al., 2022:6).

$$DSC = \frac{2TP}{2TP + FP + FN}$$

Formel 7: Berechnung des DSC (MÜLLER et al.,2022:6)

## Cohens Kappa (Kap)

Cohen's Kappa (Kap) ist eine Metrik, die misst, wie gut die Vorhersagen eines Modells mit den tatsächlichen Ergebnissen übereinstimmen. Dabei werden die durch Zufall verursachten Übereinstimmungen gemessen. Der Wert kann zwischen -1 und +1 liegen, wobei +1 für eine perfekte Übereinstimmung steht, 0 für einen zufälligen Wert und -1 für

eine komplette Unstimmigkeit. Besonders nützlich ist Kap bei unausgewogenen Datensätzen. Daher wird diese Metrik besonders im Bereich des Machine Learnings angewendet. Bei ausgewogenen Datensätzen hingegen sorgt Kap eher für höhere Werte. Außerdem lässt sich Kap nicht gut zwischen verschiedenen Datensätzen vergleichen und dient nicht direkt als Maß für die Vorhersagegenauigkeit (vgl. MÜLLER et al., 2022:6).

$$\mathrm{f_c} = \frac{(TN + FN)(TN + FP) + (FP + TP)(FN + TP)}{TP + TN + FP + FN}$$

Formel 8: Berechnung des Zufallskorrekturterms $\mathrm{f_c}$ für Kap (MÜLLER et al., 2022:7)

$$Kap = \frac{(TP + TN) - f_c}{(TP + TN + FN + FP) - f_c}$$

Formel 9: Berechung von Kap (MÜLLER et al., 2022:7)

## Hausdorff Distance (HD)

Die Hausdorff Distance misst, wie weit zwei Punktmengen auseinanderliegen, zum Beispiel zwischen der Ground Truth Segmentierung und der vorhergesagten Segmentierung. Dabei wird sich auf die Konturen konzentriert, wodurch eine Bewertung der lokalisierten Ähnlichkeit ermöglicht wird. Besonders bei komplexen Segmentierungsaufgaben spielt die exakte Vorhersage der Konturen eine wichtige Rolle. Da die HD jedoch sehr empfindlich gegenüber Ausreißern ist, wird meistens die robustere Average Hausdorff Distance verwendet (AHD). Sie misst den durchschnittlichen Abstand zwischen den Konturen (vgl. MÜLLER et al., 2022:6).

$$d(A, B) = \frac{1}{|A|} \sum_{\{a \in A\}} \min_{\{b \in B\}} |a - b|$$

Formel 10: Berechnung des durchschnittlichen Abstands von A zu B (MÜLLER et al.,2022:7)

$$AHD(A, B) = max(d(A, B), d(B, A))$$

Formel 11: Berechnung des größten Abstands in beide Richtungen (AHD) (MÜLLER et al., 2022:7)

### 2.3.3 Segment Anything Model (SAM)

In diesem Kapitel wird mit dem Segment Anything Model (SAM) ein moderner Ansatz in der Bildsegmentierung vorgestellt. Dabei wird gezeigt, wie das Modell aufgebaut ist und welche Besonderheiten SAM ausmachen.

#### Einleitung

Mit dem Segment Anything (SA) Projekt wurde ein Ansatz entwickelt, welcher neben dem Modell selbst einen großen Datensatz[26] für die Bildsegmentierung zur Verfügung stellt. Mit einer Milliarden Masken auf elf Millionen Bildern gilt der SA-1B zum Zeitpunkt der Erscheinung im Jahre 2023 als bisher größter Segmentierungsdatensatz. Auf Grundlage des Datensatzes ist das Modell in der Lage flexibel auf neue Prompts[27] zu reagieren. So kann es im Zero-Shot-Modus auf neue Bildverarbeitungen und Aufgaben übertragen werden. Ein Prompt kann dabei beispielsweise ein Punkt, eine grobe Box, eine Maske oder ein freier Text sein. Ziel des Modells ist es für jeden Prompt eine gültige Segmentierungsmaske zu erzeugen. Auch bei mehrdeutigen Prompts soll mindestens für ein Objekt eine sinnvolle Maske erstellt werden (vgl. KIRILLOV et al., 2023:1-2).

#### Vorgehensweise

SAM arbeitet nach einem einfachen Prinzip. Zuerst analysiert der Image Encoder ein Bild und wandelt es in ein Image Embedding[28] um. Dies wird einmal berechnet und kann danach für viele verschiedene Prompts angewendet werden. Dadurch wird Zeit gespart und somit werden schnelle Ergebnisse ermöglicht. Die Prompts werden durch den Prompt Encoder in eine geeignete Form gebracht und anschließend mit dem Image Embedding kombiniert. Daraus erzeugt der Mask Encoder die Segmentierungsmaske, die das gewünschte Objekt im Bild markiert. Wenn ein Prompt mehrere Bedeutungen haben kann, erzeugt das Modell mehrere Vorschläge für die Nutzenden (vgl. KIRILLOV et al., 2023:1-2).

[26] SA-1B
[27] Ein Prompt ist die Eingabe, die dem Modell sagt, was segmentieren soll. (vgl. KIRILLOV et al., 2023)
[28] Image Embedding dient dazu Bilder im Vektorraum darzustellen, damit ein Modell daraus Muster lernen kann (vgl. GALLAGHER, 2023)

## Architektur

Als nächstes wird die Architektur von SAM im Detail vorgestellt und erläutert. Die drei Hauptkomponenten umfassen den Image Encoder, Prompt Encoder und Mask Decoder. Dabei werden Vision Transformer [29] (ViT) Models angewendet mit speziellen Anpassungen für die Echtzeit-Performance. Im Folgenden werden die Komponenten kurz erklärt:

Image Encoder: Für die Eingabe von hochauflösenden Bildern wird ein leicht angepasster ViT angewendet. Der Image Encoder wird einmal pro Bild ausgeführt und kann vor der Eingabe von Prompts auf das Modell angewendet werden.

Prompt Encoder: Es werden 2 Arten von Prompts unterschieden. Die Sparse Prompts mit Punkten, Boxen und Texten sowie die Dense Prompts, die in Form von Masken vorliegen. Dabei werden Punkte und Boxen durch Positional Encodings[30] beschrieben, die mit gelernten Embeddings für den Prompt Typ kombiniert werden. Freitext wird mit einem Text Encoder in eine geeignete Darstellung umgewandelt. Masken hingegen werden mit Faltungen verarbeitet und danach mit dem Image Embedding verrechnet.

Mask Decoder: Hier wird das Image Embedding, die Prompt Embeddings sowie ein Output Token kombiniert, um die endgültige Maske zu erzeugen. Dafür werden mithilfe von Self-Attention[31] und Cross-Attention[32] die Informationen aus Bild und Prompt mehrmals miteinander verglichen, so dass das Bild erkennen kann welcher Teil relevant ist. Danach wird das Image Embedding hochskaliert und das Modell berechnet Wahrscheinlichkeit von jedem Punkt, ob dieser zum Objekt gehört. Als Ergebnis wird die gewünschte Maske ausgegeben, die das Bild vom Hintergrund trennt.

Insgesamt zeichnet sich die Architektur durch ein effizientes Design aus. Dank dieser hohen Geschwindigkeit wird eine nahtlose und interaktive Nutzung in Echtzeit ermöglicht (vgl. KIRILLOV et al., 2023:5).

---

[29] Ein Vision Transformer (ViT) ist ein Modell mit Attention-Mechanismen für die Bildklassifikation. (vgl. HUO et al., 2023:136)

[30] Positional Encodings geben jedem Element der Eingabesequenz eine Positionsinformation, damit der Transformer die Reihenfolge berücksichtigen kann. (vgl. LEE, 2023

[31] Self Attention vergleicht Informationen innerhalb einer Eingabe. (vgl. AIML.COM, 2025)

[32] Cross Attention verbindet Informationen aus zwei Eingaben. (vgl. AIML.COM, 2025)

## Weiterentwicklung

Obwohl SAM ein neues Modell ist, wurden bereits einige relevante Weiterentwicklungen vorgestellt. Die meisten Entwicklungen beziehen sich auf eine kompaktere Version oder effizientere Architektur, ohne die Performance zu beeinträchtigen. Genau wie im Erkennungskapitel werden in der unteren Tabelle die Weiterentwicklungen kurz vorgestellt.

| Version | Entwickler | Jahr | Innovation | mAP50 | mAPmask[33] |
|---|---|---|---|---|---|
| SAM | KIRILLOV et al. | 2023 | Zero-Shot, prompt based, trained on SA-1B | - | 46.5 % |
| FastSAM | ZHAO et al. | 2023 | fast mask predicition, real-time focus, reduced computational requirements | - | 37,9 % |
| EfficientSAM | XIONG et al. | 2023 | architecture improvements for efficiency, fewer parameters, lower computational cost | - | 46.5 % |
| SAM2 | YUNYANG et al. | 2023 | architecture improvements, video tracking, real-time video processing | - | - |

Tabelle 5: Übersicht zu den SAM Weiterentwicklungen (eigene Darstellung)

[33] mAPmask ist Gleichzusetzen mit mAP50-95, ist jedoch der etablierte Begriff in der Segmentierung.

# 3 Stand der Forschung

Im vorangegangenen Kapitel wurden wichtige Verfahren für die Objekterkennung und Segmentierung sowie die zu Grunde liegenden Neuronalen Netzwerke vorgestellt und erläutert. Im Folgenden Kapitel wird eine Einordung in den Forschungskontext der jeweiligen Verfahren vorgenommen. Ein Teil der Forschung bezieht sich auf die Weiterentwicklung der einzelnen Verfahren und wurde bereits tabellarisch im vorherigen Kapitel aufgearbeitet. Daher wird sich im weiteren Verlauf auf die Einsatzgebiete und deren Entwicklung konzentriert. Zunächst werden die Verfahren der Objekterkennung betrachtet, gefolgt von den Segmentierungsverfahren. Dabei wird eine kurze Einordung vorgenommen, aktuelle Forschungstrends vorgestellt und bestehende Herausforderungen erläutert.

## 3.1 Forschung in der Objekterkennung und Segmentierung

Die Forschung in der Objekterkennung wird grundsätzlich in zwei zentrale Phasen eingeteilt. Die „*traditionelle Objekterkennung*“ umfasst alle Ideen und Verfahren vor 2014. Alles danach wird als „*deep learning Objekterkennung*“ bezeichnet. Die Entwicklung von R-CNN bildet dabei die Grundlage für diese Unterscheidung. Die ersten Algorithmen basierten noch auf handgefertigten Merkmalen. Es musste also dem Computer genau vorgeschrieben werden, worauf dieser zu achten hatte. Da es damals noch nicht möglich war, Bilder automatisch zu repräsentieren mussten die Forschenden kreative Merkmalsdarstellungen und Beschleunigungstaktiken entwickeln. Heute werden diese Aufgaben von einem neuronalen Netzwerk erfüllt. Zu den wichtigsten Verfahren jener Zeit zählen der Viola Jones Detector, HOG Detector und das Deformable Part-based Model (DPM) (vgl. ZOU et al., 2023:2). Bei der Betrachtung von Deep Learning Erkennungsmodellen muss berücksichtigt werden, dass hier eine Unterscheidung in zwei Gruppen vorliegt. Zum einen gibt es die zweistufigen Verfahren, die Objektvorschläge generieren, bevor die Erkennung erfolgt, dazu zählt unter anderem R-CNN. Zum anderen die einstufigen Verfahren, die ohne Regionenvorschläge arbeiten und die Objekterkennung in einem Schritt durchführen, wie YOLO, SSD und RetinaNet (vgl. LIU et al., 2020).

Die in dieser Arbeit vorgestellten Verfahren kommen in den unterschiedlichsten Bereichen der Objekterkennung zum Einsatz. Eine vollständige Übersicht zu allen Einsatzfeldern würde jedoch den Rahmen dieser Arbeit überschreiten, daher wird sich auf den Einsatz in der Kartografie fokussiert. Es werden im Folgenden relevante Fachartikel vorgestellt, die einen Überblick zum aktuellen Forschungsstand geben.

In einer Studie von SCHNÜRER et al. (2020) werden Karten mithilfe von CNNs analysiert. Sie entwickelten ein System, welches Karten von anderen Bildern unterscheidet und Karten mit Objekten herausfiltert. Ebenso wurden 3200 Segelschiffe in historischen Karten annotiert und mit Faster R-CNN sowie RetinaNet detektiert. Sie erreichten auf COCO einen mAP50-95 Wert von 32,26 % und 36,24 %. Die Forschung betont die Relevanz für das Durchsuchen von Kartenbildern im Internet sowie für die Verbesserung der erweiterten Suche in Kartenkatalogen. Ein weiterer Ansatz untersucht die Klassifikation, Erkennung und semantische Segmentierung von geografischen Merkmalen in Karten. Dabei wird ein RetinaNet Modell verwendet. Ziel ist es Karteninhalte effektiv zu analysieren und für GIS, intelligentes Kartieren sowie für das Kartenverständnis nutzbar zu machen. Die Studie zeigt, dass Deep Learning Verfahren eine vielversprechende Grundlage für die intelligente Kartenerkennung bieten und manuelle Methoden nicht mehr den Anforderungen entsprechen (vgl. WANG et al., 2023). Ein anderes Verfahren, welches mit RetinaNet arbeitet, wurde in der Studie von KEIXUAN et al. (2024) vorgestellt. Sie untersuchen, inwieweit Verwaltungsregionen in Karten automatisch erkannt werden können. Dafür wird ein RetinaNet Modell als Multi-Target- und Cascading-Variante getestet. Bei ersterem werden alle Regionen gleichzeitig erkannt. Bei der zweiten Methode werden die Ergebnisse der Untersuchungsgebiete Taiwan, Tibet, Henan sowie chinesisches Festland einzeln erfasst und kombiniert. Die Ergebnisse zeigen, dass die Cascading Methode zuverlässiger und genauer arbeitet. Ebenso kann geschlussfolgert werden, dass KI-gestützte Objekterkennung die automatisierte Auswertung von Karten erleichtert. Vor allem im Hinblick auf die Kartenzensur und das automatisierte Lesen von Karten zeigt sich der Nutzen. Neben der Erkennung von Regionen rückt in der aktuellen Forschung zunehmend auch die automatische Erkennung von Symbolen in den Fokus. Eine Studie von ZHANG et al. (2022) befasst sich unter anderem mit der autonomen Kartierung und inwieweit die Effizienz bei der Erkennung und

Konfiguration von Punktsymbolen verbessert werden kann. Dafür wurde ein Erkennungsmethode entwickelt, die auf YOLOv3 basiert und mit einem Convolutional Block Attention Module (CBAM) erweitert wurde. Die Symbole werden dabei automatisch mit POI-Daten verknüpft, wodurch eine effiziente Kartenkonfiguration ermöglicht wird. Dieser Ansatz zeigt Potenzial für Anwendungen in der autonomen Kartierung. Auch die Studie von CAO et al. (2025) beschäftigt sich mit der Erkennung und Extraktion von Piktogrammen. Dabei werden digitale und papierbasierte touristische Karten nach den fünf Hauptkategorien Kulturlandschaft, Naturlandschaft, Menschen, Tiere und kulturelle Elemente annotiert. Verschiedene YOLO-Versionen wurden dann anhand des Datensatzes systematisch evaluiert. Mit einem Benutzerinteraktionsmechanismus wurden als nächstes die Ergebnisse überprüft und verfeinert. Anschließend wird SAM verwendet, um die Piktogramme zu extrahieren. Das Ziel ist es die Symbol-Datenbank zu erweitern und so eine Grundlage für die Symbolgestaltung und Kartenproduktion zu bieten. Mit einer durchschnittlichen Erkennungsgenauigkeit von 94,4 Prozent hat YOLOv8 zu den besten Ergebnissen geführt und die Anforderungen erfüllt. In Kombination mit SAM wurde eine effektive und robuste Lösung für die automatisierte Symbol-Extraktion in Karten vorgestellt. Der Ansatz liefert theoretische und praktische Erkenntnisse für die Steigerung der Effizienz in der Kartografie.

## 3.2 Herausforderungen und Relevanz der eigenen Arbeit

Insgesamt haben sich verschiedene Trends in der Erkennung und Segmentierung herausgestellt. An den Weiterentwicklungen der einzelnen Verfahren ist zu erkennen, dass sich vor allem auf die Effizienz und Schnelligkeit konzentriert wird. Dabei ist das Ziel die Verfahren möglichst zu vereinfachen und gleichzeitig keine Performance einzubüßen. Ein anderer Ansatz, wie der von CAO et al. (2025) stellt einen neuen Trend zur Kombination aus Erkennungs- und Extraktionsverfahren dar. Die Verbesserung der Erkennung kleiner Objekte ist dabei eine zentrale Herausforderung und spielt insbesondere in der Weiterentwicklung von YOLO eine wichtige Rolle. Details können beim Downsampling verloren gehen, wodurch nur noch wenige Informationen auf den Pixeln verbleiben. Des Weiteren erschwert die Vielfalt an Objekten und deren unterschiedliche Größe, Form sowie Stil die Generalisierung von Modellen. Auch die Qualität der Karten ist entscheidend. Vor allem bei historischen Karten kann der Scan in Bezug auf Verzerrungen

oder Beschädigungen die Aussagekraft eines Modells beeinflussen. Eine weitere Herausforderung besteht in der Überlappung von Objekten, wodurch eine genaue Detektion erschwert wird.

