INSTITUT FÜR INFORMATIK
Fachbereich Informatik und Mathematik

Masterthesis

Klassifizierung von Motiven auf antiken Münzen mit Mask R-CNN

Huy Luong

Frankfurt am Main
21.12.2022

eingereicht bei: Dr. Karsten Tolle

Professur für Texttechnologie
Goethe Universität Frankfurt am Main

Erklärung zur Abschlussarbeit

gemäß § 34, Abs. 16 der Ordnung für den Masterstudiengang Informatik vom 17. Juni 2019

Hiermit erkläre ich, Huy Luong

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst.

Ebenso bestätige ich, dass diese Arbeit nicht, auch nicht auszugsweise, für eine andere Prüfung oder Studienleistung verwendet wurde.

Zudem versichere ich, dass die von mir eingereichten schriftlichen gebundenen Versionen meiner Masterarbeit mit der eingereichten elektronischen Version meiner Masterarbeit übereinstimmen.

Frankfurt am Main, den 21. Dezember 2022

Huy Luong

Inhaltsverzeichnis

Akronyme **VII**

Glossar **IX**

1 Einleitung **1**
1.1 Relevanz . . . 1
1.2 Aufgabenstellung . . . 2
1.3 Struktur der Arbeit . . . 3

2 Theoretische Grundlagen **5**
2.1 Neuronale Netze . . . 5
2.1.1 Neuron . . . 5
2.1.2 Klassifikation . . . 6
2.1.3 Künstliche Neuronale Netze . . . 7
2.1.4 Maschinelles Lernen . . . 11
2.1.5 Backpropagation und Gradientenabstieg . . . 12
2.1.6 Convolutional Neural Network . . . 14
2.1.7 Pooling . . . 16
2.1.8 Dropout . . . 17
2.1.9 Mask R-CNN . . . 17
2.2 Prinzipien/Arten/Techniken der Bildannotation . . . 21
2.2.1 Bildannotationstypen . . . 21
2.2.2 Bildannotationstechniken . . . 23
2.3 Testmetriken . . . 23
2.3.1 Loss . . . 24
2.3.2 Precision, Recall & Mean Average Precision . . . 24

3 Konzept und Implementierung **29**
3.1 Vorbetrachtung . . . 29
3.2 Verwendete Hardware . . . 30
3.3 Verwendete Software/Tools . . . 30

3.3.1 NVIDIA Treiber 30
3.3.2 CVAT (Computer Vision Annotation Tool) 30
3.3.3 Docker 31
3.3.4 Tensorflow 32
3.3.5 Keras 32
3.3.6 Mask R-CNN 32
3.4 Implementierung 32
3.4.1 Erhebung und Wahl der Daten 34
3.4.2 Annotation 39
3.4.3 Mask R-CNN 40

4 Evaluation und Resultate **43**
4.1 Testergebnisse 43

5 Diskussion **61**

6 Resümee und Fazit **65**
6.1 Ausblick 66

Anhang **70**

Literaturverzeichnis **73**

Abbildungsverzeichnis

1.1 Unterschiedliche Darstellungen der Athena; Links: cnt-coin-32389-p-rez-CN8454 , Rechts: cnt-coin-32997-p-rezCN8497 2

2.1 Aufbau eines Neurons Quelle: [KBB+15], S. 10. 6
2.2 Angepasstes Schema einer McCulloch-Pitts-Zelle. Quelle: [Lan11] . . . 7
2.3 Aufbau eines Perzeptrons 8
2.4 Gängige Aktivierungsfunktionen. Quelle: [Mou] 9
2.5 Ein rekurrentes Netz: Es existieren Knoten (Neuronen) die eine Rückkopplung besitzen. Quelle: [DCK17] 10
2.6 Ein Fully Connected Layer. Quelle: [DCK17] 10
2.7 Unterschiedliche Lernraten und ihre Auswirkungen 13
2.8 Schematische Darstellung eines Filters, welcher mittels einer Faltungsoperation mit dem Eingabebild, einen Wert in der Ausgabe generiert. 15
2.9 Da im betrachteten Fenster zusätzlich zwei Weiße Pixel oben rechts, sowie unten links existieren, ist der Wert in der Feature-Map geringer, da der Filter weniger zu dem Bildausschnitt passt. 15
2.10 Mit jedem Filter wird eine Faltung durchgeführt, sodass für jeden Filter eine Feature-Map entsteht. Quelle: [noa] 16
2.11 Region Proposals werden mittels Selective Search generiert. Quelle: [UGS13] 17
2.12 Aufbau des R-CNN Modells. Quelle: [Gan18] 18
2.13 Schematische Darstellung des Fast R-CNN Modells. Quelle: [Gir15] . . 19
2.14 Schematische Darstellung des Faster R-CNN Modells. Quelle: [RHGS15] 19
2.15 Schematische Darstellung des Mask R-CNN Modells. Quelle: [HGDG17] 20
2.16 Semantic Segmentation mit 3 unterschiedlichen Klassen Quelle: [Han08] 22
2.17 Annotationstypen im Vergleich. Quelle: [CLO20] 22
2.18 Klassifizierungskennzahlen 25
2.19 Intersection over Union 27

3.1 Ablaufdiagramm dieser Masterthesis 29
3.2 Verschiedene Automatisierungsgrade der Annotation. Quelle: [SJZ21] 31

3.3 Docker Container auf einem Host-System Quelle: [noa21], S. 10. . . . 31
3.4 Erhebung der Daten 34

4.2 Im Gegensatz zur Münze (a), auf dem kein Motiv entdeckt wurde, wurde die Schlange auf Münze (b) korrekt klassifiziert, auch wenn die Maske das Motiv nicht optimal abdeckt. 45
4.3 Zwei unterschiedliche Motive des Löwen 45
4.5 Beispiele der Klassen `Sitting-Person` und `Standing-Person` 47
4.10 links: cnt-coin-18128-o-rez-CN4868 mitte: cnt-coin-18091-p-rez-CN5018 rechts: cnt-coin-45919-p-rez -CN20086 51
4.11 links: cnt-coin-18037-o-rez-CN11099 mitte: cnt-coin-18035-o-rez-CN11094 rechts: cnt-coin-17996-o -rez-CN10968 51
4.12 links:cnt-23453-CN11108 rechts:cnt-40126-CN12632 52
4.13 Loss per Epoch der beiden Modelle zu den Münzstätten Pergamon und Perinthos im Vergleich 52
4.14 Rechts:cnt-coin-2076-p-rez-CN1087, links:cnt-coin-17291-p-rez-CN8462 54
4.15 cnt-coin-4331-p-rez-CN1345 54
4.16 links: cnt-coin-4796-o-rez-CN3072 rechts: cnt-coin -13164-o-rez-CN3007 55
4.17 cnt-coin-1984-p-rez-CN2035 55
4.18 cnt-coin-2218-o-rez-CN1387 56
4.19 links: cnt-coin-2036-p-rez-CN1081 rechts:cnt-coin-2132-p-rez-CN1340 56
4.20 links: cnt-coin-4789-o-rez-CN3136 rechts:cnt-coin-2621-p-rez-CN2477 . 57
4.21 links: cnt-coin-9277-p-rez-CN2589 rechts:cnt-coin-2934-p-rez-CN2587 57
4.22 links: cnt-coin-18025-o-rez-CN11078 rechts:cnt-coin-5378-o-rez-CN2087 58
4.23 links: cnt-coin-17479-o-rez-CN4650 rechts:cnt-coin-4746-o-rez-CN2522 58