Vor diesem Hintergrund liegt die Relevanz dieser Arbeit darin, diese bestehenden Verfahren systematisch miteinander zu vergleichen und auf Karten anzuwenden. So kann eine Aussage zur praktischen Effektivität der Verfahren unter verschiedenen Bedingungen getroffen werden. Dafür wird ein Ansatz entwickelt, welcher im folgenden Kapitel beschrieben wird. Damit leistet die Arbeit einen Beitrag zur Unterstützung zukünftiger Entwicklungen in der Computer Vision mit Blick auf die Kartografie.

# 4 Konzeption und Methodik des Ansatzes

In diesem Kapitel werden das Konzept und die Methodik des entwickelten Ansatzes zur automatischen Erkennung sowie Extraktion vorgestellt. Aufbauend auf die theoretischen Grundlagen und den Forschungsstand werden zunächst die Zielsetzung sowie Anforderungen definiert. Anschließend erfolgt eine Beschreibung zur Systemarchitektur und zur technischen Umsetzung der einzelnen Komponenten.

## 4.1 Ziel und Anforderungen

Ziel des Ansatzes ist eine präzise und effiziente Erkennung sowie Segmentierung von Objekten in Karten. Der Ansatz wird im Folgenden als System umgesetzt und beschrieben. Dabei werden die verschiedenen angewendeten Verfahren einmal für einfache Symbole getestet und im nächsten Schritt für detaillierte Piktogramme. Diese können dann für andere Anwendungen weiterverarbeitet werden. Gleichzeitig wird durch die kombinierte Anwendung von Erkennung und Segmentierung insbesondere in der Kartografie ein neuartiger Ansatz verfolgt. Die methodische Umsetzung orientiert sich dabei an dem Ansatz von CAO et al. (2025). Im Fokus steht unter anderem die Untersuchung, wie sich diese Kombination auf die Qualität der Ergebnisse auswirkt. Des Weiteren werden die Verfahren untereinander verglichen, um den besten Ansatz für kartografische Zwecke herauszuarbeiten.

Um das definierte Ziel zu erreichen, werden folgende Anforderungen an das System gestellt. Es muss die Möglichkeit bieten, die zu analysierenden Karten einzulesen und vorzuverarbeiten. Für die Durchführung der Objekterkennung werden die wichtigsten Deep Learning Modelle herangezogen, darunter Versionen von R-CNN, YOLO, SSD und Retinanet. Im Anschluss an die Erkennung gibt es die Möglichkeit die erzeugten Bounding Boxes an ein Segmentierungsmodell wie SAM zu übergeben. Ebenso ist eine Speicherung der Ergebnisse nötig. Sowohl das Modell als auch die Karten mit den Bounding Boxes werden dann in einem geeigneten Format abgespeichert. Zuletzt findet eine Auswertung der Ergebnisse nach den etablierten Bewertungsmetriken statt.

Damit die Ergebnisse nachvollziehbar und überprüfbar sind, muss das System reproduzierbar sein. Dafür werden festgelegte Trainings- und Validierungssplits sowie

eine konsistente Datensatzstruktur angewendet. So wird sichergestellt, dass die Resultate aus allen Verfahren bei erneuter Ausführung unter gleichen Bedingungen identisch ausfallen. Zufallsprozesse wie die Initialisierung der Gewichte oder die Datenreihenfolge können so keinen Einfluss auf die Vergleichbarkeit der Modelle haben. Ebenso sollten die vorhandenen Ressourcen in Google Colab effizient genutzt werden. Eine angemessene Batchgröße, optimiertes Laden oder die Vorverarbeitung der Daten können die Effizienz beeinflussen. Insgesamt sollte ein ausgeglichenes Verhältnis zwischen Laufzeit und Genauigkeit bestehen. Ziel ist daher eine praxisnahe und ressourcenschonende Ausführung des Systems. Für die Nachvollziehbarkeit ist ebenso eine transparente Dokumentation aller Implementierungsschritte notwendig. Dies umfasst die Datenvorverarbeitung, Modellarchitektur, alle relevanten Parameter sowie Trainings und Evaluationsprozesse.

## 4.2 Systemarchitektur

Die Systemarchitektur bildet die Grundlage für den entwickelten Ansatz. Sie umfasst alle funktionalen Modulabschnitte, die für die Umsetzung des Systems relevant sind. Es wird in die Module Datenvorverarbeitung, Objekterkennung, Objektextraktion und Modellbewertung eingeteilt. Das Ziel der Architektur ist es eine modulare, reproduzierbare und effiziente Umsetzung zu gewährleisten, sodass eine leicht verständliche Nutzung und Anpassung des Systems ermöglicht wird.

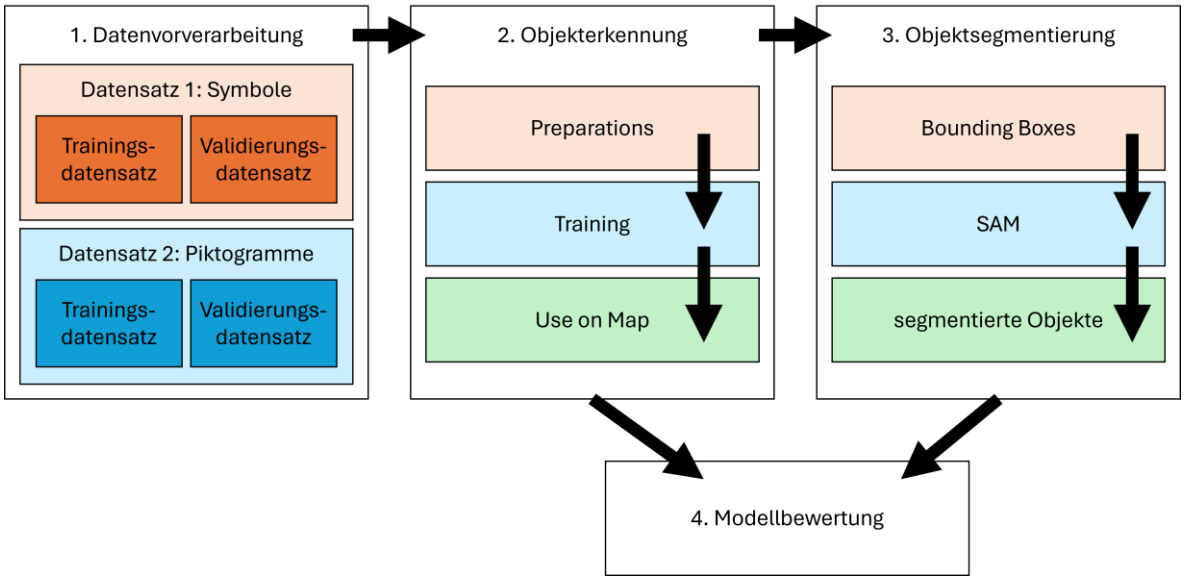



Abbildung 7: Systemarchitektur (eigene Darstellung)

## 1. Datenvorverarbeitung

Nachdem das Kartenmaterial zur Verfügung steht, werden die Daten für das weitere Vorgehen vorbereitet. Dafür werden zunächst die relevantesten Symbole oder Piktogramme identifiziert und manuell mit Ground Truth Bounding Boxes annotiert. So können die Modelle während des Trainings die exakte Position und Klasse der Objekte lernen. Anschließend werden die Daten in Trainings- und Validierungsdatensätze aufgeteilt, um eine konsistente und vergleichbare Auswertung der Modelle sicherzustellen. Aufgrund der Größe der Kartenbilder und um den Trainingsdatensatz zu vergrößern, werden im nächsten Schritt die Karten in kleinere Tiles aufgeteilt. Ergänzend dazu können im Training Datenaugmentierungen, Ankeranpassungen oder Gewichtungen vorgenommen werden.

## 2. Objekterkennung

Die Objekterkennung bildet den zentralen Bestandteil des Systems. Hier werden verschiedene Deep Learning Modelle implementiert und angewendet. Zum Einsatz kommen dabei die Verfahren YOLO, Faster R-CNN, SSD und Retinanet. Diese können mit verschiedenen Parametern angepasst werden, um für den individuellen Datensatz das beste Ergebnis zu erzielen. Insgesamt ist der Erkennungsteil für jedes Verfahren in die drei Abschnitte Preparations, Training und Use on Map unterteilt. Im ersten Abschnitt erfolgen notwendige Vorbereitungen. Dazu zählen der Import der erforderlichen Bibliotheken und die Installation des jeweiligen Verfahrens. Im Trainingsteil wird das Training initialisiert und zugehörige Funktionen definiert. Darüber hinaus besteht die Möglichkeit, das trainierte Modell zum Testen auf ein Bild anzuwenden. Der letzte Abschnitt umfasst einen Ablauf, der speziell auf das Training großflächiger Karten ausgerichtet ist. Hier wird die Eingabekarte zunächst in Tiles aufgeteilt, deren Größe den Tiles der Trainingsdaten entspricht. Anschließend wird das Modell auf jedem Tile einzeln angewendet. Die trainierten Tiles werden dann wieder zu einer vollständigen Karte zusammengeführt. Für jeden Schritt werden die erstellten Daten in einer zugehörigen Ordnerstruktur abgespeichert. Die angewendeten Daten liegen im YOLO-Format vor und müssen für die anderen Verfahren in Pixelkoordinaten umgewandelt werden. Dafür dient der Zwischenabschnitt Setting Up.

### 3. Objektsegmentierung

Um die Objektsegmentierung anzuwenden, werden im ersten Schritt alle zuvor erstellten Bounding Boxes des verwendeten Verfahrens als Bild abgespeichert. Jede Box begrenzt damit den Bereich, innerhalb dessen die Segmentierung durchgeführt wird. Danach kann SAM auf diesen Bildern verwendet werden. Ein weiterer Ansatz zur Segmentierung basiert auf einfachen Bildverarbeitungstechniken, welcher im letzten Schritt ebenfalls genutzt werden kann. Die segmentierten Symbole oder Piktogramme werden dann ausgegeben und können für nachfolgende Analysen oder Anwendungen bereitgestellt werden. Die segmentierten Bilder werden automatisch als PNG abgespeichrt, können jedoch auch in ein SVG umgewandelt werden.

### 4. Modellbewertung

Im abschließenden Abschnitt in einer weiteren Datei werden etablierte Bewertungsmetriken herangezogen, um die Erkennungs- und Segmentierungsleistung auszuwerten. Hier werden die in Kapitel 2.2.1 und 2.3.2 vorgestellten Bewertungsmetriken angewendet. Ebenso werden qualitative Vergleiche zwischen den Modellen durchgeführt, um Unterschiede erkennen und interpretieren zu können.

Insegesamt wurde die Architektur so konzipiert, dass sie vollkommen modular in Google Colab ausgeführt werden kann. Alle Schritte können nach der Reihe ausgeführt werden und enthalten kurze Kommentare zu den Funktionen des Codes. Dadurch lässt sich der gesamte Prozess leicht nachvollziehen und erweitern. Ebenso können nach dem selben Prinzip andere Datensätze angewendet werden. Insgesamt ermöglicht die Systemarchitektur eine strukturierte und vergleichbare Untersuchung zu verschiedenen Machine Learning Verfahren zur automatischen Erkennung und Segmentierung von Objekten in Karten.

## 4.3 Methodische und technische Umsetzung

In den vorherigen Unterkapiteln wurde das Ziel und die Systemarchitektur beschrieben. Im Folgenden wird die methodische und technische Umsetzung erläutert. Es werden die verwendete Entwicklungsumgebung, Datenstruktur, Methodik und der technische

Aufbau aufgezeigt. Die detaillierte Beschreibung der Implementierung und Ausführung der Modelle erfolgt dann im nächsten Kapitel.

## Entwicklungsumgebung

Für die Umsetzung des entwickelten Ansatzes wird die cloudbasierte Umgebung Google Colab verwendet. Die Umgebung ermöglicht eine einfache Integration von Python Bibliotheken und stellt gleichzeitig eine GPU-Unterstützung zur Verfügung. Durch die plattformunabhängige Nutzung wird das System reproduzierbar und das Projekt kann leicht weitergegeben werden. Grundsätzlich bietet Google Colab eine kostenlose GPU-Nutzung. Im Rahmen dieser Arbeit wurde zusätzlich Colab Pro verwendet, um erweiterte Recheneinheiten und längere Laufzeiten zu ermöglichen. Die Implementierung des Systems erfolgt in Python 3.12.12, wobei eine NVIDEA Tesla T4 GPU und der Cuda Version 12.4 benutzt wurde. Die Beschleunigung durch die GPU in Colab gewährleistet eine effiziente Durchführung der Trainings- und Auswertungsprozesse, ohne dass lokale Hochleistungshardware benötigt wird. Es werden zwei voneinander gtrennte Colab Dateien erstellt. Eine beinhaltet die Anwendung der Modelle und die andere den Code für die Auswertung.

Für die Umsetzung in Python wurden verschiedene Bibliotheken verwendet. Pytorch diente zur Implementierung der Objekterkennungsmodelle, wobei die wichtigsten Importe torch und torchvision genutzt wurden. Für die Datenverarbeitung und numerische Berechnungen wurde NumPy angewendet, während Pillow für das Bearbeiten und Laden der Bilddaten verwendet wurde. Um Ergebnisse zu visualisieren kam Matplotlib zum Einsatz. Die Segment Anything Bibliothek wurde benötigt zur Implementierung des gleichnamigen Modells zur Objektsegmentierung. Die Kombination aus diesen Bibliotheken ermöglicht modulare und flexible Anpassungen im System. Insgesamt basiert die Wahl der Umgebungskomponenten auf einer guten Dokumentation, einfachen Reproduzierbarkeit sowie effizienten Nutzung von Ressourcen. Dadurch lässt sich das komplette System einfach teilen, erweitern und auf andere Datensätze übertragen.

## Datenstruktur

Die Datenverwaltung folgt einer einheitlichen und reproduzierbaren Struktur. Die Datensätze sind in Trainings- und Validierungsdaten aufgeteilt. Ebenso gibt es für den zweiten Datensatz eine unabhängige Testkarte, um die Funktionsfähigkeit der Modelle zu überprüfen. Diese einheitliche Struktur wird auf alle Modelle gleich angewendet und ermöglicht so eine Vergleichbarkeit der Ergebnisse. Die Annotationen liegen im YOLO-Format vor, werden jedoch für die Verwendung der anderen Modelle in kompatible Pixelkoordinaten umgewandelt. Des Weiteren wird eine standardisierte Ordnerstruktur und eine einheitliche Benennung der Daten angewendet, um die Nachvollziehbarkeit zu gewährleisten.

## Methodik

Die gesamte Methodik ist nach einem klar definierten und modularen Ablauf gegliedert, welcher eine strukturierte Durchführung ermöglicht. Nachdem die Karten vorbereitet wurden, werden sie in den Trainingsprozess überführt. Die Objekterkennungsverfahren werden dann auf den annotierten Bildern trainiert und validiert. Anschließend kann das trainierte Modell auf weiteren Bildern angewendet werden. Die erkannten Objekte werden dann als Bounding Box ausgegeben und im Anschluss an das Segmentierungsmodul übergeben. Zur Abgrenzung der Symbole und Piktogramme vom Hintergrund werden SAM oder Bildverarbeitungsverfahren eingesetzt. Zum Schluss werden die zuvor definierten Bewertungsmetriken angewendet, um eine Vergleichbarkeit der Verfahren sicherzustellen. Durch die Struktur des Verfahrens wird ein konsistenter und effizienter Prozess für die Objekterkennung und Extraktion in Karten zur Verfügung gestellt.

## Technischer Aufbau

Die Module Objekterkennung und Segmentierung sind in einem Notebook integriert. Für die Auswertung der Modelle wurde ein weiteres Notebook eingerichtet. Sie sind in ausklappbare Überschriften nach Thema unterteilt. So können die Codeblöcke für jeden Abschnitt einzeln aufgerufen werden. Diese Abschnitte sind dann nach Funktion in weitere Unterabschnitte eingeteilt. Lediglich die Datenvorverarbeitung wurde in separaten Dateien vorgenommen. Jedes Modell kann dabei unabhängig voneinander

trainiert und ausgeführt werden. Dadurch können einzelne Komponenten gezielt bearbeitet werden, ohne das gesamte System zu verändern. Die Codeblöcke sind thematisch geordnet und enthalten kurze Kommentare zum verwendeten Code, sodass dieser nachvollzogen werden kann. Dadurch lassen sich Modelle unter den gleichen Bedingungen erneut erzeugen. Außerdem werden die Parametereinstellungen konsistent über alle Modelle gleich verwendet.

# 5 Implementierung

In diesem Kapitel wird die praktische Umsetzung des zuvor konzipierten Systems im Detail vorgestellt. Die definierten Module werden hier auf Grundlage der beschriebenen Systemarchitektur implementiert. Zunächst wird die zugrunde liegende Datenbasis vorgestellt, gefolgt von der Durchführung des Trainings und der anschließenden Segmentierung.

## 5.1 Vorbereitung der Daten

Für die Bearbeitung des Ansatzes wurden zwei voneinander getrennte Datensätze mit digitalisierten Karten verwendet. Diese wurden eigens für die Untersuchung erstellt und dienen sowohl fürs Training als auch für die Evaluation. Die Datengrundlage umfasst zum einen Karten mit einfachen Symbolen und zum anderen Karten mit aufwendig gestalteten Piktogrammen. Beide Datensätze wurden jedoch nach dem selben Prinzip erstellt. Nachdem die Karten zur Verfügung stehen, wurden sie zunächst in 1024x1024 Tiles zerschnitten. Diese Aufteilung wurde vorgenommen, da großflächige Karten für die Objekterkennung ungeeignet sind und die Modelle auf kleineren Bildern besser arbeiten. Als nächstes wurden die Annotationen der zu erkennenden Objekte aufbereitet. Sie dienen als Ground Truth und wurden mit dem Tool makesense.ai erzeugt. Hier wurden als erstes die Karten hineingeladen und alle Klassen für die Annotationen festgelegt. Dann wurden die Ground Truth Bounding Boxes für jedes Tile manuell erstellt. Sobald die gewünschten Objekte annotiert wurden, kann das Projekt im YOLO-Format heruntergeladen werden. Die Bilder und Annotationen wurden dann für die beiden Datensätze jeweils in Trainings- und Validerungsdaten aufgeteilt. Tiles ohne Annotationen wurden dann entfernt. Diese aufbereiteten Daten können als ZIP-Datei in das Colab Projekt geladen werden, um die Verfahren auf die Daten anzuwenden.

### Datensatz 1: einfache Symbole

Für die Erstellung des ersten Datensatzes wurden Karten aus dem Datenbestand des Geomedien-Labors der BHT herangezogen. Der Bestand umfasst Karten aus verschiedenen Regionen der Welt. Für die Untersuchung im Rahmen dieser Arbeit wurden daraus geeignete Karten ausgewählt. Dabei wurden insgesamt 12 Karten für die Analyse

angewendet. Diese wurden dann in 10 Trainings- und 2 Validierungskarten aufgeteilt. Es handelt sich um topografische Karten aus dem Kartenwerk Topografische Karte 1:10 000. Die verfügbaren Karten wurden 1993 bis 1995 in der ersten Auflage vom Landesvermessungsamt Brandenburg herausgegeben und beinhalten ausschließlich Regionen aus Brandenburg. Des Weiteren besitzen alle Karten eine Gauß-Krüger Projektion. Dargestellt werden detallierte topografische Darstellungen mit standardisierten Symbolen. Die Folgenden Symbole haben sich als besonders relevant herausgestellt und werden mit der Anzahl an annotierten Labels dargestellt.

| Nummer | Klasse | Annotationen | Beispielbild |
|---|---|---|---|
| Klasse 0 | Mischwald | 36 Labels |  |
| Klasse 1 | Bodenpunkt | 87 Labels |  |
| Klasse 2 | Nadelwald | 310 Labels |  |
| Klasse 3 | Gebüsch | 230 Labels |  |
| Klasse 4 | Schilf | 850 Labels |  |
| Klasse 5 | Laubwald | 75 Labels |  |

Tabelle 6: Inhaltsdarstellung Datensatz 1 (eigene Darstellung, Daten basierend auf GEOMEDIENLABOR)

Ebenfalls annotiert wurden die Symbole Kirche und Denkmal. Da für diese beiden Symbole allerdings nur sehr wenige Labels gefunden wurden, werden sie bewusst vor dem Training herausgenommen, um eine zu starke Verzerrungen durch wenige Labels zu vermeiden.

## Datensatz 2: Piktogramme

Dieser Datensatz setzt sich aus unterschiedlichen historischen Karten zusammen. Es handelt sich dabei um thematische Karten, die mit zahlreichen verschiedenen Piktogrammen arbeiten. Insgesamt wurden 25 Karten herausgesucht, davon 20 Karten aus der Quelle The Big Ten Academic Allience und 5 aus The Library of Congress. Die Mehrheit der Karten stellt Regionen in Amerika dar, einzelne Karten zeigen jedoch auch

andere Teile der Welt. Hier liegt der Fokus nicht darauf identische Objekte zu finden, sondern vielmehr das Modell so zu trainieren, dass Objekte der selben Gruppe zugeordnet werden können ohne exakt gleich zu sein. Nach CAO et al. (2025) haben sich die Kategorien Human Landscape, Natural Scenery, Human, Animal und Culture für touristische Karten als besonders geeignet erwiesen[34]. Da der Datensatz aus historischen Karten eine ähnliche vielfalt an Piktogrammen aufweist, dienen diese Kategorien auch hier als Grundlage. Es wurde lediglich die Kategorie Vehicle hinzugefügt, da Transportmittel in den historischen Karten häufig auftreten und in keine der bestehenden Klassen passen. Da die Karten jedoch unterschiedliche Formate besitzen, wurde auch bei diesen Daten eine Tilegröße von 1024x1024 genutzt. Sollte ein zu kleines Tile diesem Format nicht entsprechen, wurde es durch Padding[35] automatisch auf die passende Größe erweitert. Der Datensatz wurde auch hier in Training und Validierung eingeteilt. Da jedoch alle Piktogramme unterschiedlich aussehen und jede Karte einen eigenen Stil hat, können nicht mehr vollständige Karten für das Training und die Validierung genutzt werden. Stattdessen wurden die Tiles jeder Karte zufällig in 20% Validierung und 80% Training aufgeteilt und am Ende zu einem Datensatz zusammengefügt. Die Tabelle 7 zeigt Beispielhaft die Klassenverteilung.

| Nummer | Klasse | Annotationen | Beispielbild |
|---|---|---|---|
| Klasse 0 | Human Landscape | 193 Labels |  |
| Klasse 1 | Natural Scenery | 196 Labels |  |
| Klasse 2 | Human | 1276 Labels |  |
| Klasse 3 | Animal | 811 Labels |  |

[34] Für einen besseren Vergleich wurden die englische Bezeichnungen beibehalten.
[35] Padding fügt bei zu kleinen Tiles Pixel hinzu, sodass alle die gleiche Größe haben.

| | | | |
|---|---|---|---|
| Klasse 4 | Culture | 197 Labels |  |
| Klasse 5 | Vehicle | 340 Labels |  |

Tabelle 7: Inhaltsdarstellung Datensatz 2 (eigene Darstellung, Daten basierend BIG TEN ACADEMIC ALLIANCE & LIBRARY OF CONGRESS)

## 5.2 Modellversionen

In diesem Unterkapitel wird die Implementierung des Ansatzes erläutert. Die zuvor besprochenen Datensätze werden dann auf die daraus erstellten Modelle angewendet und im nächsten Schritt ausgewertet. Im Zuge der Objekterkennung und Objektsegmentierung wurden unterschiedliche Modelle verwendet, die sich in ihrer Architektur zum Teil stark unterscheiden. Alle Modelle wurden mithilfe des PyTorch Framework in Colab implementiert. Im Folgenden werden die angewendeten Versionen und deren Implementierung erläutert:

**YOLOv5**

Die Implementierung von YOLOv5 erfolgt über die offizielle Version von Ultralytics. Es wurde die Variante YOLOv5s verwendet, diese kann jedoch nach Bedarf ausgetauscht werden. Dabei handelt es sich um die kleinste Modellvariante. Sie bietet eine gute Balance zwischen Genauigkeit und Rechenaufwand bei begrenzten Hardwarekapazitäten. Um das Training auf den eigenen Daten zu beschleunigen, dienen vortrainierte Gewichte, die auf dem COCO-Datensatz trainiert wurden. Dadurch kann das Modell schneller bestimmte Kanten und Merkmale lernen. YOLOv5 zeichnet sich aus durch eine Kombination aus Geschwindigkeit, Genauigkeit und Handhabung. Es befindet sich seit 2020 in Entwicklung und wird durch die Community oder Ultralytics kontinuierlich optimiert. Dadurch ist das Modell ein stabile und zuverlässige Option für Anwendungen in der Objekterkennung, Bildsegmentierung oder Bildklassifizierung (vgl. JOCHER, 2020). Die Klassenanzahl wurde durch eine manuell erstellt YAML-Datei festgelegt.

### YOLOv8

Für einen Vergleich von zwei verschiedenen YOLO-Varianten, wurde ebenfalls YOLOv8 implementiert. Dieses Modell wurde 2023 veröffentlicht und bietet eine gute Kombination in Bezug auf Genauigkeit, Geschwindigkeit und Vielseitigkeit. Es verfügt über weitere Funktionen und Optimierungen im Vergleich zu älteren Varianten. Beispielsweise besitzt YOLOv8 modernere Backbone- und Neckarchitekturen. Im Vergleich zu neueren YOLO-Versionen gibt es umfangreiche Dokumentationen und Community Beiträge (vgl. JOCHER et al.,2023a). Bei einem Vergleich der Modelle YOLO11 und YOLOv8 in der Dokumentation von Ultralytics wird zwar für die bestmögliche Leistung bei neuen Projekten YOLO11 empfohlen, es wird aber auch die Zuverlässigkeit von YOLOv8 betont. Letztendlich hängt die Wahl von den spezifischen Projektanforderungen ab (vgl. JOCHER, 2024b). Aufgrund des größeren Ecosystems rund um YOLOv8 und da es weiterhin als sehr gutes Modell gilt, wurde sich im Rahmen der Arbeit für die Implementierung von YOLOv8 entschieden. Auch für dieses Modell wird eine YAML-Datei benötigt.

### Faster R-CNN

Faster R-CNN wurde 2016 von REN et al. veröffentlicht und gilt als Nachfolger von Fast R-CNN. Als Backbone wurde dabei ein ResNet-50 in Kombination mit einem Feature Pyramid Network (FPN) als Neck verwendet. Das ResNet-50 dient zur Extraktion von hierarchischen Merkmalen aus dem Eingabebild (vgl. HE et al., 2016). Das FPN hingegen ermöglicht die Nutzung von Merkmalskarten mit unterschiedlicher Auflösung, um Objekte mit unterschiedlichen Größen zu erkennen (vgl. LIN, 2017b). Die Kombination aus Faster R-CNN und ResNet-50-FPN als Backbone wird in vielen Studien eingesetzt und ist in der Forschung weit verbreitet. Auch hier werden vortrainierte Gewichte auf dem COCO Datensatz initialisiert, um den Prozess zu beschleunigen und die Leistung zu verbessern. Da 9 Klassen annotiert wurden, jedoch nur 7 relevant sind, wurde die Schicht zur Klassifizierung dementsprechend angepasst. Die Klassenanzahl setzt sich zusammen aus den 6 annotierten Klassen und dem Hintergrund.

## SSD

Das SSD-Modell wurde 2016 von LIU et al. entwickelt und dient in dieser Arbeit als weiteres Vergleichsmodell. Die Implementierung in PyTorch arbeitet mit dem Namen SSD300_VGG16. Das Modell ist auf eine Standardeingabe von Bildern mit 300x300 Pixeln ausgelegt. Als Backbone wird dabei das VGG-16-Netzwerk verwendet, welches als Standard für die Implementierung von SSD gilt. Des Weiteren gibt es die SSD512 Variante, welche jedoch nicht in PyTorch sondern nur in Caffe implementiert ist (vgl. LIU et al., 2016). Sollte für ein Projekt eine andere Eingangsauflösung oder ein anderer Backbone benötigt werden, sind eigene Anpassungen oder Implementierungen des Modells notwendig. Die Klassen wurden auch hier auf 7 angepasst. Ebenso wurden zur Initialisierung trainierte COCO-Gewichte verwendet.

## RetinaNet

RetinaNet wurde 2017a von LIN et al. vorgestellt. Es wurde ebenfalls mit einem ResNet-50 Backbone und einem FPN Neck implementiert. Genauso wurden vortrainierte COCO Gewichtungen geladen und die Klassen auf 7 angepasst. Zuvor wurden die unbedeutenden Klassen durch einen weiteren Codeblock aus dem Datensatz entfernt. Diese Anpassung war nötig, da die Verwendung von mehreren Klassen bei RetinaNet in der Implementierung zu Kompatibilitätsproblemen führte. Die anderen Modelle passen ihre Klassen bei der Implementierung automatisch an, sodass die letzten beiden Klassen herausgenommen werden. Die Focal Loss Werte wurden entsprechend der empfohlenen Standardparameter von LIN et al. (2017a) mit $Y = 2.0$ und $\alpha = 0.25$ konfiguriert, damit unterrepräsentierte Klassen eine stärkere Gewichtung erhalten.

## SAM

2023 haben KIRILLOV et al. bei Meta AI Reasearch SAM zur Objektsegmentierung vorgestellt. Dieser Ansatz gehört mit zu den modernsten Bildsegmentierungstools. SAM basiert auf einer ViT-Architektur und wurde auf dem Datensatz SA-1B mit über 1 Milliarden Segmentierungen trainiert (vgl. KIRILLOV et al, 2023). Für die Implementierung wurde das offizielle GitHub-Repository installiert. Dabei wurde das ViT-H-Backbone[36] geladen, um

[36] b (base), l (large), h (huge) (vgl. JOCHER et al., 2023c)

das vortrainierte Modell direkt ohne Training verwenden zu können. Eine weitere Möglichkeit wäre der Einsatz von SAM 2. Da dieses Modell jedoch hauptsächlich für die Verarbeitung von Videos und sehr großen Datensätzen optimiert wurde, bietet es für den entwickelten Ansatz keinen signifikanten Vorteil gegenüber SAM 1 (vgl. RAVI et al., 2024).

## 5.3 Trainingsumgebung

Nachdem die Implementierung der Verfahren vorgenommen wurde, wird im nächsten Schritt die Trainingsumgebung eingerichtet. Das Training ist notwendig, um die vortrainierten Daten an die beiden spezifischen Datensätze dieser Arbeit anzupassen. Alle angewendeten Objekterkennungsmodelle nutzen vortrainierte Gewichte auf dem COCO-Datensatz. SAM hingegen wurde auf dem SA-1B-Datensatz vortrainiert. Dies erleichtert das Training, da das Modell nicht von Grund auf neu trainiert werden muss. So sind die Modelle bereits in der Lage grundlegende Formen zu erkennen und lernen während des Trainings gezielt sich auf die speziellen Anforderungen der neuen Daten zu konzentrieren. Für einen Vergleich zwischen den Modellen und einen einheitlichen Trainingsprozess wurden die folgenden Trainingsparameter angewendet. Diese sind für alle Modelle gleich und können je nach Anwendung beliebig angepasst werden.