5.1 CN Coin 6649 61
5.2 CN Coin 6493 61
5.3 CN Coin 15705 62

Tabellenverzeichnis

3.1 Die zehn häufigsten Tiere, die im Webportal *Corpus Nummorum* aufgelistet sind. 35
3.2 Datensatz der stehenden und sitzenden Person 36
3.3 Datensatz mit drei unterschiedlichen Portraitarten 36
3.4 Datensatz aus der Münzstätte Pergamon 37
3.5 Limitierter Datensatz 37
3.6 Datensatz aus der Münzstätte Perinthos 38

4.1 Loss und mAP Werte über 15 Epochen 44
4.2 Precision, Recall und F1-Score der einzelnen Klassen bei 60 Testbildern 44
4.3 `loss` und mAP Werte der Modelle a.) und b.) 46
4.4 Precision, Recall und F1-Score der beiden Klassen `Standing Person` und `Sitting Person`. 46
4.5 Precision, Recall und F1-Score der Klassen `Left`, `Right` und `Front`. . 47
4.6 `loss`-Werte an unterschiedlichen Epochen 49
4.7 `loss`-Werte bei unterschiedlicher Anzahl an Trainingsbildern 49
4.8 Precision, Recall und F1-Score der 9 Klassen die aus dem Datensatz der Münzstätte Pergamon stammen. 50
4.9 `loss` und mAP Werte der beiden Modelle zur Erkennung der Münzstätten Pergamon und Perinthos 53
4.10 Precision, Recall und F1-Score der 13 Klassen die aus dem Datensatz der Münzstätte Perinthos stammen 53
4.11 Klassifizierungen des Modells e.) auf Bildern der Münzstätte Pergamon 57

Akronyme

CNN Convolutional Neural Network. 18

R-CNN Region-based Convolutional Neural Network. 17

SVM Support Vector Machine. 18

Glossar

Bounding-Box Rechteckiges Fenster, welches die Position von Objekten auf Bildern markiert. 17

Centroids Mittelpunkt der eine Kreisfläche definiert. 21

Ground Truth Realität auf die ein Machine Learning Model trainiert wird, beziehungsweise Trainings/Testdaten von denen angenommen wird, dass der Informationsgehalt der gewünschten Wahrheit entspricht. 21

Okklusion Überdeckung von Objekten. 23

Overfitting Neuronales Netz passt sich zu stark den Trainingsdaten an und ist nicht im Stande zu generalisieren und Daten zu klassifizieren die nicht in den Trainingsdaten enthalten sind. 11, 17

Reinforcement Learning Machine learning Ansatz bei dem ein Modell ohne gelabelte Daten trainiert wird, sondern über Belohnungspunkte. 11

Supervised Machine Learning Machine learning Ansatz bei dem ein Modell mittels gelabelten Daten trainiert wird, um anschließend einen Input auf einen Output zuzuweisen. 11, 12, 30

Underfitting Neuronales Netz ist nicht im Stande erfolgreich zu klassifizieren. 10

Unsupervised Machine Learning Machine learning Ansatz bei dem ein Modell ohne gelabelte Daten trainiert wird, um anschließend einen Input auf einen Output zuzuweisen. 11

1 Einleitung

In dieser Arbeit treffen das Gebiet der Numismatik mit dem Gebiet der Computer Vision aufeinander.

Die Numismatik beschäftigt sich mit dem Thema Geld und dessen Geschichte. Aufgrund des Alters antiker Münzen, sind viele dieser Münzen beim Fund durch ihren schlechten Erhaltungszustand nur schwer zu identifizieren. Einerseits liegt das zum Teil an der altersbedingten Korrosion der Münzen und andererseits an der Prägequalität der Münzen selbst.

Mithilfe heutiger Technologien stehen Möglichkeiten zur Verfügung, Numismatiker und Numismatikerinnen bei der Klassifizierung von antiken Münzen zu unterstützen. Mit wachsender Datenmenge steigt der manuelle Aufwand, Münzen zu klassifizieren und einer gegebenenfalls bereits vorhandenen Klasse zuzuordnen. Die zunehmende Bedeutung künstlicher neuronaler Netze im Bereich der Klassifizierung bietet diverse Möglichkeiten zur Automatisierung der Klassifikation antiker Münzen.

Ziel dieser Arbeit ist es mittels Machine Learning Methoden aus dem Bereich der Computer Vision Motive auf unterschiedlichsten Münzen zu erkennen und diese somit grobgranular zu klassifizieren. Im Laufe der Arbeit werden unterschiedliche Modelle trainiert und evaluiert.

Ein großer Bestandteil dieser Arbeit ist die Annotation der Motive auf Münzen, welche wiederum als Input-Daten für das Klassifizierungsmodell genutzt werden.

Zudem wird die Wahl des Tools, mit dem diese Annotationen durchgeführt werden, in der folgenden Ausarbeitung erörtert.

1.1 Relevanz

Machine Learning Algorithmen finden heutzutage in vielen Bereichen ihre Anwendung. Das autonome Fahren, die Gesichtserkennung, Spam-Filter und die medizinische Diagnose sind nur wenige Beispiele für die erfolgreiche Verwendung von Machine Learning Algorithmen [Bal15][FHY19][GC09][HE20].

Somit bietet sich an, Machine Learning Algorithmen auch im Bereich der Numismatik einzusetzen. Besonders das Feld Computer Vision hat das Potenzial die Analyse und Klassifikation von antiken Münzen zu unterstützen.