Zunächst wurden die Optimierungseinstellungen angepasst. Sie dienen dazu das Lernverhalten eines Neuronalen Netzes zu steuern. Dabei werden die Gewichte und Lernraten angepasst, um die Verlustfunktion zu minimieren. Für diese Arbeit wurde der Stochastic Gradient Descent (SGD) Optimizer verwendet. Hier werden die Gewichte nach jeder einzelnen Trainingsinstanz angepasst (vgl. RAIAAN et al., 2024:11). SGD wurde gewählt, da es trotz langsamerer Geschwindigkeiten im Training meist eine bessere Generalisierung erreicht als adaptive Optimizer wie Adam (vgl. ZHOU et al., 2020). Vor allem bei YOLO passiert die Auswahl des Optimizers standardmäßig automatisch. Je nach Modelltyp und Trainingsframework wird dann meistens AdamW oder SGD angewendet (vgl. JOCHER et al., 2023b). Für eine bessere Vergleichbarkeit wurde jedoch für alle Modelle SGD als Optimizer festgelegt. Die Parameter Lernrate, Momentum und Weight Decay werden durch den Optimizer bestimmt. Als Lernrate wurde eine 0,003 festgelegt, um stabile und moderate Gewichtsupdates zu gewährleisten. Das Momentum von 0,9 beschleunigt die Konvergenz und glättet Schwankungen. Ein Weight Decay von

0,0005[37] kann Overfitting reduzieren und unterstützt die Generalisierung auf neuen Objekten. Die gewählten Werte entsprechen dem Standard in der Objekterkennungscommunity und wurden durch ausprobieren optimiert. Diese Einstellungen ermöglichen ein vergleichbares Training und stellen sicher, dass Leistungsunterschiede nicht auf unterschiedliche Optimierungsstrategien zurückzuführen sind. Ein weiterer wichtiger Parameter ist die Batchgröße. Für den vorhandenen GPU Speicher wäre eine Größe von 4 oder 8 denkbar. Letztendlich wurde sich über alle Modelle hinweg für eine Batchgröße von 4 entschieden, um stabile Updates bei kleinen Datensätzen zu gewährleisten. Jeder Batch entspricht einem Trainingsschritt, in dem der Optimizer die Gewichte des Modells aktualisiert (vgl. TAM, 2023). Die Modelle erreichen dabei meist in sehr unterschiedlicher Dauer eine stabile Konvergenz. Daher wurden als Trainingsdauer 100 Epochen gewählt, um allen Modellen genügend Zeit für das Training zu geben. Ein Early Stopping[38] wurde bewusst nicht eingesetzt, um eine einheitliche Trainingsdauer und somit einen fairen Vergleich zu erzielen. Außerdem wird die Verlustfunktion für die Modelle unterschiedlich berechnet und ist in den Frameworks integriert, daher kann der Loss nicht als Indikator für ein einheitliches Early Stopping genutzt werden. Während des Trainings wird der Loss kontinuierlich überwacht und bei einer Verbesserung das Modell der jeweiligen Epoche abgespeichert. Das vorherige Modell wird dann überschrieben, sodass nur das beste Ergebnis ausgegeben wird. Des Weiteren wurde für einen stabilen Trainingsprozess ein Learning Rate Scheduler eingesetzt. Dieser passt die Lernrate während des Trainings an den Optimierungsprozess an, wodurch die Leistung verbessert und die Trainingszeit verringert wird. Dabei wird zu Beginn des Trainings eine höhere Lernrate gewählt, um die Gewichte grob anzupassen. Für feinere Anpassungen und um Overfitting zu vermeiden, wird dann zum Ende des Trainings die Lernrate schrittweise verkleinert (vgl. TAM, 2023). Für den entwickelten Ansatz wurde ein Learning Rate Scheduler vom Typ StepLR gewählt. Dieser reduziert alle 10 Epochen die Lernrate um den Faktor 0,8. Insgesamt bilden die Parameter eine einheitliche Trainingsgrundlage, die es ermöglicht alle Modelle miteinander zu vergleichen und die Leistungsunterschiede zu bewerten.

[37] 5e-4

[38] Ein Early Stopping sorgt dafür, dass das Training beendet wird, sobald sich der Loss eine bestimmte Anzahl an Epochen nicht mehr verbessert.

## 5.4 Modellanwendung

Nachdem eine große Karte in die 1024x1024 Tiles zerlegt wurde, wird eine Inferenz auf allen Tiles des Ordners ausgeführt. So wird immer auf derselben Eingabegröße trainiert wie die Trainingsdaten. Um nun die trainierten Modelle auf Bilder anzuwenden, werden einheitliche Parameter verwendet. Es wurde für alle Modelle eine Confidence Treshold von 0,2 festgelegt. Dieser sorgt dafür, dass nur Bounding Boxes erstellt werden, bei denen sich das Modell zu mindestens 20 % sicher ist, dass es sich um das klassifizierte Objekt handelt. Ebenso wurde ein IoU Threshold von 0,5 bestimmt. Sollten mehrere Bounding Boxes um ein einziges Objekt erzeugt worden sein, verbindet dieser Threshold mithilfe von NMS alle Bounding Boxes, die sich zu 50% überschneiden zu einer Bounding Box. Die beiden Parameter können jedoch nach Anwendungsfall individuell angepasst werden. Letztendlich sorgen sie dafür, dass Erkennungen von ausreichender Sicherheit und geringer Überschneidung im Ergebnisbild erzeugt werden. Anschließend werden diese Erkennungen in einem Ordner abgespeichert, welcher diese an das implementierte SAM-Modell übergibt.

# 6 Evaluierung

In diesem Kapitel werden die Ergebnisse der zuvor beschriebenen Trainingsprozesse ausgewertet und analysiert. Die in Kapitel 2.2.1 und 2.3.2 erläuterten Bewertungsmetriken werden nun angewendet, um anschließend die Leistungsfähigkeit und Effektivität der implementierten Modelle zu beurteilen. Der Fokus liegt auf dem Vergleich der Modelle sowie der Analyse der Trainingsverläufe. Dabei wird die Leistung auf beiden Datensätzen betrachtet.

## 6.1 Objekterkennung

In diesem Unterkapitel werden die Objekterkennungsmodelle miteinander verglichen. Die Ergebnisse werden dann analysiert und interpretiert. Es werden dabei beide Datensätze mit folgender Struktur dargelegt. Als erstes werden die Standardmetriken betrachtet, gefolgt von zusätzlichen Metriken und den Klassenmetriken sowie dem Trainingsverlauf. Danach wird das beste Modell angewendet und die Ergebnisse ausgewertet.

### 6.1.1 Datensatz 1

**Standardmetriken**

Die Ergebnisse des ersten Datensatzes sind in der unteren Tabelle zusammengefasst. Dargestellt werden die Parameter mAP50, mAP50:95, Precision und Recall für alle 5 Modelle. Sie zeigen den Mittelwert über alle Klassen hinweg und geben einen grundlegenden Einblick zur Leistungsfähigkeit der Modelle. Angewendet wurden die Modellparameter, welche im letzen Kapitel beschrieben wurden.

| Modell | mAP50 | mAP50:95 | Precision | Recall |
|---|---|---|---|---|
| YOLOv8 | 0,8921 | 0,4264 | 0,8057 | 0,9059 |
| YOLOv5 | 0,8887 | 0,4252 | 0,8224 | 0,8465 |
| FR-CNN | 0,5471 | 0,2089 | 0,5788 | 0,8824 |
| RetinaNet | 0,0583 | 0,0207 | 0,0780 | 0,7549 |
| SSD | 0,0531 | 0,0145 | 0,0833 | 0,6422 |

Tabelle 8: Standardmetriken Datensatz 1 (eigene Darstellung)

Die Tabelle zeigt, dass der höchste mAP Wert sowohl bei mAP50 als auch bei mAP50:95 vom YOLOv8 Modell erreicht wurde. Dicht dahinter YOLOv5 mit einer ähnlichen, aber minimal geringeren Leistung. Die Precision ist bei YOLOv5 und Recall bei YOLOv8 leicht höher. Faster R-CNN erreichte eine moderate Leistung, wobei mAP50 knapp 40% und mAP50:95 knapp 50% unter den Werten der YOLO Modelle liegen. Auffallend hierbei ist der Unterschied in Recall und Precision. Obwohl Faster R-CNN einen hohen Recall Wert erreichte, lag die Precision knapp 0,25 unter den entsprechenden YOLO Werten. SSD und RetinaNet erzielten deutlich geringere Werte als die anderen Modelle. Zusammenfassend zeigen die Ergebnisse, dass die YOLO Modelle die höchste Erkennungsleistung unter den betrachteten Modellen erreichten.

Die Ergebnisse lassen sich auf die unterschiedliche Architektur der Modelle zurückführen. Die YOLO Modelle besitzen beispielsweise eine moderne und ankerfreie Struktur sowie tiefe Feature Pyramiden. Dadurch wird eine effektive Erkennung von kleinen und überlappenden Objekten ermöglicht, was zu hohen und stabilen Precision und Recall Werten führte. Die Werte für Faster R-CNN lassen sich daraus schlussfolgern, dass zweistufige Detektoren in zwei Phasen ablaufen. Anders als bei einstufigen Detektoren, welche direkt das gesamte Bild betrachten, werden hier zunächst potenzielle Objektregionen vorgeschlagen und Objekte erst im zweiten Schritt in diesen Gebieten lokalisiert. Dadurch werden zwar viele Objekte erkannt, jedoch erhöht sich auch die Wahrscheinlichkeit, dass bestimmte überlappende oder fehlerhafte Objekte erkannt werden. Diese hohe Anzahl an fehlerhaft erkannten Objekten führt letztendlich zu einer geringeren Precision. Die vielen Regionenvorschläge hingegen sorgen dafür, dass die meisten Objekte auch in den Regionen enthalten sind und nur wenige übersehen werden. Daher hat Faster R-CNN einen vergleichsweise hohen Recall Wert. Die geringen Werte für SSD und RetinaNet lassen sich durch die einfache Architektur und die nicht so tiefen Feature Pyramiden erklären. Dadurch haben die Modelle Probleme einfache und kleine Objekte zu identifizieren. Ein weiterer Grund liegt in der maximalen Eingabeauflösung der Modelle. Während die Daten eine Tilegröße von 1024x1024 Pixeln besitzen, verarbeitet SSD das Eingabebild standardmäßig mit 300x300 Pixeln. Durch die Skalierung in das Eingabeformat kann es passieren, dass kleine und beieinanderliegende Objekte verloren

gehen oder undeutlich werden. In Kombination mit der vergleichsweise einfachen Architektur führt dies zu einer weit geringeren Erkennungsleistung.

## Zusätzliche Metriken

Für eine detaillierte Bewertung der Modelle werden die weiteren Metriken F1-Score F1:50 und F1:50-95[39] sowie Mean IoU herangezogen. Sie bieten einen Einblick in die Genauigkeit der Bounding Boxes und zeigen die Balance zwischen Precision und Recall. Im Folgenden werden diese Bewertungsmetriken in der Tabelle 9 dargestellt.

| Modell | F1:50 | F1:50-95 | Mean IoU |
|---|---|---|---|
| YOLOv8 | 0,8497 | 0,5098 | 0,7311 |
| YOLOv5 | 0,8343 | 0,4641 | 0,7159 |
| FR-CNN | 0,699 | 0,3437 | 0,4863 |
| RetinaNet | 0,1414 | 0,0684 | 0,4378 |
| SSD | 0,1475 | 0,0572 | - |

Tabelle 9: zusätzliche Metriken Datensatz 1 (eigene Darstellung)

YOLOv8 erreichte mit einem F1:50 von knapp 0,85 und einem F1:50-95 Wert von 0,51 sowie einem Mean IoU von 0,73 die besten Werte der erstellten Modelle. Dies zeigt ein ausgewogenes Verhältnis zwischen Precision und Recall. Ebenso besitzt YOLOv8 die höchste Genauigkeit der Bounding Boxes im Verhältnis zu den Ground Truth Boxes. Auch hier liegt die Leistungsfähigkeit von YOLOv5 nur leicht unter den Werten von YOLOv8. Bei FR-CNN liegen beide F1 Werte im Verhältnis zum mAP höher als bei den YOLO Modellen. Dies ist auf den hohen Recall und der gleichzeitig geringeren Precision zurückzuführen. Bei den YOLO Modellen sind Precision und Recall hingegen ausgegeglichener, wodurch F1 und mAP ähnliche Werte aufweisen. RetinaNet und SSD weisen auch hier schwächere Werte auf, was die geringe Erkennungsleistung bestätigt. Zudem konnte kein Wert für ein Mean IoU für SSD berechnet werden. Die Betrachtung der zusätzlichen Werte bestätigt letztendlich die bisherige Auswertung der Ergebnisse.

[39] Selbe Schwellenwerte wie bei mAP50 und mAP50-95.

## Klassenmetriken

In der Klassenanalyse wird nur auf YOLOv8 eingegangen, da YOLOv5 in der vorangegangenen Auswertung in allen Metriken ähnliche Werte aufwies und der Leistungsunterschied zu den anderen Modellen sehr groß ausfällt. Für die Analyse der Leistungsunterschiede in den Klassen genügt die Betrachtung des besten Modells.

| Nummer | Klasse | mAP50 | mAP50:95 | Precision | Recall |
|---|---|---|---|---|---|
| 0 | Mischwald | 0,856 | 0,398 | 0,887 | 0,750 |
| 1 | Bodenpunkt | 0,969 | 0,398 | 0,706 | 1,00 |
| 2 | Nadelwald | 0,940 | 0,491 | 0,922 | 0,957 |
| 3 | Gebüsch | 0,867 | 0,501 | 0,659 | 0,778 |
| 4 | Schilf | 0,857 | 0,295 | 0,809 | 0,928 |
| 5 | Laubwald | 0,842 | 0,455 | 0,951 | 0,667 |

Tabelle 10: Klassenmetriken Datensatz 1 (eigene Darstellung)

Grundsätzlich weisen alle Klassen bei mAP50 hohe Werte auf und liegen im Bereich 0,84 bis 0,97. Beim mAP50:95 liegen bis auf Schilf alle Klassen bei einem Wert zwischen 0,4 und 0,5. Am besten erkannt wird der Nadelwald mit durchgängig guten Werten in allen Metriken. Precision und Recall variieren zum Teil stark. Die Symbole für Gebüsch, Mischwald und Laubwald besitzen einen geringeren Recall als die anderen Klassen. Dennoch erreichten sowohl der Mischwald als auch der Nadelwald eine hohe Precision. Dies lässt darauf hindeuten, dass diese Klassen zwar zuverlässige Erkennungen tätigen, aber viele Symbole übersehen werden. Das Gebüschsymbol hat eine geringe Precision, erreichte aber dennoch den höchsten mAP50:95 Wert. Es werden also viele Vorhersagen getätigt, die aber einige falsche Objekte erkennen. Werden richtige Objekte allerdings erkannt, werden die Boxen gut platziert, daher der hohe Wert fü mAP50:95. Schilf hingegen besitzt eine geringere mAP50:95, obwohl die anderen Metriken hohe Werte aufweisen. Viele Schilf Symbole werden daher korrekt erkannt, aber die Bounding Boxes passen über die Schwellenwerte verteilt nicht gut auf die Ground Truth Boxes. Insgesamt zeigen die Ergebnisse, dass YOLOv8 alle Klassen gut erkennt. Dennoch bestehen

Unterschiede im Verhältnis von Precision und Recall sowie in der Genauigkeit der vorhergesagten Bounding Boxes.

## Trainingsverlauf

Zur weiteren Analyse wird das Trainingsverhalten für das beste Modell YOLOv8 betrachtet. Dafür sind in der Abbildung 6 Trainings- und Validierungsloss je Epoche dargestellt. Zu sehen sind der Box Loss, Class Loss und Distribution Focal Loss (DFL). Der Box Loss misst die Genauigkeit der vorhergesagten Box zur Ground Truth Box, der Class Loss bewertet die Klassenzugehörigkeit und der DFL Loss dient zur Feinabstimmung der Boxes. Alle Verlustfunktionen zusammen ergeben den Total Loss, welcher zeigt, wie gut das Modell insgesamt arbeitet (vgl. TORRES, 2024). Die Auswertung der unterschiedlichen Verlustfunktionen gibt einen detaillierten Einblick in das Lernverhalten des Modells und veranschaulicht die Stabilität der Konvergenz.

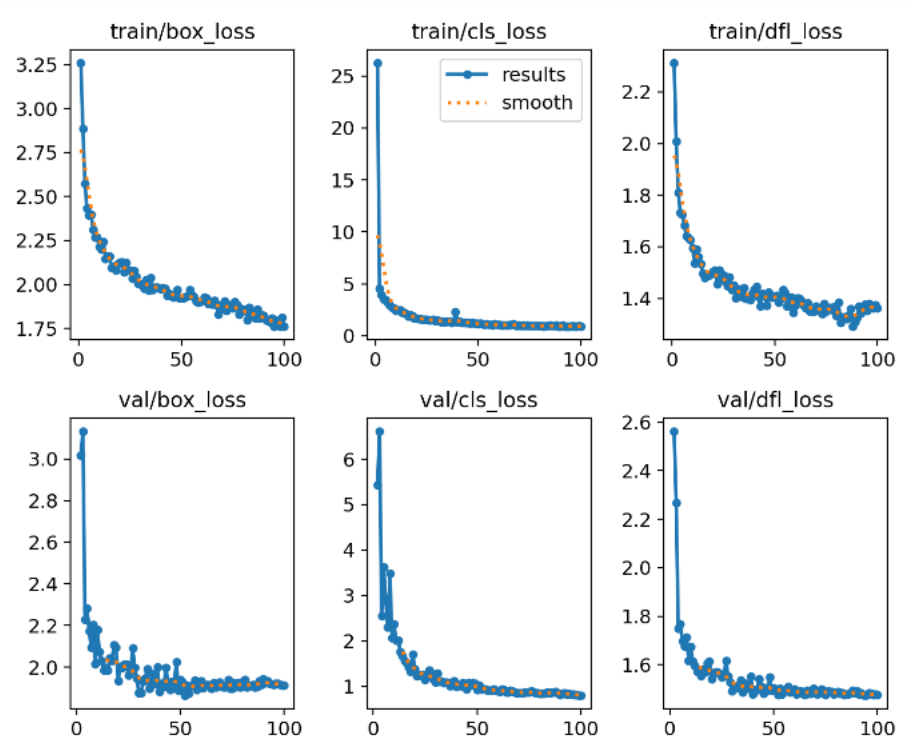



Abbildung 8: Auswertung der Verlustfunktionen YOLOv8 Datensatz 1 (eigene Darstellung)

Der Box Loss fällt kontinuierlich von 3.25 auf 1,75 im Training und zeigt auch in der Validierung ein ähnliches Verhalten mit einigen kleinen Ausreißern innerhalb der ersten 50 Epochen. Das Modell lernt also die Objekte immer genauer zu bestimmen. Der Class

Loss nimmt vor allem am Anfang stark ab. Er sinkt schon in der ersten Epoche von 25 auf 5 und pendelt sich nach 20 Epochen bei ungefähr 1,5 ein. Daher wird die Klassenzuordnung sehr schnell erlernt. Der DFL Loss zeigt ein ähnliches Muster wie der Box Loss und fällt von ungefähr 2,35 auf einen Wert von 1,4. Letztendlich weisen alle Loss-Kurven eine typische L-Form auf, sie beginnen mit hohen Werten und fallen innerhalb der ersten Epochen stark ab, um dann abzuflachen. Diese Werte deuten auf ein stabiles und effektives Training hin. Da der Validierungsloss grundlegend dem Trainingsloss folgt ist auch kein Zeichen auf Overfitting zu erkennen. Der Verlauf der mAP-Werte bestätigt die Loss-Kurven. Sowohl die mAP50 als auch mAP50:95 Werte steigen im Trainingsverlauf weiter an und stabilisieren sich gegen Ende der Kurve. Es zeigt sich, dass das Modell schon frühzeitig nach knapp 50 Epochen viele Symbole korrekt erkennt und danach eher Feinabstimmungen durchführt, was zum Teil auch auf den eingebauten Scheduler zurückzuführen ist.

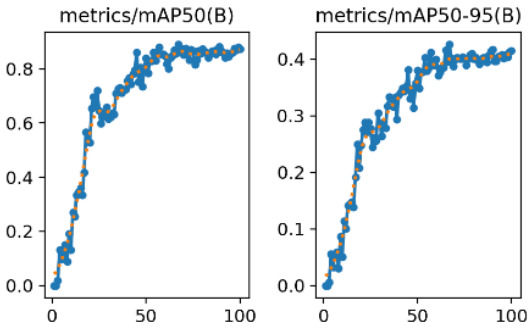



Abbildung 9: mAP50 und mAP50-95 je Epoche im Trainingsverlauf (eigene Darstellung)

## Anwendung

Nachdem die Metriken ausgewertet wurden, wird nun das beste Modell für die Erkennung angewendet. Dafür werden die Erkennungen der Validierungsbilder betrachtet.

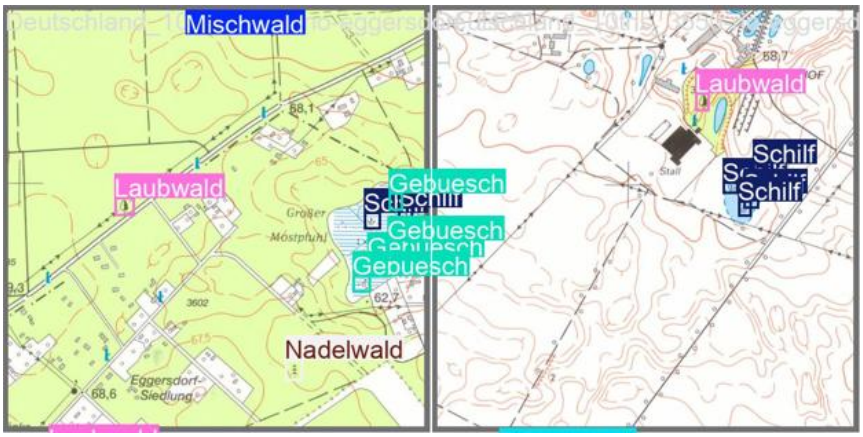



Abbildung 10: Beispieltiles aus den Validerungsdaten mit Annotationen Datensatz 1 (eigene Darstellung, Daten basierend auf GEOMEDIENLABOR)

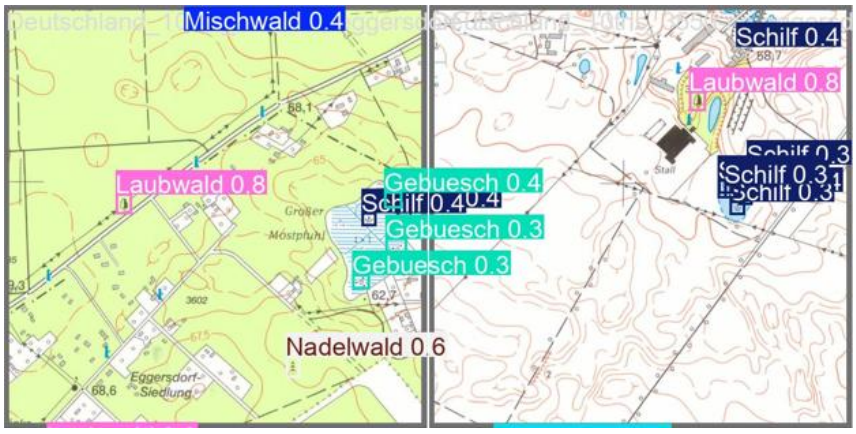



Abbildung 11: Beispieltiles aus den Validerungsdaten mit Vorhersagen Datensatz 1 (eigene Darstellung, Daten basierend auf GEOMEDIENLABOR)

Die Abbildungen zeigen zwei beispielhafte Tiles aus dem Validierungsdatensatz. Die Abbildung 8 zeigt die manuell erstellten Annotationen, die Abbildung 9 die Erkennungen mit dem dahinter stehenden Konfidenzwert. Die Darstellung dient dazu einschätzen zu können, wie gut bestimmte Symbole in der Realität vom Modell erkannt werden. Den höchsten Konfidenzwert erreichte dabei das Symbol für den Laubwald mit einem Wert von 0,8, dahinter der Nadelwald mit 0,6. Die Symbole für Gebüsch und Schilf erreichten hingegen geringere Werte von 0,3 bis 0,4. Dieser Unterschied im Konfidenzwert kann auf die unterschiedlichen Hintergründe zurückgeführt werden. Während die Symbole für Laub- und Nadelwald meistens auf einem einheitlichen grünen oder weißen Hintergrund dargestellt sind, variieren die Hintergründe für Schilf und Gebüsch deutlich mehr. Vor

allem die blau gestreiften Hintergründe, die Moorlandschaften darstellen, können eine zuverlässige Erkennung durch das Modell erschweren.

### 6.1.2 Datensatz 2

**Standardmetriken**

Auch für den zweiten Datensatz werden im folgenden die Standardmetriken evaluiert. Die herangehensweise ist dieselbe wie für den ersten Datensatz.

| Modell | mAP50 | mAP50:95 | Precision | Recall |
|---|---|---|---|---|
| YOLOv8 | 0,0239 | 0,0106 | 0,5139 | 0,0249 |
| YOLOv5 | 0,5317 | 0,2970 | 0,6494 | 0,4904 |
| FR-CNN | 0,0565 | 0,0328 | 0,0677 | 0,0745 |
| RetinaNet | 0,013 | 0,0071 | 0,0100 | 0,0845 |
| SSD | 0,0083 | 0,0037 | 0,0161 | 0,0659 |

Tabelle 11: Standardmetriken Datensatz 2 (eigene Darstellung)

Die Auswertung des zweiten Datensatzes zeigt deutlich geringere Werte für alle Metriken. Bis auf YOLOv5 haben alle Modelle sehr wenig gelernt. Diese Werte sind insgesamt auf die nicht zum Datensatz passenden Modellparameter zurückzuführen. Aufgrund der fehlenden Anpassungen auf die Besonderheiten des komplexeren Datensatzes sind die Modelle kaum in der Lage zu lernen. Lediglich YOLOv5 war robust genug, mit den gewählten Einstellungen zu arbeiten und die Konvergenz zu erreichen. Hier wurde eine moderate Leistung erreicht. Dennoch lernen die Modelle, aber deutlich langsamer als bei dem ersten Datensatz, dies zeigt die Auswertung der Ergebnisse des YOLOv8 Modells. Beispielsweise zeigt die Entwicklung der mAP Werte eine stetige Verbesserung, ist aber weit entfernt von einer Konvergenz. Daher kann davon ausgegangen werden, dass das Modell über mehr Epochen auch noch weiter lernen würde. Eine weitere Möglichkeit wäre es die Lernrate zu erhöhen.

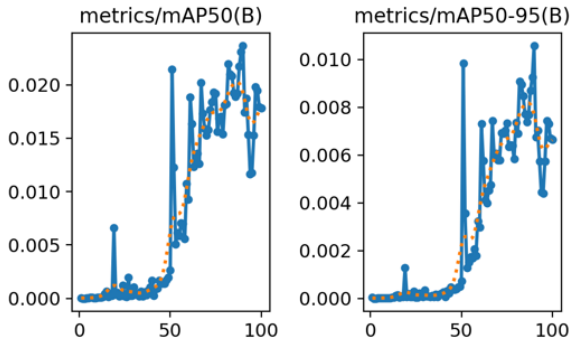



Tabelle 12: mAP50 und mAP50-95 je Epoche für YOLOv8 (eigene Darstellung)

## Zusätzliche Metriken

| Modell | F1:50 | F1:50-95 | Mean IoU |
|---|---|---|---|
| YOLOv8 | 0,0179 | - | - |
| YOLOv5 | 0,5588 | 0,0331 | 0,7403 |
| FR-CNN | 0,0709 | 0,0709 | 0,1392 |
| RetinaNet | 0,0179 | 0,0095 | 0,0954 |
| SSD | 0,0258 | 0,0128 | 0,1086 |

Tabelle 13: zusätzliche Metriken Datensatz 2 (eigene Darstellung)

Auch die Werte der zusätzlichen Metriken sind erwartungsgemäß gering. Auffallend ist der sehr geringe F1:50:95 Wert für YOLOv5 im Vergleich zum F1:50. Dies bedeutet, dass YOLOv5 zwar die Objekte erkennt, die Bounding Boxes jedoch nicht gut ausgerichtet werden. Mögliche Gründe dafür können die unzureichende Anzahl an Trainingsdaten sein, ungenügende Auflösung oder eben die nicht optimal gewählten Modellparamter. Die Ergebnisse untermauern die Annahme, dass die Modelle Probleme haben Merkmale mit den angewendeten Modellparametern zu lernen. lediglich YOLOv5 hat insgesamt eine ausreichende Leistung mit geringer Genauigkeit erzielt.

## Klassenmetriken

In Zuge der Auswertung der Klassenergebnisse wird auch hier das beste Modell gewählt. YOLOv5 hat sich dabei klar als bestes Modell herausgestellt.

| Nummer | Klasse | mAP50 | mAP50:95 | Precision | Recall |
|---|---|---|---|---|---|
| 0 | Human Landscape | 0,392 | 0,208 | 0,512 | 0,391 |
| 1 | Natural Scenery | 0,239 | 0,157 | 0,379 | 0,333 |
| 2 | Human | 0,817 | 0,432 | 0,795 | 0,754 |
| 3 | Animal | 0,664 | 0,381 | 0,723 | 0,540 |
| 4 | Culture | 0,486 | 0,259 | 0,761 | 0,400 |
| 5 | Vehicle | 0,592 | 0,346 | 0,712 | 0,513 |

Tabelle 14: Klassenmetriken Datensatz 2 (eigene Darstellung)

Die Tabelle 14 zeigt eine hohe Varianz in den Leistungsunterschieden zwischen den Klassen. Die mAP50 Werte liegen in einem Bereich von 0,239 und 0,817. Human Landscape erreichte dabei ein moderates Ergebnis über alle Klassen hinweg mit leicht schwächeren Werten als bei den anderen Klassen. Lediglich Natural Scenery weist eine deutlich schwächere Leistung auf als alle anderen Klassen. Das Modell hat also Probleme natürliche Objekte zu identifizieren. Dies ist auf die hohe Varianz in den Piktogrammen zurückzuführen. Es wurden meistens Bäume oder Berge annotiert und eine klare Abgrenzung zur Klasse Human Landscape war nicht immer gegeben. Die besten Ergebnisse hingegen zeigt die Klasse Human. Hier wurden in allen Metriken gute Werte erzielt. Diese hohe Erkennungsleistung kommt durch die gleichen Eigenschaften von Menschen zustande. Sie alle besitzen immer eine ähnliche Struktur, egal in welchem Stil sie erstellt wurden. Ebenso ist es die Klasse mit den meisten Annotationen. Auch die Klasse Animal mit den zweitmeisten Annotationen wird gut erkannt. Nur der Recall ist im Verhältnis zu den anderen Metriken dieser Klasse geringer. Es werden daher einige Tiere vom Modell übersehen, aber wenn sie erkannt werden, sind sie auch oft korrekt. Die Klasse Culture war schon in der Erstellung der Annotationen nicht leicht zuzuordnen. Hauptsächlich besteht sie aus Flaggen oder kulturellen Objekten, wie Musikinstrumenten. Dennoch wurde eine moderate Leistung erreicht mit einer hohen Precision und einem geringen Recall. Die letzte Klasse Vehicle erreichte insgesamt ähnliche Werte, wie die Animal Klasse. Auch sie wurde zum Teil gut erkannt aufgrund der

wenigen Variation von Objekten. Hauptsächlich wurden Schiff annotiert, die oft eine ähnliche Struktur aufweisen. Letztendlich konnten einige Klassen gute Werte erreichen. Der Recall ist dabei meist geringer als die Precision, was auf ein konservatives Verhalten des Modells hinweist. Es werden eher weniger Klassen erkannt als zu viele. Die Varianzen in den Klassen lassen sich überwiegend auf die unterschiedliche Klassenverteilung oder die Komplexität der Klasse zurückzuführen.

## Trainingsverlauf

Im weiteren Verlauf werden die Verlustfunktionen und die Entwicklung des mAP betrachtet. Sie helfen zu verstehen, wie gut das Modell lernen konnte.

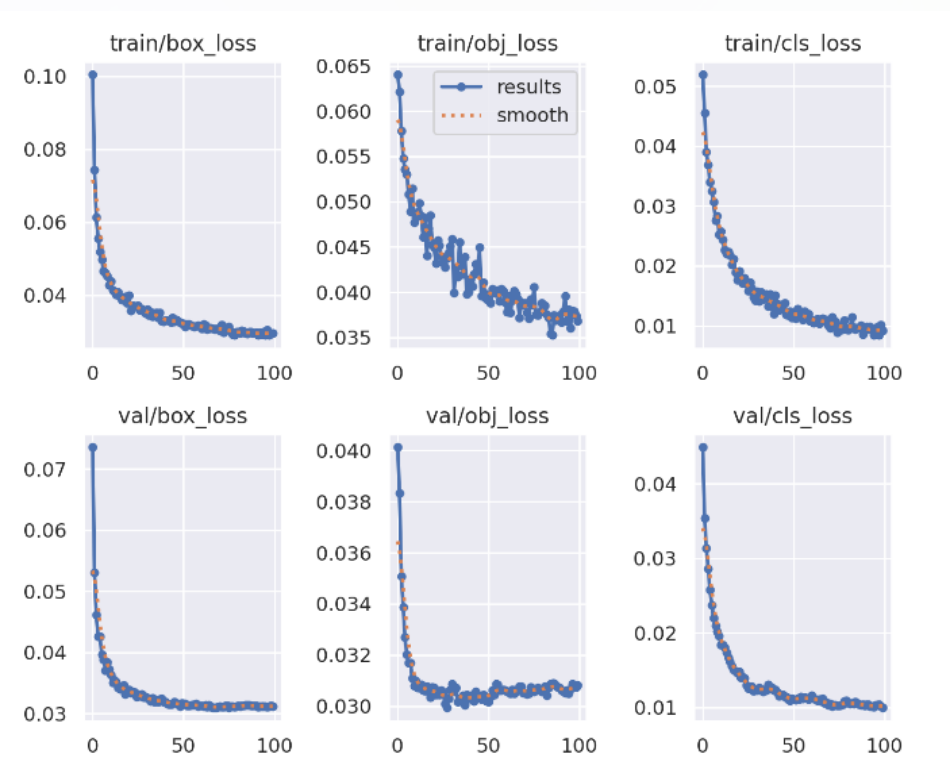



Abbildung 12: Auswertung der Verlustfunktionen YOLOv8 Datensatz 2 (eigene Darstellung)

Die Abbildung 10 zeigt die verschiedenen Verlustfunktionen für das YOLOv5 Modell. Grundsätzlich haben sie alle die klassische abfallende Kurve, wie im ersten Datensatz. Der Loss startet hoch und sinkt immer weiter ab. Dies spricht für ein gut funktionierendes Training ohne starkes Overfitting. Auch die Werte für mAP50 und mAP50:95 in Abbildung 10 zeigen die zu erwartende Kurve. Alle Graphen deuten darauf hin, dass das Modell die

Konvergenz weitestgehend erreicht hat. Eine starke Verbesserung in weiteren Epochen ist daher nicht zu erwarten.