Die Performanz des Modells zur Erkennung der antiken Münzen ist abhängig von den Input-Daten mit denen es trainiert wird. Dabei ist neben der Quantität der Daten auch

die Qualität von Relevanz. Bereits die Prägequalität der Münzen stellt die erste Herausforderung für das Modell dar. Die Qualität der Münzen war in der Vergangenheit aufgrund technologischer Einschränkungen nicht mit der heutigen Prägequalität zu vergleichen, weshalb die gleiche Prägung auf verschiedenen Münzen eine teilweise sehr hohe Variabilität aufweisen kann. Des Weiteren litten die meisten antiken Münzen im Laufe der Zeit an Korrosion und Abrieb.

Das in dieser Arbeit entwickelte Klassifizierungsmodell soll den Prozess der Münzklassifizierung maschinell unterstützen. Die Idee des Einsatzes von Computer Vision im Bereich der Numismatik ist dabei keine Neuheit. Algorithmen aus dem Bereich der Computer Vision, wie beispielsweise *SIFT* und *SURF*, wurden bereits zur Erkennung von antiken Münzen verwendet [ZKZ07][ZKS08].

1.2 Aufgabenstellung

Ziel dieser Arbeit ist es, unter Nutzung eines geeigneten Machine Learning Frameworks, eine adäquate Klassifizierung von Motiven auf antiken Münzen der Datenbank von *Corpus Nummorum* zu entwickeln.

Die Klassifizierung nach der Datenbank von Corpus Nummorum ist teilweise sehr feingranular. Münzen die ähnliche Motive aufweisen und sich nur in Feinheiten unterscheiden (Abb 1.1), werden in unterschiedliche Klassen eingeteilt.

Abbildung 1.1: Unterschiedliche Darstellung der Athena, die jeweils anderen Typen zugehörig sind.

Die hohe Anzahl an unterschiedlichen Klassen erschwert das Trainieren von Klassifizierungsmodellen. In dieser Arbeit soll geprüft werden inwiefern *Mask R-CNN* sich eignet, um Münzen, beziehungsweise die auf ihnen befindlichen Motive, grobgranular zu Klassifizieren, sodass zukünftige Arbeiten auf diese grobgranulare Einteilung zurückgreifen können.

Das für die Masterarbeit genutzte Framework Mask R-CNN bietet die Möglichekeit, Modelle mit unterschiedlichen Annotationsarten zu trainieren. Neben der Wahl eines geeigneten Machine Learning Frameworks, soll die Auswirkung der Annotationsarten auf die

Performanz der Modelle untersucht werden. Dazu werden diverse Modelle erstellt, deren Klassifizierungsergebnisse anschließend evaluiert werden.

Zusätzlich zur Implementierung der Klassifizierungsmodelle werden annotierte Datensätze benötigt, welche für die Trainingsphase der Modelle eingesetzt werden.

Die Wahl sowie die manuelle Annotation dieser Daten sind ebenfalls Bestandteil dieser Arbeit.

1.3 Struktur der Arbeit

Die Masterarbeit ist wie folgt strukturiert:

Das vorangegangene Kapitel führte in die Thematik dieser Arbeit ein und gab einen Überblick über die Relevanz, sowie die Aufgabenstellung dieser Masterarbeit.

Das folgende Kapitel beschäftigt sich mit den theoretischen Grundlagen der Bilderkennung mittels künstlichen neuronalen Netzen. Dafür wird auf die Entwicklung künstlicher neuronaler Netze eingegangen, sowie der grundlegende mathematische Hintergrund beleuchtet. Darauf folgt die Entwicklung von *Convolutional Neural Networks* hin zu den in dieser Arbeit verwendeten *Mask R-CNNs*.

Im zweiten Abschnitt des Kapitels folgt eine Übersicht verschiedener Annotationsarten, die für die Bilderkennung verwendet werden.

Der letzte Abschnitt des Kapitels beinhaltet die Definitionen der verwendeten Evalutationsmetriken.

Das dritte Kapitel gibt einen Überblick über die Software, welche für die Umsetzung der im zweiten Kapitel besprochenen Machine Learning Konzepte verwendet werden. Es wird genauer auf das Tool eingegangen, welches für die Annotation, der in dieser Arbeit genutzten Daten, verwendet wird. Anschließend werden die unterschiedlichen Versuchsreihen aufgelistet und beschrieben, sodass in den folgenden Kapiteln auf die einzelnen Versuche referenziert werden kann. Im selben Kapitel werden die verwendeten Daten, für die bereits definierten Versuchsreihen angegeben. Die Daten werden anhand der Klassen tabellarisch aufgeschlüsselt.

Anschließend werden im darauffolgenden Kapitel die verschiedenen Modelle der Versuchsreihen getestet und deren Ergebnisse dokumentiert und evaluiert. Dazu werden die zuvor erwähnten Evaluationsmetriken verwendet, um die Testergebnisse zu interpretieren.

Es folgt das Kapitel, in dem die Ergebnisse analysiert, sowie Probleme und Erkenntnisse aus der Umsetzung diskutiert werden.

Zuletzt folgt eine Zusammenfassung, ein Fazit, sowie mögliche Ausblicke die aus dieser Masterarbeit hervorgehen.

2 Theoretische Grundlagen

Für die Lösung des Problems, welches diese Arbeit als Thematik behandelt, ist das Gebiet der Computer Vision von fundamentaler Bedeutung. Mittels Objekterkennung sollen verschiedene Motive auf den betrachteten antiken Münzen erkannt und klassifiziert werden. Dabei wird ein neuronales Netz verwendet, welches auf die Erkennung der Motive, die sich auf den Münzen befinden, trainiert wird. Konkreter wird das Konzept von Mask R-CNN[HGDG17] verwendet, welches eine Erweiterung von Convolutional Neural Networks ist.
Die Architektur, sowie die Funktionsweisen beider Modelle werden im Folgenden genauer erläutert. Des Weiteren wird der theoretische Hintergrund zu (künstlichen) neuronalen Netzen nach [Mur12],[GBC16] sowie [KBB+15] beleuchtet.

2.1 Neuronale Netze

Für einen Menschen ist das Unterscheiden einer Katze und eines Hundes relativ einfach. Um mit Maschinen ebenfalls Objekte klassifizieren zu können, werden künstliche neuronale Netze verwendet. Diese haben die neuronalen Netze im menschlichen Gehirn als biologisches Vorbild.
Das menschliche Hirn besteht aus über 100 Milliarden Neuronen [KBB+15]. Neuronen sind über Synapsen mit tausenden anderen Neuronen verbunden und kommunizieren chemisch über Neurotransmitter miteinander. Signale innerhalb eines Neurons werden elektrisch weitergeleitet.