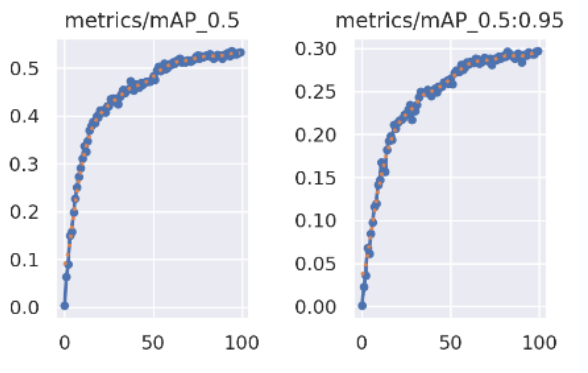



Abbildung 13: mAP50 und mAP50-95 je Epoche für YOLOv5 (eigene Darstellung)

## Anwendung

Im Folgenden wird das YOLOv5 Modell auf die Validierungsbilder angewendet und die Ergebnisse ausgewertet. Ebenso wird eine neue unabhängige Karte ausprobiert, die nicht Teil der Validierungsbilder ist.


Abbildung 14: Beispieltiles aus den Validerungsdaten mit Annotationen Datensatz 2 (eigene Darstellung, Daten basierend auf BIG TEN ACADEMIC ALLIANCE)


Abbildung 15: Beispieltiles aus den Validerungsdaten mit Vorhersagen Datensatz 2 (eigene Darstellung, Daten basierend auf BIG TEN ACADEMIC ALLIANCE)

Die Auswertung für zwei beispielhafte Tiles aus dem Validierungsdatensatz zeigt insgesamt einen hohen Konfidenzwert von 0,7 bis 0,9 für die Human Klasse. Ebenfalls erkannt mit einem Wert von 0,7 wurde der Hund. Obwohl nur ein Objekt als Ground Truth in diesem Ausschnitt annotiert wurde, erkannte das Modell drei weitere Objekte der Human Landscape Klasse mit einem Wert von 0,4 bis 0,6. Da es zu aufwendig gewesen wäre jedes Haus auf dieser Karte zu annotieren, wurden nur beispielhafte Objekte gewählt, dies hat jedoch direkt Einfluss auf die Precision und den Recall der Klasse. Die Objekte wurden zwar korrekt erkannt[40], werden aber als False Positive vom Modell ausgewertet. Da dieses Vorgehen über beide Datensätze einheitlich so angewendet wurde, lässt sich schlussfolgern, dass das Modell bessere Leistung erzielte, als es die Metriken widerspiegeln.

[40] True Positive (TP)

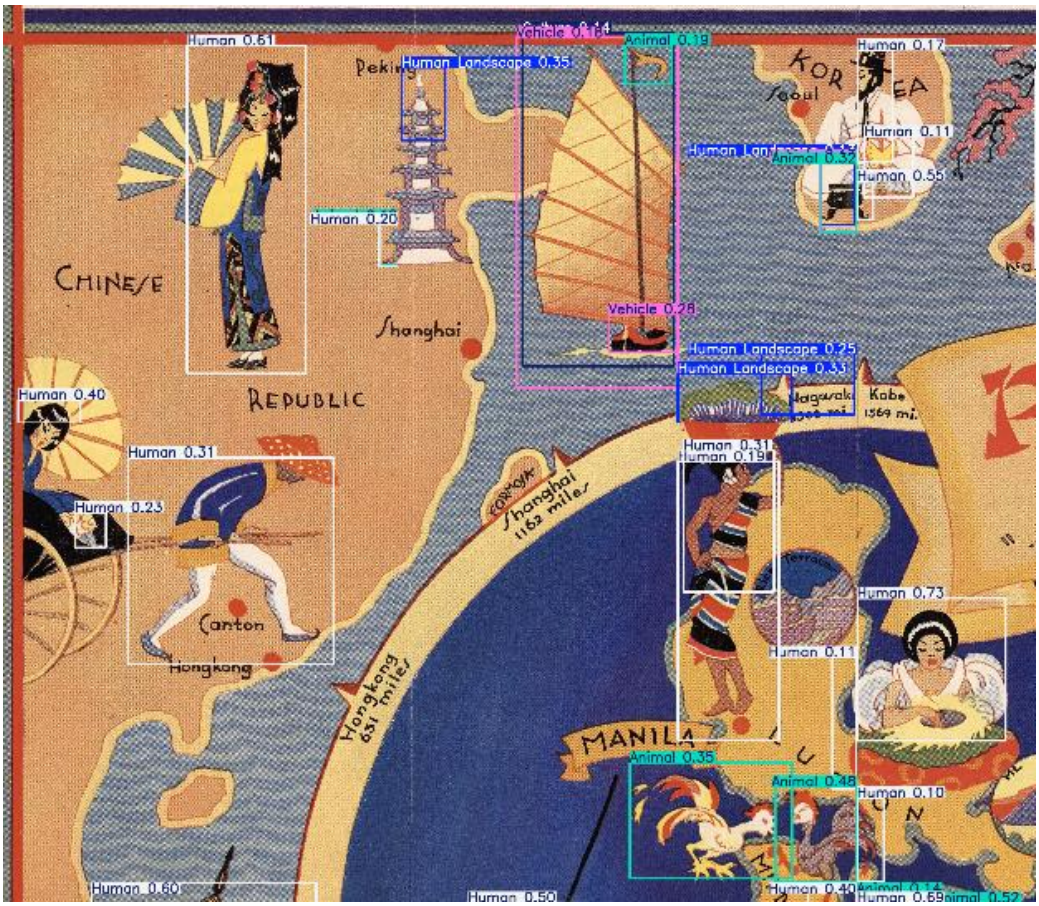



Abbildung 16: Vorhersagen auf einer unabhängigen Karte (eigene Darstellung auf Basis von WHITE, 1930)

Für die Anwendung einer neuen Karte, wurde eine thematische Karte von den Phillipinen gewählt. Sie stammt aus der David Ramsey Map Collection und enthält inhaltlich alle Piktogrammklassen wie die Trainingsdaten. In Abbildung 14 ist ein Ausschnitt aus dem Erkennungsergebnis zu sehen. Es fällt auf, dass die Klassen Human und Animal am besten erkannt werden. Auch das Schiff wurde korrekt als Vehicle und der Turm korrekt als Human Landscape identifiziert jedoch mit geringer Sicherheit. Insgesamt konnten die meisten Objekte gut erkannt und einer Klasse zugeordnet werden. Dennoch gibt es einige falsch erkannte Objekte wie die Flagge vom Schiff, welche als Animal wahrgenommen wurde. Diese besitzen allerdings meist einen geringen Konfidenzwert und können durch eine Anpassung des NMS Wertes ausgeschlossen werden, jedoch mit dem Risiko, dass weniger richtige Objekte erkannt werden. Ebenfalls sind die Bounding Boxes nicht immer gut platziert. Vor allem bei dem Turm wurde nur die Spitze erkannt.

## YOLOv8 Reevaluation

Da die Ergebnisse für YOLOv8 mit den verwendeten Modellparametern keine brauchbaren Ergebnisse geliefert haben, wurde ein weiterer Trainingsversuch gestartet. Dafür wurde YOLOv8 ohne einen Detection Trainer angewendet. Die Werte für den Optimizer wurden nicht verändert, die Lernrate bleibt bei 0,003 und der Weight Decay bei 0,0005. Standardmäßig wird AdamW verwendet, welcher für YOLOv8 optimiert ist. Dies hat den Vorteil, dass ein anderer Scheduler verwendet wird und dieser nicht überschrieben werden muss. Dadurch verändert sich die Trainingszeit deutlich. Mit diesen Einstellungen konnte nun nach 100 Epochen ein mAP50 von 0,550 und ein mAP50:95 von 0,304 erreicht werden. Die Metriken der einzelnen Klassen unterscheiden sich dabei nur geringfügig von den Ergebnissen aus YOLOv5. Ebenso zeigt der Verlauf der mAP Werte in Abb 15 , dass die Leistungsverbesserung immer weiter abflacht. Dies deutet darauf hin, dass weiteres Training keine wesentlichen Verbesserungen bringen würde und das Modell nahe dran ist, die Konvergenz zu erreichen. Da sowohl YOLOv5 als auch YOLOv8 ähnliche Werte erreichten, ist davon auszugehen, dass die vorhandenen Daten keine höheren Werte zulassen. Für bessere Leistungsergebnisse müsste eine Augmentation[41], ein Weightening der unausgeglichenen Klassen oder einfach ein größerer Datensatz verwendet werden.

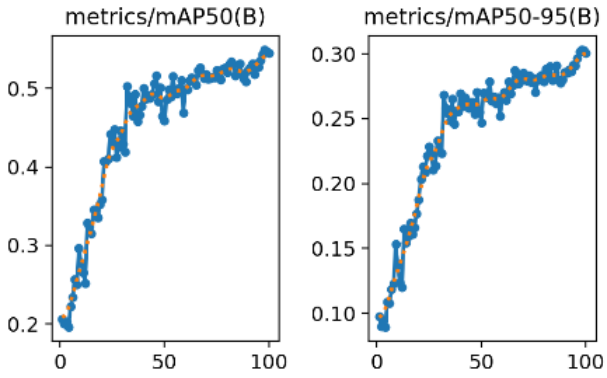



Abbildung 17: mAP50 und mAP50-95 je Epoche für YOLOv8 (eigene Darstellung)

[41]Augmentation bedeutet, eine künstliche Vergrößerung des Datenbestands vorzunehmen.

## 6.2 Segmentierung mit SAM

Im Anschluss an die Objekterkennung wurde SAM auf die erkannten Bounding Boxes angewendet. Für die Auswertung dieser Ergebnisse wurde einmal das beste Modell YOLOv8 für den ersten Datensatz und das YOLOv5 Modell für den zweiten Datensatz auf den Validierungsbildern durchgeführt. Die erkannten Bounding Boxes wurden dann als Bildausschnitt abgespeichert. Auf diesen Ausschnitten wurden dann die relevanten Objekte manuell maskiert. Diese dienen als Ground Truth, um die Metriken für SAM zu bestimmen. Ziel ist es aus den übergebenen Bounding Boxes die Objekte möglichst genau zu segmentieren.

### 6.2.1 Datensatz 1

Aufgrund der Fragmentierung und der geringen Größe der Objekte des ersten Datensatzes war eine manuelle Erstellung von pixelgenauen Ground Truth Annotationen für alle Objekte schwer umsetzbar. Auch für den zweiten Datensatz wäre diese Vorgehensweise sehr aufwendig gewesen, weshalb in der Forschung häufig Teams oder mehrere Probanden eine solche Aufgabe übernehmen. Daher wurde nur ein representatives Beispiel für jede Klasse untersucht. Diese wurden zunächst manuell selektiert und anschließend annotiert.

| Nummer | Klasse | Maske | Segmentierung | Bounding Box |
|---|---|---|---|---|
| 0 | Mischwald |  |  |  |
| 1 | Bodenpunkt |  |  |  |
| 2 | Nadelwald |  |  |  |
| 3 | Gebüsch |  |  |  |
| 4 | Schilf |  |  |  |

| | | | | |
|---|---|---|---|---|
| 5 | Laubwald |  |  |  |

Tabelle 15: visuelle Segmentierungsergebnisse Datensatz 1 (eigene Darstellung, Daten basierend auf GEOMEDIENLABOR)

In der Tabelle 15 sind die Ergebnisse der Segmentierung je Klasse dargestellt. Sie zeigt die von SAM erstellte Maske und anschließende Segmentierung sowie die Bounding Box, welche als Vorlage gilt. Insgesamt fällt auf, dass SAM Probleme hat Symbole zu erkennen, die aus nicht zusammenhängenden Pixeln bestehen. Es fehlen bei den Symbolen für Gebüsch als auch für Schilf die äußeren Teile des Symbols. Sie werden von SAM nicht als Teil wahrgenommen und daher entfernt. Dies liegt an der Struktur von SAM. Es wurde nur ein mittiger Punkt gesetzt, welcher als Grundlage für die SAM Segmentierung dient. An diesem orientiert sich SAM und arbeitet dann mit Farb- und Strukturunterschieden. Es gibt die Möglichkeiten dieses Problem zu umgehen, indem weitere Punkte im Bild gesetzt oder verschiedene Ausgaben miteinander kombiniert werden. Beide Versuche haben jedoch eher zu einer Verschlechterung der Segmentierung geführt. Dies liegt wahrscheinlich an der kleinen Größe der Objekte und der damit verbundenen schwachen Auflösung. Außerdem ist SAM auf natürliche Objekte vortrainiert. Daher können komplexere Strukturen besser erkannt werden. Ein ähnliches Problem liegt bei dem Symbol für den Laubwald vor, hier wird der innere Teil des Symbols als Objekt wahrgenommen. Ebenso beim Mischwald Symbol, die Objekte stehen mit einer Lücke nebeneinander und der Referenzpunkt liegt mittig, wodurch der Hintergrund als Objekt identifiziert wurde. Mit einer weiteren Python Funktion konnte die Maske jedoch invertiert werden. SAM segmentiert grundsätzlich zuverlässig, stößt aber bei den kleinen und fragmentierten Objekten an seine Grenzen. Nachbearbeitungen, wie die Inversion, können bestimmte Fehler ausgleichen, eine fehlerfreie Segmentierung ist jedoch nicht möglich.

| Nummer | Klasse | IoU | HD | DSC |
|---|---|---|---|---|
| 0 | Mischwald | 0,6873 | 3,00 | 0,8146 |
| 1 | Bodenpunkt | 0,7599 | 3,61 | 0,8636 |

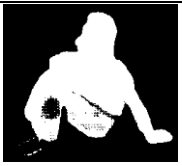
| | | | | |
|---|---|---|---|---|
| 2 | Nadelwald | 0,6312 | 2,24 | 0,7739 |
| 3 | Gebüsch | 0,3575 | 11,00 | 0,5267 |
| 4 | Schilf | 0,2727 | 9,49 | 0,4286 |
| 5 | Laubwald | 0,4089 | 10,30 | 0,5804 |

Tabelle 16: quantitative Segmentierungsergebnisse Datensatz 1 (eigene Darstellung)

Auch die Bewertungsmetriken bestätigen im allgemeinen die Ergebnisse aus der visuellen Auswertung. Mischwald, Bodenpunkt und Nadelwald erreichen in allen Metriken deutlich bessere Werte als die anderen drei Klassen. Sie weisen eine hohe Überschneidung mit dem Ground Truth auf und erreichten IoU Werte zwischen 0,63 und 0,76 sowie einem DSC von 0,77 bis 0,86. Auch die HD ist mit 2,24 bis 3,61 Pixeln niedrig, was für eine geringe Abweichung spricht. Gebüsch, Schilf und Laubwald hingegen erreichten niedrige Werte und eine hohe HD. Dies ist auf unvollständig maskierte oder falsch interpretierte Randbereiche der Symbole zurückzuführen.

### 6.2.2 Datensatz 2

Im folgenden Unterkapitel wird der zweite Datensatz in Hinblick auf die Segmentierungsqualität ausgewertet. Die Vorgehensweise gleicht der des ersten Datensatzes.

| Nummer | Klasse | Maske | Segmentierung | Bounding Box |
|---|---|---|---|---|
| 0 | Human Landscape |  |  |  |
| 1 | Natural Scenery |  |  |  |
| 2 | Human |  |  |  |

| | | | | |
|---|---|---|---|---|
| 3 | Animal |  |  |  |
| 4 | Culture |  |  |  |
| 5 | Vehicle |  |  |  |

Tabelle 17: visuelle Segmentierungsergebnisse Datensatz 2 (eigene Darstellung, Daten basierend auf BIG TEN ACADEMIC ALLIANCE)

Die visuelle Auswertung des zweiten Datensatzes in Tabelle 17 zeigt die Segementierungsqualität von SAM für die Piktogramme. Für die Beispielbilder wurden dabei verschiedene Ergebnisse erzielt. Das Haus für die Klasse Human Landscape wurde grundsätzlich erkannt, jedoch mit Schwächen die Konturen wahrzunehmen. Außerdem wurde der linke Teil des Hauses nicht als zugehörig betrachtet. Die Segmentierung des Baums für Natural Scenery zeigt, dass SAM das Objekt aufgrund der Farbe nur schwer vom Hintergrund trennen kann. Daher bleiben bestimmte Hintergrundpixel bestehen. Auch der Baumstumpf, welcher eine andere Farbe hat und eine Lücke in der Kontur bildet, wurde nur minimal wahrgenommen. Bis auf leichte Konturfehler und die nicht erkannten Füße wurde der Mensch aus der Klasse Human gut erkannt. Auch das Schwein aus der Animal Klasse konnte grundsätzlich gut erkannt werden, jedoch fehlt die komplette Kontur. Für die Culture Klasse wurde ein Wappen gewählt, welches viele Farben und Formen beinhaltet. Hier konnte nur das obere Dreieck als Objekt wahrgenommen werden. Dies liegt vor allem an der Platzierung des Orientierungspunktes. Das beste erkannte Objekt stammt aus der Vehicle Klasse und stellt ein Flugzeug dar. Hier konnte das Objekt fast perfekt vom Hintergrund unterschieden werden. Vor allem aufgrund der Einfarbigkeit des Flugzeugs und des Hintergrunds konnte eine gute Segmentierung erzielt werden.