2.1.1 Neuron

Ein Neuron besteht aus dem Zellkörper (Soma), mehreren Eingängen (Dendriten) und dem Ausgang (Axon). Neuronen werden durch ihr Axon über die Dendriten anderer Neuronen verbunden (Synapsen).

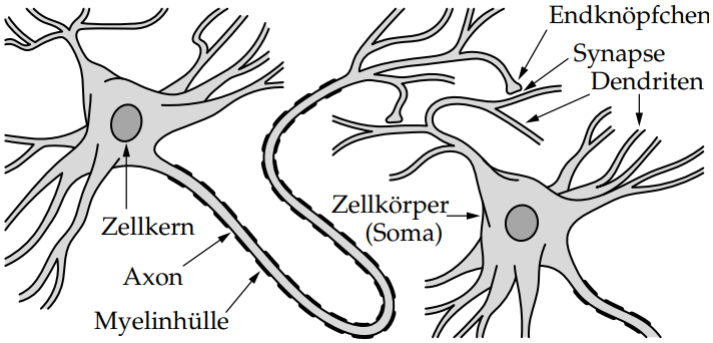


Abbildung 2.1: Aufbau eines Neurons

Jedes Neuron kann durch seine Dendriten Neurotransmitter aufnehmen, die elektrische Reize auslösen. Diese werden am Zellkörper akkumuliert. Innerhalb des Zellkörpers kann es dadurch zu einer Steigerung oder Erniedrigung des elektrischen Potenzials kommen. Wenn die Änderung des elektrischen Potenzials einen Schwellenwert übersteigt, wird dieses (auch *Aktionspotenzial*) über das Axon weitergeleitet (gefeuert), was bewirkt, dass Neurotransmitter ausgeschüttet werden, die wiederum dazu führen, dass sich das elektrische Potenzial der Neuronen ändert, die mit dem Ausgangsneuron verbunden sind.

Ausgelöst wird das Aktionspotenzial niemals teilweise, sondern entweder ganz oder gar nicht. Das Aktionspotenzial wird an den Synapsen gewichtet, sodass das weitergeleitete Signal verstärkt oder geschwächt wird.

Auf diese Art und Weise werden vom Gehirn Informationen verarbeitet. Das neuronale Netz wird durch neue Informationen und Erfahrungen immer wieder umstrukturiert. Durch die stetige Umstrukturierung lernen Neuronen in den entscheidenden Momenten zu feuern oder inaktiv zu bleiben. Die Gewichtungen der Synapsen werden dabei angepasst. Abgesehen von der erwähnten Anpassung der Neuronen und ihrer Verbindung zu anderen Neuronen, gibt es weitere Prozesse, wie beispielsweise die Entstehung neuer Synapsen oder der Wegfall bestehender, welche für das menschliche Lernen mitverantwortlich sind.

2.1.2 Klassifikation

Bei einer Klassifikation geht es generell darum eine Eingabe x_i aus der Eingabemenge $X = \{x_1, x_2, ..., x_{n-1}, x_n\}$ mit $i \in \{1, ..., n\}$ und $n \in \mathbb{N}$ einer Klasse y_j aus der Ausgabemenge $Y = \{y_1, y_2, ..., y_{m-1}, y_m\}$ mit $j \in \{1, ..., m\}$ und $m \in \mathbb{N}$ zuzuordnen.
Bezogen auf eine Bilderkennung sollte ein auf einem Bild vorhandenes Motiv einer gesuchten Klasse zugeordnet werden.
Künstliche neuronale Netze sollten demnach im Idealfall möglichst viele Eingaben korrekt der Ausgabemenge zuordnen.
Formell ist das Ziel eine Funktion zu approximieren, die diese Zuordnung abbilden kann.

Man nehme an, die Zuordnung wird folgendermaßen mathematisch ausgedrückt:

$$y = f(x) \tag{2.1}$$

Im Trainingsprozess soll dann die unbekannte Funktion mithilfe von gelabelten Datensätzen approximiert werden. Damit soll das Modell im Stande sein, Eingaben zu klassifizieren, die ihm bis dato unbekannt waren.

2.1.3 Künstliche Neuronale Netze

Im Jahre 1943 haben Warren McCulloch und Walter Pitts erstmals ein vereinfachtes Modell eines künstlichen Neurons entworfen und beschrieben [MP43]. Die nach ihnen benannte *McCulloch-Pitts-Zelle* kann nur binäre Signale verarbeiten, und auch nur binäre Signale als Ausgabe erzeugen. Sie verfügt über diverse Eingangsleitungen, sowie mehrere hemmende Leitungen. Zudem besitzt sie einen reellwertigen Schwellenwert.

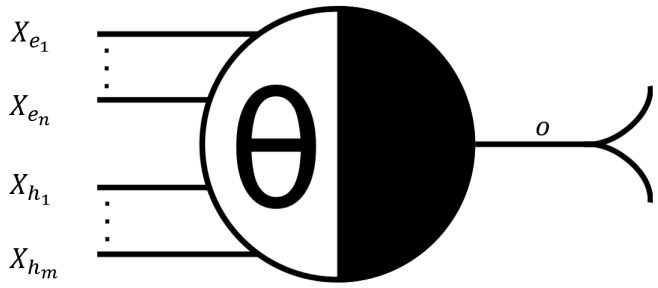


Abbildung 2.2: Schema einer McCulloch-Pitts-Zelle mit den Eingängen x_{e_i} mit $i \in \{1, ..., n\}$, den hemmenden Eingängen x_{h_j} mit $j \in \{1, ..., m\}$, dem Schwellewert Θ, sowie dem Ausgang o.

Eine McCulloch-Pitts-Zelle generiert eine 1 als Ausgabe, wenn keiner der hemmenden Eingänge 1 ist, sowie die Summe der anderen Eingangswerte den Schwellenwert übersteigt. In allen anderen Fällen generiert die McCulloch-Pitts-Zelle eine 0.
1958 publizierte Frank Rosenblatt auf der Grundlage McCullochs und Pitts das Perzeptron-Modell [Ros58]. Im Vergleich zu den McCulloch-Pitts-Zellen haben einfache Perzeptronen keine hemmenden Eingänge. Stattdessen besitzen sie anpassbare Gewichte, welche mit den Eingangswerten multipliziert werden.

Die Schwäche des Perzeptrons liegt allerdings darin, dass es nur für Klassifikationen genutzt werden kann, dessen Klassen linear separierbar sind [MP69]. Um diese Schwäche zu überwinden, wurden künstlich neuronale Netze verwendet, die aus mehreren Schichten an Neuronen bestehen (engl.: Multilayer Perceptron [GBC16].s.225). Das Perzeptron nach Rosenblatt ist bis heute Grundbaustein künstlicher neuronaler Netze.

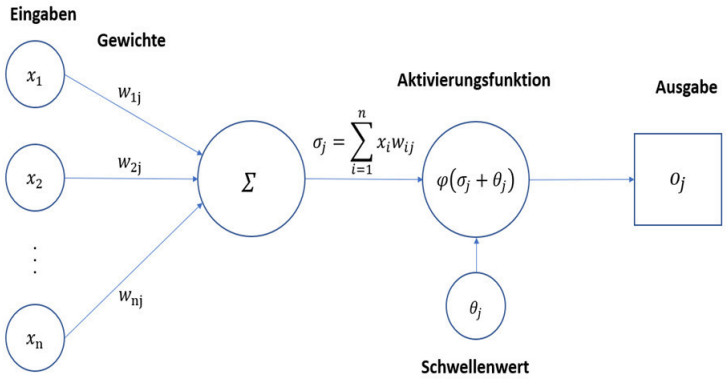


Abbildung 2.3: Aufbau eines Perzeptrons

Abb 2.3 zeigt den schematischen Aufbau eines Perzeptrons. Die Eingaben x_i mit $i \in \{1, ..., n\}$ können zum einen aus einer Beobachtung oder Messung entstammen, zum anderen aber auch aus den Ausgaben anderer Neuronen generiert worden sein. Die Gewichte w_{ij} (wobei j für das betrachtete Neuron steht) verstärken oder hemmen die Eingaben x_i und werden anfangs zufällig gewählt.
Für eine Klassifikation wird als erstes die gewichtete Summe σ_j der Eingabewerte berechnet:

$$\sigma_j = \sum_{i=1}^{n} x_i w_{ij} \tag{2.2}$$

Der Ausgabewert o_j ergibt sich anschließend aus σ_j, der Aktivierungsfunktion φ und dem Schwellenwert θ_j

$$o_j = \varphi(\sigma_j + \theta_j) \tag{2.3}$$

Für die Aktivierungsfunktion φ können unterschiedliche Funktionen verwendet werden. Zu unterscheiden sind allerdings nicht differenzierbare Funktionen von differenzierbaren Funktionen. Für die Ausführung des Lernalgorithmus werden letztere Funktionen benötigt. Häufig verwendete Aktivierungsfunktionen sind die Sigmoidfunktion, ReLU (engl.: Rectified Linear Unit) und der Tangens Hyperbolicus (Abb 2.4).

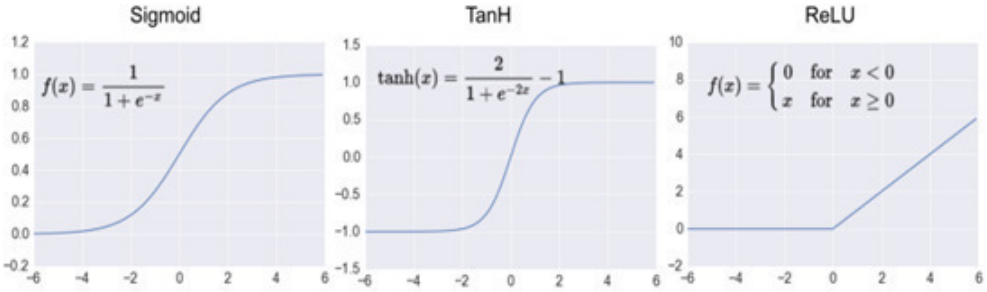


Abbildung 2.4: Gängige Aktivierungsfunktionen

Die Neuronen sind untereinander verbunden und werden in Schichten angeordnet. Dabei wird zwischen drei Schichten unterschieden:

1. Die erste Schicht wird Input Layer genannt und repräsentiert die Eingabe.
2. Die zweite Schicht wird Hidden Layer genannt und besteht aus einer Anzahl von Zwischenschichten.
3. Die letzte Schicht ist der Output Layer, der die Ausgabe repräsentiert und das Klassifikationergebnis liefert.

Neuronen können unterschiedlich verbunden sein. Dies wird als Netzwerktopologie bezeichnet. Die Netzwerktopologie beschreibt im Allgemeinen die Anzahl und die Art und Weise wie Neuronen im Netzwerk miteinander verbunden sind.

Ein künstliches neuronales Netz lässt sich als Graph darstellen, wobei die Neuronen als Knoten repräsentiert sind und die Verbindungen der Neuronen als Kanten. Typische Strukturen sind Feedforward-Netze, bei denen eine Schicht immer nur mit der nächsten Schicht verbunden ist, sowie rekurrente Netze, bei denen es eine Rückkopplung gibt, die durch rückwärts gerichtete Kanten typisiert ist.

Feedforward-Netze zeichnen sich dadurch aus, dass der Input nur in eine Richtung weitergegeben wird. Die Ausgabe der vorangegangenen Schicht ist die Eingabe der kommenden Schicht.

Im Gegensatz dazu besitzen rekurrente Netzwerke aufgrund der rückwärts gerichteten Kanten (Abb 2.5) Zyklen. Die Möglichkeiten der Weitergabe der Ausgabe eines Neurons sind auch vielfältiger. So ist es nicht nur möglich, dass ein Neuron seine Ausgabe an ein Neuron übergibt, welches sich in einer folgenden Schicht befindet, sondern es ist ebenfalls möglich, dass die Ausgabe an ein Neuron in vorangegangener Schicht, in der selben Schicht, oder sogar auf das selbige Neuron erneut gelangt.

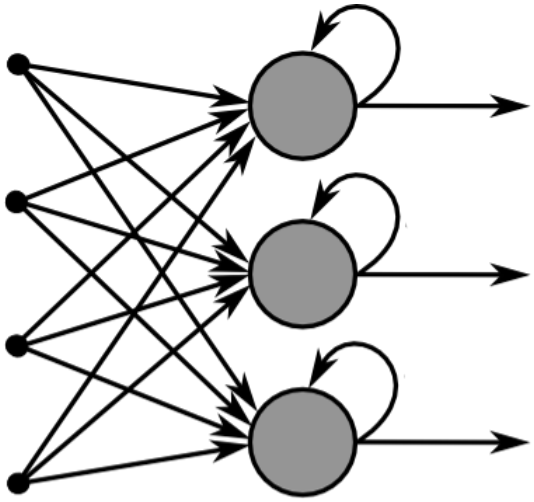


Abbildung 2.5: Ein rekurrentes Netz: Es existieren Knoten (Neuronen) die eine Rückkopplung besitzen.