Zusammenfassend sind die meisten Unterschiede in der Segmentierungsqualität auf die Mehrfarbigkeit des Objekts oder des Hintergrunds zurückzuführen. Auch die Platzierung des Orientierungspunktes spielt eine Rolle. Möglicherweise könnten mehrere gut platzierte Punkte eine Verbesserung erzielen. Diese müssten allerdings auf die Klassen angepasst sein. Des Weiteren kann der Zeichenstil der Piktogramme die Segmentierung beeinflussen. Piktogramme mit Lücken in den Konturen, wie es bei dem Baum der Fall ist, erschweren die Zuordnung für SAM. Ebenso kann die Auflösung entscheidend sein. Dennoch werden die Objekte grundlegend erkannt und zeigen eine höhere Segmentierungsqualität als die fragmentierten Symbole aus dem ersten Datensatz.

| Nummer | Klasse | IoU | HD | DSC |
|---|---|---|---|---|
| 0 | Human Landscape | 0,4196 | 77,23 | 0,5912 |
| 1 | Natural Scenery | 0,7158 | 15,81 | 0,8343 |
| 2 | Human | 0,6936 | 34,71 | 0,8191 |
| 3 | Animal | 0,5870 | 85,00 | 0,7398 |
| 4 | Culture | 0,2475 | 34,37 | 0,3968 |
| 5 | Vehicle | 0,9220 | 5,10 | 0,9594 |

Tabelle 18: quantitative Segmentierungsergebnisse Datensatz 2 (eigene Darstellung)

Die Analyse der quantitativen Metriken zeigt eine deutliche Streuung in den Klassen. Insgesamt liegt die durchschnittliche IoU bei knapp 0,6 und schwankt zwischen 0,25 bis 0,92. Auch der DSC erreichte einen Durchschnitt von 0,72 mit Werten zwischen 0,4 und 0,96. Diese Werte weisen insgesamt auf eine solide Segmentierungsqualität hin. Die hohe HD ist auf die variierende und hohe Auflösung der Eingabekarten zurückzuführen. Daher kann diese Metrik in diesem Datensatz nur bedingt Auskunft über die Segmentierungsqualität geben. Die Klasse Vehicle bestätigt die visuelle Auswertung und erreichte mit Abstand die höchsten Werte. Ebenfalls gute Werte erreichten die Klassen Human und Natural Scenery, die weitestgehend mit dem Ground Truth übereinstimmen. Schwächere Ergebnisse zeigen die Klassen Human Landscape, Animal und Culture.

Aufgrund ihrer Komplexität in Farbe oder Form hat SAM hier Problme eine vollständige Maske zu Erzeugen.

Zusammenfassend konnten zum Teil gute Ergebnisse erzielt werden, jedoch beeinflussen viele Faktoren die Qualität der SAM Segmentierungen. Da nur ausgewählte Bounding Boxes analysiert wurden, kann keine allgemeingültige Aussage für die Segmentierungsqualität des gesamten Datensatzes getroffen werden. Unterschiedliche Klassenobjekte können unter bestimmten Bedingungen andere Ergebnisse erzielen. Die Ergebnisse haben aber auch gezeigt, dass SAM auf kontrastreichen und klaren Symbolen eine gute Segmentierung durchführen kann.

# 7 Diskussion

Die vorliegende Arbeit hatte zum Ziel, einen Ansatz zu entwickeln, der dazu in der Lage ist, bestimmte Objekte in Karten zu erkennen und anschließend zu segmentieren. Dafür orientierte sich die Untersuchung an dem Fachartikel von Cao et al. (2025), um den bestehenden Ansatz aus der Kombination von Objekterkennung und Segmentierung auf eigene Daten zu übertragen und vergleichend zu untersuchen.

Die erste Analyse der einfachen Symbole erreichte unter YOLOv8 einen durchschnittliche mAP50 von 0,85 und eine mAP50-95 von 0,51. Damit erreichte der erste Datensatz insgesamt bessere Werte als der zweite Datensatz mit den Piktogrammen. Dieser konnte unter YOLOv5 eine mAP50 von 0,56 und eine mAP50-95 von 0,30 erzielen. Dies bestätigt, dass visuell einfache und einheitliche Symbole deutlich leichter erkannt werden können als komplexe, variierende und überlappende Piktogramme in historischen Themenkarten.

Dies ist vor allem auf die Kartenart zurückzuführen, während der erste Datensatz aus konsistenten topografischen Karten besteht, enthält der zweite historische Themenkarten, die eine visuelle Aufbereitung von Ereignissen oder Orten bereitstellen. Zudem ist aufgefallen, dass Modellparameter die Leistung stark beeinflussen und je nach Projekt optimiert werden können, um noch mehr Leistung aus den Modellen herauszuholen. Ebenso stellt die geringe Menge an Kartenmaterial eine Limitation dar. Aufgrund der wenigen verfügbaren Karten stand nur bedingt Lernmaterial zur Verfügung, insbesondere im Vergleich zu Studien, die häufig Datensätze mit mehreren tausend Bildern einsetzen. Für kleinere Datensätze können oft Augmentationen oder eine Gewichtung Leistungsverbesserungen erzielen, indem der Datensatz künstlich vergrößert wird. Ein kleineres Tileformat hätte möglicherweise Einfluss auf die Erkennung von kleinen Symbolen und wäre vor allem für SSD vorteilhaft. Zusätzlich müssen Faktoren, wie Kartenstil oder Maßstab berücksichtigt werden, da verschiedene Maßstäbe oder Stile Einfluss auf die Erkennungsqualität haben können.

Die nachfolgende Segmentierung ergab, dass der zweite Datensatz aufgrund der komplexeren Formen besser segmentiert werden konnte. Dies hängt mit den vortrainierten natürlichen Objekten von SAM zusammen. Die Qualität ist jedoch abhängig

von den Referenzpunkten und der Menge an verfügbaren Ground Truth Masken, da SAM auf bestimmte Objektformen trainiert werden kann. Ein größerer Datenbestand an Annotationen hätte daher sowohl das Training verbessert als auch die Evaluation. Eine weitere Möglichkeit für weiterführende Untersuchungen wäre es alternative Segmentierungsverfahren, wie Mask R-CNN zu vergleichen.

Ähnlich wie bei der Studie von Cao et al. (2025) zeigen die Ergebnisse, dass die Leistungsfähigkeit der Objekterkennung und Objektsegmentierung stark von der Kartenart, der Komplexität der Darstellung und den verfügbaren Daten abhängt, was für zukünftige Weiterentwicklungen berücksichtigt werden sollte. Kleine Datensätze führen zu einer Variation in den Ergebnissen der Klassen und ein komplexer Hintergrund oder Überlappungen erschweren die Segmentierung, vor allem bei einer geringen Auflösung. Dennoch konnte insgesamt ein funktionsfähiges System entwickelt werden, dass mehrere moderne Objekterkennungsverfahren mit der Segmentierung kombiniert und auf einer großen Eingabekarte angewendet werden kann. Dieses System funktioniert unabhängig vom Datensatz und kann individuell angepasst werden. Die Anwendung von Segmentierung in Verbindung mit der Objekterkennung ermöglicht eine neue Perspektive in der Kartografie. Viele Symbole oder detaillierte Piktogramme können auf diese Weise leicht aufbereitet und wiederverwendet werden, auch wenn eine Nachbearbeitung nötig sein kann. Ein solches System kann beispielsweise dazu genutzt werden, handgezeichnete Piktogramme aus alten thematischen Karten zu segmentieren und anschließend für eine Neuaufbereitung derselben Karte oder für andere moderne Arbeiten wiederverwendet zu werden. Dadurch kann die Effizienz der Kartenerstellung und Analyse merklich verbessert werden. Dabei muss jedoch beachtet werden, dass Nutzungsrechte für die Verarbeitung bestehen. Bei sehr alten historischen Karten sollte dies kein Problem sein, da das Schutzrecht abgelaufen ist.

Die Entwicklung automatisierter Erkennungs- und Segmentierungsverfahren in der Kartografie steht noch am Anfang. Die Modelle haben ihre Grenzen, doch sie werden kontinuierlich verbessert. Beispielsweise kommen jedes Jahr neue YOLO Versionen raus, welche die vorherigen Modelle übertreffen. Für die Kartografie wäre es in der Zukunft denkbar einen großen Datensatz wie COCO oder PASCAL zu entwickeln, der speziell für

kartografische Objekte konzipiert ist. So könnten die Modelle besser vortrainiert werden und noch bessere Ergebnisse auf Karten erzielen.

# 8 Fazit

Die Masterarbeit verfolgte das Ziel, ein System zu entwickeln, welches die automatisierte Objekterkennung mit der Objektsegmentierung vereint und auf Karten angewendet werden kann. Für die Einschätzung der Leistungsfähigkeit wurden zwei unterschiedlich Datensätze evaluiert. Es konnte ein vollständiger Arbeitsablauf realisiert werden, der einfache Symbole sowie komplexe Piktogramme verarbeiten kann. Insgesamt zeigen die Ergebnisse, dass einheitliche und klare Strukturen besser erkannt werden konnten als komplexe und variierende Piktogramme. Die Segmentierung hingegen konnte komplexe Symbole gut erkennen. Die Qualität wird jedoch von der Auflösung, dem Kartenstil und der Komplexität des Hintergrunds stark beeinflusst. Das System konnte funktionsfähig umgesetzt werden, besitzt aber einige Einschränkungen. Die begrenzte Datenmenge der Trainingsdaten, der Detaillierungsgrad der Darstellungen und die Kartenart beeinflussen die Effektivität des Systems. Für zukünftige Betrachtungen ist es vielversprechend einen großen kartografiebezogenen Trainingsdatensatz zu entwickeln. Ebenso können Modellparameter zur Anpassung und Feinabstimmung der Modelle untersucht werden. Auch Vergleiche mit anderen oder zukünftigen Modellen können die Qualität verbessern. Eine weitere Möglichkeit wäre die Entwicklung eines Frontends, um die Erkennungen und Segmentierungen direkt über eine benutzerfreundliche grafische Oberfläche zu ermöglichen. Letztendlich leistet die Arbeit einen Beitrag zur Weiterentwicklung automatisierter Prozesse innerhalb der Kartografie und eröffnet Möglichkeiten diesen Ansatz weiter auszubauen.

# 9 Literaturverzeichnis


AIML.COM (2025): Cross-Attention vs Self-Attention. – Online in Internet: https://aiml.com/explain-cross-attention-and-how-is-it-different-from-self-attention/ [Stand: 2025-11-16].

BIG TEN ACADEMIC ALLIANCE (Hrsg.): BTAA Geoportal – Online in Internet: https://geo.btaa.org/ [Stand: 2025-11-16]

BOCHKOVSKIY, ALEXEY; WANG, CHIEN-YAO & HONG-YUAN MARK LIAO (2020): YOLOv4: Optimal Speed and Accuracy of Object Detection. – arXiv Preprint: arXiv:2004.10934 [Stand: 2025-11-16].

CAO, DI; YAN, XINRAN; LI, JINGJING; LI, JIAYAO & LILI WU (2025): Automated Icon Extraction from Tourism Maps: A Synergistic Approach Integrating YOLOv8x and SAM. – In: ISPRS International Journal of Geo-Information, vol. 24 (2), 55.

DU, KEIXUAN; REN, FU; WANG, YONG; CHE, XIANGHONG; LIU, JIPING; HOU, JIAXIN & ZEWEI YOU (2024): Integration of Spatial and Co-Existence Relationships to Improve Administrative Region Target Detection in Map Images. – In: ISPRS International Journal of Geo-Information, vol. 13 (6), 216.

EITCA ACADEMY (2023): Wie werden Faltungen und Pooling in CNNs kombiniert, um komplexe Muster in Bildern zu lernen und zu erkennen? – Online in Internet: https://de.eitca.org/k%C3%BCnstliche-Intelligenz/eitc-ai-dltf-tiefes-Lernen-mit-Tensorflow/Faltungs-Neuronale-Netze-im-Tensorfluss/Grundlagen-der-Faltung-neuronaler-Netze/Pr%C3%BCfungs%C3%BCberpr%C3%BCfung-Grundlagen-der-Faltungs-Neuronalen-Netze/Wie-werden-Faltungen-und-Pooling-in-CNNs-kombiniert%2C-um-komplexe-Muster-in-Bildern-zu-lernen-und-zu-erkennen%3F/ [Stand: 2025-11-16].

EVERINGHAM, MARK; VAN GOOL, LUC; WILLIAMS, CHRISTOPHER K. I.; WINN, JOHN & ANDREW ZISSERMAN (2010): The PASCAL Visual Objext Classes (VOC) Challenge. – In: International Journal of Computer Vision, vol. 88 (2), 303–338.

FAUSETT, LAURENE (1994): Fundamentals of Neural Networks. – Prentic Hall International, Inc. (Hrsg.) – Upper Saddle River, New Jersy, USA.

FU, CHENG-YANG; LIU, WEI; RANGA, ANATH; TYAGI, AMBRISH & ALEXANDER C. BERG (2017): DSSD: Deconvolutional Single Shot Detector. – arXiv Preprint: arXiv:1701.06659 [Stand: 2025-11-16].

GALLAGHER, JAMES (2023): What is an Image Embedding? – Online in Internet: https://blog.roboflow.com/what-is-an-image-embedding/ [Stand: 2025-11-16].

GEOMEDIEN LABOR BHT (Hrsg.): Topografische Karte 1:10000. – Berliner Hochschule für Technik.

GIRSHICK, ROSS; DONAHUE, JEFF; DARRELL, TREVOR & JITENDRA MALIK (2014): Rich feature hierarchies for accurate object detection and semantic segmentation. – In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, Ohio, USA, 580–587.

GIRSHICK, ROSS (2015): Fast R-CNN. – arXiv Preprint: arXiv:1504.08083 [Stand: 2025-11-16].

GOODFELLOW, IAN; BENGIO, YOSHUA & AARON COURVILLE (2016): Deep Learning. – The MIT Press (Hrsg.) – Cambridge, Massachusetts, USA.

GROSSBERG, STEPHEN (2013): Recurrent neural Networks. – In: Scholarpedia, vol. 8 (2), 1888.

GU, CHUNHUI; LIM, JOSEPH J.; ARBELAEZ, PABLO & JITENDRA MALIK (2009): Recognition using regions. – IEEE (Hrsg.): 2009 IEEE Conference on Computer Vision an Pattern Recognition (CVPR), Miami, Florida, USA, 1030–1037.

HE, KAIMING; ZHANG, XIANGYU; REN, SHAOQING & JIAN SUN (2016): Deep Residual Learning for Image Recognition. – In: Proceeding of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Nevada, USA, 770–778.

HE, KAIMING; GKIOXARI, GEORGIA; DOLLAR, PIOTR & ROSS GIRSHICK (2017): Mask R-CNN. – arXiv Preprint: arXiv:1703.06870 [Stand: 2025-11-16].

HUO, YINGZI; JIN, KAI; CAI, JIAHONG; XIONG, HUIXUAN & JIACHENG PANG (2023): Vision transformer (ViT)-based Applications in Image Classification. – IEEE (Hrsg.): 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), New York, New York, USA, 135–140.

ISMAIL, WALAA N.; ALSALAMAH, HESSAH A.; HASSAN, MOHAMMED MEHEDI & EBTESAM MOHAMED (2023): AUTO-HAR: An adaptive human activity recognition framework using an automated CNN architecture design. – In: Heliyon, vol. 9 (2), e13636.

JIA, YANGQING; SHELHAMER, EVAN; DONAHUE, JEFF; KARAYEV, SERGEY; LONG, JONATHAN; GIRSHICK, ROSS; GUADARRAMA, SERGIO & TREVOR DARRELL (2014): Caffe: Convolutional Architecture for Fast Feature Embedding. – In: Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, Florida, USA, 675–678.

JOCHER, GLENN (2020): YOLOv5 by Ultralytics (Version 7.0). – Online in Internet: https://github.com/ultralytics/yolov5 [Stand: 2025-11-16].

JOCHER, GLENN; MUNAWAR, MUHAMMAD RIZWAN; YASIN, MOHAMMED; QIU, JING; ALEX; DERRENGER, PAULA; NOYCE, MATTHEW; ULTRALYTICS ASSISTANT; BURHAN; CHAURASIA, AYUSH & FATIH AKYON (2023a): Entdecken Sie Ultralytics

YOLOv8. – Online in Internet: https://docs.ultralytics.com/de/models/yolov8/ [Stand: 2025-11-16].

JOCHER, GLENN; JAN & BURHAN (2023b): Reference for ultralytics/engine/trainer.py. – Online in Internet: https://docs.ultralytics.com/reference/engine/trainer/ [Stand: 2025-11-16].

JOCHER, GLENN; JAN; QIU, JING & BURHAN (2023c): Reference for ultralytics/models/sam/build.py. – Online in Internet: https://docs.ultralytics.com/reference/models/sam/build/ [Stand: 2025-11-16].

JOCHER, GLENN; MUNAWAR, MUHAMAD RIZWAN; DERRENGER, PAULA; YASIN, MOHAMMED; QIU, JING & FRANCESCO MATTIOLI (2024a): Ultralytics YOLO11. – Online in Internet: https://docs.ultralytics.com/de/models/yolo11/ [Stand: 2025-11-16].

JOCHER, GLENN (2024b): YOLO11 vs YOLOv8: Detailed Comparison. – Online in Internet: https://docs.ultralytics.com/compare/yolo11-vs-yolov8/ [Stand: 2025-11-16].

KATTE, TRUPTI (2018): Recurrent Neural Network and its Various Architecture Types. – In: International Journal of Research and Scientific Innovation, vol. 5 (3), 124 – 129.

KIRILLOV, ALEXANDER; MINTUN, ERIC; RAVI, NIKHILA; MAO, HANZI; ROLLAND, CHLOE; GUSTAFSON, LAURA; XIAO, TETE; WHITEHEAD; BERG, ALEXANDER C.; LO, WAN-YEN; DOLLÁR, PIOTR & ROSS GIRSHICK (2023): Segment Anything – In: Proceedings of the IEEE/CVF Internationl Conference on Computer Vision (ICCV), Paris, Frankreich, 4015–4026.

KOTTHAPALLI, MANIKANTA; RAVIPATI, DEEPIKA & RESHMA BHATIA (2025): YOLOv1 to YOLOv11: A Comprehensive Survey of Real-Time Object Detection Innovations and Challanges. – arXiv Preprint: arXiv:2508.02067 [Stand: 2025-11-16].

KRIZHEVSKY, ALEX; SUTSKEVER, ILYA & GEOFFREY E. HINTON (2012): ImageNet Classification with Deep Convolutional Neural Networks. – F. PEREIRA; C. J. BURGES; L. BOTTOU & K. Q. WEINBERGER (Hrsg.): Advances in Neural Information Processing Systems (NIPS 2012), 25, lake Tahoe, Nevada, USA, 1 (4).

LEE, FANGFANG (2023): What is Positional Encoding? – Online in Internet: https://www.ibm.com/think/topics/positional-encoding [Stand: 2025-11-15].

LI, CHUYI; LI, LULU; JIANG, HONGLIANG; WENG, KAIHENG; GENG, YIFEI; LI, LIANG; KE, ZAIDAN; LI, QINGYUAN; CHENG, MENG; NIE, WEIQIANG; LI, YIDUO; ZHANG, BO; LIANG, YUFEI; ZHOU, LINYUAN; XU, XIAOMING; CHU, XIANGXIANG; WEI, XIAOMING & XIAOLIN WEI (2022): YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. – arXiv Preprint: arXiv:2209.02976 [Stand: 2025-11-16].

LI, XIANG; WANG, WENHAI; WU, LIJUN; CHEN, SHUO; HU, XIAOLIN; LI, JUN; TANG, JINHUI & JIAN YANG (2020): Generalized Focal Loss: Learning Qualified and

Distributed Bounding Boxes for Dense Object Detection. – arXiv Preprint: arXiv:2006.04388 [Stand: 2025-11-16].

LI, ZUOXIN; YANG, LU & FUQIANG ZHOU (2017): FSSD: Feature Fusion Single Shot Multibox Detector. – arXiv Preprint: arXiv:1712.00960 [Stand: 2025-11-16].

LIBRARY OF CONGRESS (Hrsg.): Search Maps. – Online in Internet: https://www.loc.gov/maps/ [Stand: 2025-11-16].

LIN, TSUNG-YI; MAIRE, MICHAEL; BELONGIE, SERGE; HAYS, JAMES; PERONA, PIETRO; RAMANAN, DEVA; DOLLÁR, PIOTR & C. LAWRENCE ZITNICK (2014): Microsoft COCO: Common Objects in Context. – Springer International Publishing – European Conference on Computer Vision, Cham, Schweiz, 740–755.

LIN, TSUNG-YI; GOYAL, PRIYA; GIRSHICK, ROSS; HE, KAIMING & PIOTR DOLLÁR (2017a): Focal Loss for Dense Object Detection. – In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venedig, Italien, 2980–2988.

LIN, TSUNG-YI; DOLLÁR, PIOTR; GIRSHICK, ROSS; HE, KAIMING; HARIHARAN, BHARATH & SERGE BELONGIE (2017b): Feature Pyramid Networks for Object Detection. – In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii, USA, 2117–2125.

LIU, WEI; ANGUELOV, DRAGOMIR; ERHAN, DUMITRU; SZEGEDY, CHRISTIAN; REED, SCOTT; FU CHENG-YANG & ALEXANDER C. BERG (2016): SSD: Single Shot MultiBox Detector. – BASTIAN LEIBE; JIRI MATAS; NICU SEBE & MAX WELLING (Hrsg.): Computer – Vision ECCV 2016, vol. 14, Amsterdam, Niederlande, 21-37.

LIU, LI; QUYANG, WANLI; WANG, XIAOGANG; FIEGUTH, PAUL; CHEN, JIE; LIU XINWANG & MATTI PIETIKÄINEN (2020): Deep Learning for Generic Object Detection: A Survey. – In: International Journal of Computer Vision, vol. 128, 261–318.

MAMMONE, ALESSIA; TURCHI, MARCO & NELLO CHRISTIANINI (2009): Support vector machines. – In: Wiley Interdisciplinary Reviews: Computational Statistics, vol. 1 (3), 283–289.

MEDSKER, LARRY & LAKHAMI C. JAIN (1999): Recurrent Neural Networks: Design and Applications. – Boca Raton, Florida, USA: CRC Press.

MINAEE, SHERVIN; BOYKOV, YURI; PORIKILI, FATIH; PLAZA, ANTONIO; KEHTARNAVAZ, NASSER & DEMETRI TERZOPOULOS (2020): Image Segmentation Using Deep Learning: A Survey. – arXiv Preprint: arXiv:2001.05566 [Stand: 2025-11-16].

MÜLLER, DOMINIK; SOTO-REY, INAKI & FRANK KRAMER (2022): Towards a guideline for evaluation metrics in medical image segmentation. – In: BMC Research Notes, vol. 15 (1), 210.

O'SHEA, KEIRON & RYAN NASH (2025): An Introduction to Convolutional Neural Networks. – arXiv Preprint: arXiv:1511.08458 [Stand: 2025-11-16].

PADILLA, RAFAEL; NETTO, SERGIO L. & EDUARD A. B. DA SILVA (2020): A Survey on Performance Metrics for Object-Detection Algorithms. – IEEE (Hrsg.): 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), Niterói, Brasilien, 237–242.

RAIAAN, MOHAIMENUL AZAM KHAN; SAKIB, SADMAN; FAHAD, NUR MOHAMMED; MAMUN, ABDULLAH AL; RAHMAN, MD. ANISUR; SHATABDA & MD. SADDAM HOSSAIN MUKTA (2024): A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks. – In: Decision Analytics Journal, 11, 100470.

RAVI, NIKHILA; GABEUR, VALENTIN; HU, YUANG-TING, HU, RONGHANG; RYALI, CHAITANA; MA, TENGYU; KHEDR, HARITHAM; RÄDLE, ROMAN; ROLLAND, CHLOE; GUSTAFSON, LAURA; MINTUN, ERIC; PAN, JUNTING; ALWALA, KALYAN VASUDEV; CARION, NICOLAS; WU, CHAO-YUAN; GIRSHICK, ROSS; DOLLÁR, PIOTR & CHRISTOPH FEICHTENHOFER (2024): SAM 2: Segment Anything in Images and Videos. – arXiv Preprint: arXiv:2408.00714 [Stand: 2025-11-16].

REDMON, JOSEPH; DIVVALA, SANTOSHI; GIRSHICK, ROSS & ALI FARHADI (2015): You Only Look Once: Unified, Real-Time Object Detection. – In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), las Vegas, Nevada, USA, 779–788.

REDMON, JOSEPH & ALI FARHADI (2016): YOLO9000: Better, Faster Stronger. – arXiv Preprint: arXiv:1612.08242 [Stand: 2025-11-16].

REDMON, JOSEPH & ALI FARHADI (2018): YOLOv3: An Incremental improvement. – arXiv Preprint: arXiv:1804.02767 [Stand: 2025-11-16].

REN, SHAOQING; HE, KAIMING; GIRSHICK, ROSS & JIAN SUN (2015): Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. – CORINNA CORTES; DANIEL D. LEE; ROMAN GARNETT; NEIL D. LAWRENCE & MASASHI SUGIYAMA (Hrsg.): Advances in Neural Information Processing Systems 28 (NIPS) (29th Annual Conference on Neural Information Processing Systems 2015), Montreal, Canada, 91–99.

PYKES, KURTIS (2024): Cross-Entropy Loss Function in Machine Learning: Verbesserung der Modellgenauigkeit. – Online in Internet: https://www.datacamp.com/de/tutorial/the-cross-entropy-loss-function-in-machine-learning [Stand: 2025-11-16].

SCHNÜRER, RAIMUND; SIEBER, RENE; SCHMID-LANTER, JOST; ÖZTIRELI, A. CENGIZ & LORENZ HURNI (2020): Detection of Pictorial Map Objects with Convolutional Neural Networks. – In: The World of Mapping, vol. 58 (1), 50–68.

SIMONELLI, JULIUS (2020): What Do These Different AP Values Mean. – Online in Internet: https://jss367.github.io/what-do-these-different-ap-values-mean.html [Stand: 2025-11-16].

TAM, ADRIAN (2023): Using Learning Rate Scheduler in PyTorch Training. – Online in Internet: https://machinelearningmastery.com/using-learning-rate-schedule-in-pytorch-training/ [Stand: 2025-11-16].

TANVEER, M.; GANAIE, M. A.; BEHESHTI, IMAN; GOEL, TRIPTI; AHMAD, NEHAL; LAI, KUAN-TING; HUANG, KAIZHU; ZHANG, YU-DONG; SER, JAVIER DEL & CHIN-TENG LIN (2023): Deep learning for brain age estimation: A systematic review. – In: Information Fusion, vol. 96, 130–143.

TERVEN, JUAN; CÓRDOVA-ESPARZA, DIANA-MARGARITA & JULIO-ALEJANDRO ROMERO-GONZÁLEZ (2023): A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. – In: Machine Learning and Knowledge Extraction, vol. 5 (4), 1680–1716.

TIAN, YUNJIE, YE, QIXIANG & DAVID DOERMANN (2025): YOLOv12:Attention-Centric Real-Time Object Detectors. – arXiv Preprint: arXiv:2502.12524 [Stand: 2025-11-16].

TORRES, JANE (2024): What is DFL loss in yolov8? – Online in Internet: https://yolov8.org/what-is-dfl-loss-in-yolov8/ [Stand: 2025-11-16].

UIJLINGS, J. R. R.; VAN DE SANDE, K. E. A.; GEVERS, T. & A. W. M. SMEULDERS (2013): Selective Search for Object Recognition. – In: International Journal of Computer Vision, 104 (2), 154–171.

UPRETI, ANJEEL (2022): Convolutional Neural Network (CNN). A Comprehensive Overview. – Online in Internet: 10.20944/preprints202208.0313.v3 [Stand: 2025-11-16].

VERSCHAE, RODRIGO & JAVIER RUIZ DEL SOLAR (2015): Object Detection and Future Directions. – In: Frontiers in Robotics and AI, vol. 2, 29.

WANG, AO; CHEN, HUI; LIU, LIHAO; CHEN, KAI; LIN, ZIJIA; HAN, JUNGONG & GUIGUANG DING (2024): YOLOv10: Real-Time End-to-End Object Detection. – arXiv Preprint: arXiv:2405.14458 [Stand: 2025-11-16].

WANG, CHIEN-YAO; BOCHKOVSKIY, ALEXEY & HONG-YUAN LIAO (2022): YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. – arXiv Preprint: arXiv:2207.02696 [Stand: 2025-11-16].

WANG, CHIEN-YAO; YEH I-HAU & HONG-YUAN Mark LIAO (2024): YOLOv9: Learning What You Want to Learn Using Programmable Gradient information. – arXiv Preprint: arXiv:2402.13616 [Stand: 2025-11-16].

WANG, YONG; DU, KEIXUAN; CHE, XIANGHONG; MA RUIYUAN & FU REN (2023): Recognition and Semantic Information Extraction for Map Based on Deep Learning. – In: Proceedings of the ICA, vol. 5, 25.

WALTERS, AUSTIN G. (2019): Classify Sentences via a Recurrent Neural Network (LSTM). – Online in Internet: https://austingwalters.com/classify-sentences-via-a-recurrent-neural-network-lstm/ [Stand: 2025-11-16].

WHITE, RUTH TAYLOR (1930): A cartograph of the major Philippine Islands. – DAVID REMSEY MAP COLLECTION (Hrsg.). – Online in Internet: https://www.davidrumsey.com/luna/servlet/detail/ RUMSEY~8~1~272056~90045755:A-cartograph-of-the-major-Philippin?mi=1&trs=2&qvq=q:A%20cartograph%20of%20the%20major%20Philippine%20Islands.;lc:RUMSEY~8~1# [Stand: 2025-11-16]

WUTTKE, LAURENZ (2024): Künstliche Neuronale Netzwerke: Defintion, Einführung, Arten und Funktion. – Online in Internet: https://datasolut.com/neuronale-netzwerke-einfuehrung/ [Stand: 2025-11-16].

XIONG, YUNYANG; ZHOU, CHONG; XIANG, XIAOYU; WU, LEMENG; ZHU, CHENCHEN; LIU, ZECHUN; SURI, SAKSHAM; VARADARAJAN, BALAKRISHNAN; AKULA, RAMYA; LANDOLA, FORREST; KRISHNAMOORTHI, RAGHURAMAN; SORAN, BILGE & VIKAS CHANDRA (2024): Efficient Track Anything. – arXiv Preprint: arXiv:2411.18933 [Stand: 2025-11-16].

YUKATA, SASAKI (2007): The truth of the F-measure – Teach tutor Mater. – Online in Internet: https://www.researchgate.net/publication/268185911_The_truth_of_the_F-measure [Stand: 2025-11-16].

ZHANG, HAOYANG; WANG, YING; DAYOUB, FERAS & NIKO SÜNDERHAUF (2020): VarifocalNet: An IoU-aware Dense Object Detector. – arXiv Preprint: arXiv:2008.13367 [Stand: 2025-11-16].

ZHANG, HUILI; ZHOU, XIAOWEN; LI, HUAN; ZHU, GE & HONGWEI LI (2022): Machine Recognition of Map Point Symbols Based on YOLOv3 and Automatic Configuration Associated with POI. – In: ISPRS International journal of Geo-Information, vol. 11 (11), 540.

ZHANG, SHIFENG; CHI, CHENG; YAO, YONGQIANG; LEI, ZHEN & STAN Z. LI (2020): Bridging the Gap Between Anchor-based and Anchor-free Detection via Adaptive Training Sample Selection. – arXiv Preprint: arXiv:1912.02424 [Stand: 2025-11-16].

ZHAO, XU; DING, WENCHAO; AN, YONGQI; DU, YINGLONG; YU, TAO; LI, MIN; TANG, MING & JINQIAO WANG (2023): Fast Segment Anything. – arXiv Preprint: arXiv:2306.12156 [Stand: 2025-11-16].

ZHOU, PAN; FENG, JIASHI; MA, CHAO; XIONG, CAIMING; CHU HONG HOI, STEVEN & E WEINAN (2020): Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning. – H. LAROCHELLE; M. RANZATOR; R. HADSELL; M. F. BALCAN & H. LIN (Hrsg.): Advances in Neural Information Processing Systems (NeurIPS 2020), Bd. 33, 21285–21296.

ZHU, HAIDI; WEI, HAORAN; LI, BAOQING; YUAN, XIAOBING & NASSER KEHTARNAVAZ (2020): A Review of Video Object Detection: Datasets, Metrics and Methods. – In: Applied Sciences, vol. 10 (21), 7834.

ZOU, ZHENGXIA; CHEN, KEYAN; SHI, ZHENWEI; GUO, YUHONG & JIEPING YE (2023): Object Detection in 20 Years: A Survey. – IEEE (Hrsg.): Proceedings of the IEEE, 111 (3), 257–276.

**Zusätzliche Materialien:** Der Code zum System und der dazugehörigen Auswertung ist auf einem beigelegten USB-Stick enthalten. Dort befinden sich alle Dateien, welche für die Anwendung benötigt werden.

**Hinweis zum Einsatz von KI:** Im Verlauf der Arbeit wurde gelegentlich das KI-Werkzeug ChatGPT angewendet. Es diente zur Unterstützung bei der Code-Erstellung oder zur Übersetzung und sprachlichen Aufbereitung von Fachliteratur. Die inhaltliche Verantwortung liegt vollumfänglich beim Autor.