Ein weiterer wichtiger Faktor ist neben der Art der Verbindungen, die Menge der Verbindungen. Schichten werden häufig voll vernetzt (*Fully Connected Layer*). Das bedeutet ein Knoten ist mit jedem anderen Knoten der folgenden Schicht verbunden, da optimale Verbindungen nur schwer nachvollziehbar sind. Der Nachteil, welcher sich dadurch ergibt, ist die Komplexität des Netzes aufgrund von redundanten Verbindungen, welcher dazu führt, dass der Trainingsprozess verlangsamt wird.

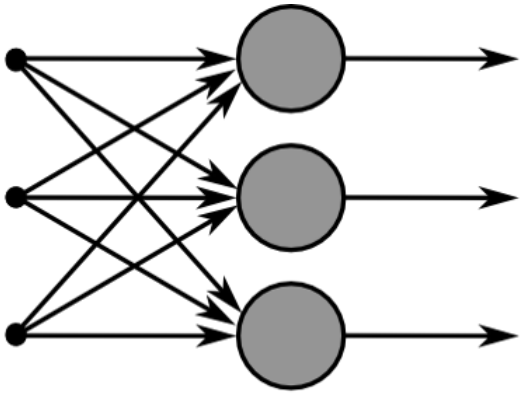


Abbildung 2.6: Ein Fully Connected Layer: Jeder Eingangsknoten ist mit jedem Neuron der nächsten Schicht über eine Kante verbunden.

Die Leistungsfähigkeit eines Netzwerks ist stark durch seine Topologie bestimmt. Die Wahl einer optimalen Netzwerkstruktur ist abhängig von der zu lösenden Problemstellung, sowie der Art und Anzahl der Eingabedaten. Ist das Netzwerk zu klein und die Struktur zu simpel, ist eine erfolgreiche Klassifikation selten zu erreichen. Sogenanntes *Underfitting*

tritt auf. Underfitting kann gelöst werden, indem der Trainingsprozess verlängert oder die Komplexität des Netzes durch Hinzufügen weiterer Schichten gesteigert wird.
Overfitting ist der Gegensatz zum Underfitting. Netze die zu viele Neuronen besitzen und dadurch zu komplex oder zu lange trainiert worden sind, sind nicht im Stande zu generalisieren. Das bedeutet sie sind im Stande Daten aus dem Trainingsdatensatz korrekt zu klassifizieren, können allerdings durch die fehlende Generalisierung keine neuen Daten klassifizieren. Das Netz lernt nicht nach allgemeinen Merkmalen Vorhersagen zu treffen, sondern passt sich den Daten im Trainingsprozess an. Es lernt die Trainingsdaten auswendig [Sch14].

2.1.4 Maschinelles Lernen

Damit ein künstliches neuronales Netz Eingaben klassifizieren kann, muss es im Voraus trainiert werden und lernen die möglichen Klassen zu erkennen. Der Lernprozess eines künstlichen neuronalen Netzes besteht darin, die Gewichte der Verbindungen zwischen den Neuronen iterativ anzupassen, sodass das Netz bei einer beliebigen Eingabe, eine möglichst passende Ausgabe erzeugt.
Dabei gibt es drei verschiedene Arten wie ein neuronales Netz trainiert wird:

1. Die erste Trainingsmethode wird als *Supervised Machine Learning* bezeichnet. Grundvoraussetzung für das Supervised Machine Learning ist, dass für eine Menge an Eingabewerten die jeweiligen passenden und gewünschten Ausgabewerte zugeordnet sind. Der passende Ausgabewert zu einem Eingabewert wird als *Label* bezeichnet.
   Bei jeder Iteration wird für eine Eingabe die Diskrepanz zwischen dem Label der Eingabe und der Ausgabe des neuronalen Netzes bestimmt. Die Gewichte der Neuronen werden in Abhängigkeit dieser Diskrepanz dahingehend verändert, dass das Netz zukünftig die gewünschte Ausgabe generiert.
   Ziel des Trainings ist es, dass das Netz auch im Stande ist, Daten klassifizieren zu können, mit denen das Netz nicht trainiert wurde.

   Supervised Machine Learning-Algorithmen werden aufgrund der hohen Lerngeschwindigkeit bevorzugt.

2. Im Gegensatz dazu steht das *Unsupervised Machine Learning* bei dem nur Eingabedaten übergeben werden. Das neuronale Netz versucht dann Muster und Strukturen in den Eingabedaten zu erkennen. Ein Beispiel wären *Clusteralgorithmen*, welche versuchen aus einer Vielzahl an Datenpunkten ähnliche Daten miteinander zu Gruppieren.

3. Beim *Reinforcement Learning* werden dem Netz im Gegensatz zum Supervised Machine Learning nicht dessen konkrete Fehler (Diskrepanzen) mitgeteilt. Stattdessen erhält das Netz bei richtigen Entscheidungen positive Belohnungspunkte und bei falschen Entscheidungen negative Belohnungspunkte. Das Netz muss dann ohne gerichteten Fehler

die Gewichte im neuronalen Netz anpassen, weshalb die Methode bedeutend langsamer als das Supervised Machine Learning ist.

2.1.5 Backpropagation und Gradientenabstieg

Ein verbreitetes Verfahren zum Einlernen von neuronalen Netzen ist *Backpropagation* und wird beim Supervised Learning verwendet. Bei der Backpropagation wird der Fehler zwischen Ist-Ausgabe des Netzes (Voraussage) und der Soll-Ausgabe (dem Label) berechnet, was als *loss* bezeichnet wird. Dieser wird anschließend an das Netzwerk zurückgegeben. Anschließend werden die Gewichte in Abhängigkeit des Fehlers angepasst [NRS+10].

Die Anpassung wird mittels Gradientenabstieg umgesetzt, einer Methode, die in der Numerik eingesetzt wird um Optimierungsprobleme zu lösen. Hierbei sind die Gewichte des neuronalen Netzes zu optimieren.
Für die Anwendung des Gradientenabstiegs, bedarf es einer Fehlerfunktion E, welche die Abweichung der Ist-Ausgabe von der Soll-Ausgabe berechnet. Zu bestimmen ist das Minimum dieser Funktion. Die Parameter werden so lange iterativ in Richtung des negativen Gradienten angepasst, bis keine weitere Verbesserung der Ausgabe erzielt wird.

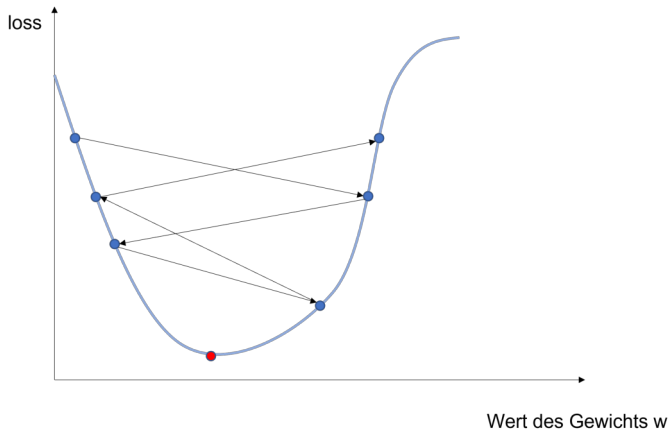
Es sei E eine beliebige Fehlerfunktion. Diese ist aufgrund der Abhängigkeit zur Ausgabe des Netzes auch abhängig von den Gewichten des Netzes. Der Gradient ΔE wird bestimmt durch die partielle Ableitung von E nach w_{ij}.

$$\Delta E = \frac{\delta E}{\delta w_{ij}} \tag{2.4}$$

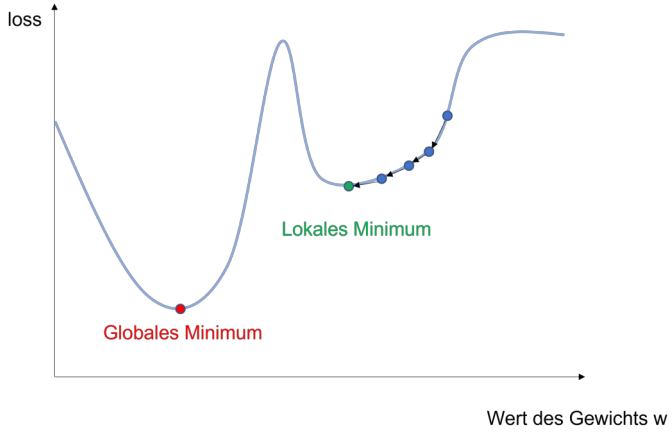
Das Gewicht wird in negativer Richtung des Gradienten geändert. Dabei wird eine sogenannte Lernrate η hinzugenommen, die den Grad der Anpassung in jeder Iteration bestimmt. Somit ergibt sich für das neue Gewicht:

$$w_{ij_{neu}} = w_{ij} - \eta \cdot \frac{\delta E}{w_{ij}} \tag{2.5}$$

Dabei kann sowohl eine zu niedrig als auch eine zu hoch gewählte Lernrate problematisch werden. Ist die Lernrate zu hoch gewählt, passen die Gewichte sich zu stark an, sodass das Minimum nicht erreicht werden kann, sondern stets übersprungen wird [Abb 2.7 (a)]. Wird die Lernrate zu niedrig gewählt, kann es passieren, dass ein lokales, statt ein globales Minimum erreicht wird [Abb 2.7 (b)].



(a) Hohe Lernrate



(b) Niedrige Lernrate

Abbildung 2.7: Unterschiedliche Lernraten und ihre Auswirkungen

Im Kontext des Trainings von neuronalen Netzen sind zwei Begriffe zu definieren, die im weiteren Verlauf dieser Arbeit genutzt werden. Die zwei Begriffe lauten *Batch* und *Epochs*. Der Wert des Batches (auch *Batch-size*) definiert, nach wie vielen Eingaben und den dazugehörigen Ausgaben des Netzes, eine Anpassung der Gewichte erfolgt. Je kleiner die Batch-size ist, desto höher ist auch die Anzahl an Gewichtsanpassungen in jeder Trainingsiteration.

Der Begriff Epochs gibt an, wie oft der gesamte Trainingsdatensatz durchlaufen wird, bis der Trainingsprozess beendet wird. Sobald über alle Eingaben des Trainingsdatensatzes iteriert wurde, ist eine Epoch abgeschlossen und es beginnt ein neue Epoch bei der wieder über alle Trainingsdaten iteriert wird.

2.1.6 Convolutional Neural Network

Convolutional Neural Networks sind eine spezielle Form des mehrlagigen Perzeptrons. Sie werden oft für die Verarbeitung von Bild und Audiodaten verwendet. Sie eignen sich gut um Muster in den Eingabedaten zu erkennen, indem sie mehrschichtige Perzeptrons um Convolutional-Layers ergänzen.

Im Folgenden beziehen sich die Beschreibungen auf Eingaben die aus Bildern bestehen, da in dieser Arbeit auch letztlich mit Bilddaten gearbeitet wird. Dies ermöglicht eine anschauliche Illustration der Modelle.

Eingaben sind zumeist zweidimensionale Arrays an Pixeln (dreidimensional wenn man Farbbilder betrachtet. Jeder Farbkanal ist dabei ein zweidimensionales Array), welche die Belichtung an der jeweiligen Stelle angeben.

Convolutional-Layers bestehen aus mehreren sogenannten Filtern (auch *Featuredetector* oder *Kernel* genannt). Diese haben die Funktion bestimmte Muster zu erkennen, wie zum Beispiel horizontale Linien, vertikale Linien oder Ecken. Ein Filter ist eine Matrix mit variabler Größe. Die Werte der Matrix sind abhängig vom zu erkennenden Muster. Jeder Filter läuft einmal komplett über das Ausgangsbild, und tastet das Bild iterativ ab. Die Werte des Filters werden dabei mit den Werten des Ausschnitts des Eingangsbildes multipliziert, aufsummiert und anschließend durch die Anzahl der Zellen des Filters dividiert. Dieser Wert wird dann auf einer neuen Matrix gespeichert. Dieser Prozess wird als *Convolution* bezeichnet.

Die neu generierte Matrix, auch Feature-Map genannt, gibt mit ihren Werten an, wie stark das Muster des Filters an dieser bestimmten Stelle des Eingabebildes repräsentiert wird.

Schichten, welche sich weiter vorne im Netz befinden, tendieren dazu weniger komplexe Muster zu erkennen, während folgende Schichten sich dahingehend entwickeln komplexere und spezialisierte Muster zu erkennen [Wu17].

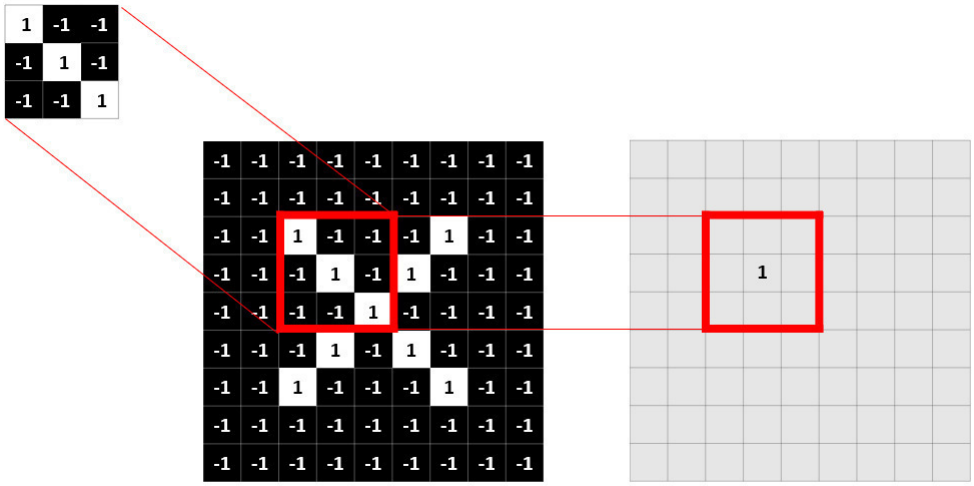


Abbildung 2.8: Schematische Darstellung eines Filters, welcher mittels einer Faltungsoperation mit dem Eingabebild, einen Wert in der Ausgabe generiert.

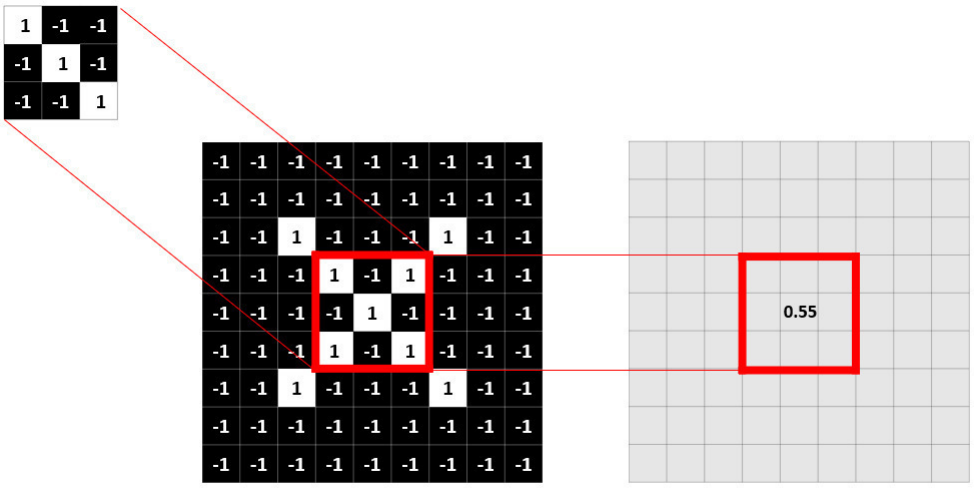


Abbildung 2.9: Da im betrachteten Fenster zusätzlich zwei Weiße Pixel oben rechts, sowie unten links existieren, ist der Wert in der Feature-Map geringer, da der Filter weniger zu dem Bildausschnitt passt.

Die Convolution (zu deutsch Faltung) wird mit einer vorgegeben Anzahl an Filtern wiederholt, sodass man am Ende aus einem Eingabebild, einen Stapel (Abb 2.10) an gefilterten Bildern erhält, welche unterschiedliche Features hervorheben.

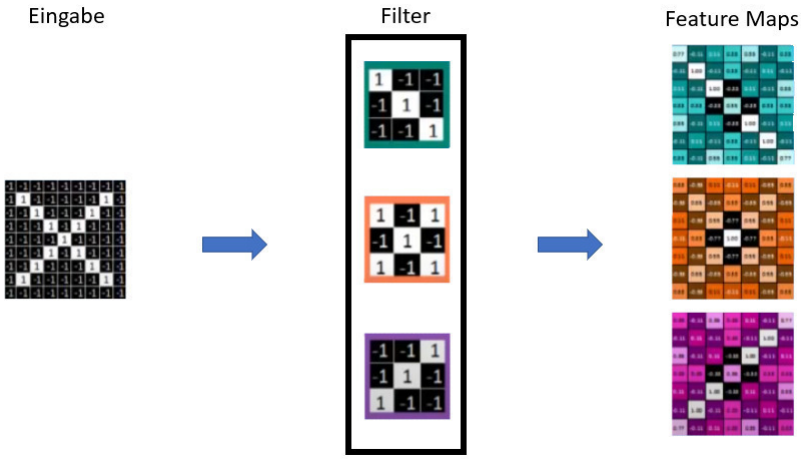


Abbildung 2.10: Mit jedem Filter wird eine Faltung durchgeführt, sodass für jeden Filter eine Feature-Map entsteht.

2.1.7 Pooling

Die Komplexität des Netzes nimmt aufgrund der Menge an Feature-Maps stark zu. Um dem entgegen zu wirken wird das sogenannte *Pooling* angewandt. Beim Pooling wird die Dimension der Feature-Maps reduziert. Zunächst wird dafür ein quadratisches Fenster definiert. Dieses läuft ähnlich wie beim Anwenden der Filter, iterativ über die Eingabe, in unserem Fall die Feature-Maps. Innerhalb dieses Fensters wird der größte Wert übernommen und in eine neue Matrix geschrieben, was auch als *Max-Pooling* bezeichnet wird.

Als Ergebnis erhält man eine kleinere Feature-Map, die weiterhin die relevanten Muster enthält, allerdings von den Dimensionen her kleiner ist.

Variable Größen sind hier zum einen die Größe des Fensters und zum anderen die sogenannte *Stride*, die bestimmt in welchen Schritten das Fenster über die Eingabe läuft. Zusätzlich zum Stride ist *Padding* eine weitere variable Größe. Durch das Padding wird die Eingabe durch Nullen rundum erweitert, sodass das Fenster mehr Iterationen durchläuft. Die Dimension der Ausgabe bestimmt sich letztendlich durch die Eingabedimension, Stride, Padding und der Größe des Kernels.

$$\text{Ausgabedimension} = \lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor \tag{2.6}$$

Wobei $n_h \times n_w$ für die Eingabegröße, $k_h \times k_w$ für die Größe des Kernels, p_w/p_h für die Anzahl an Padding ausgedrückt in Zeilen und Spalten, und s_w/s_h für die Schrittgröße in

horizontaler und vertikaler Richtung steht.

2.1.8 Dropout

Um zu verhindern, das beim Trainieren eines neuronalen Netzes Overfitting auftritt, werden in den Trainingsiterationen jeweils eine Menge an Neuronen in den Schichten zufällig deaktiviert, was als Dropout [SHK+14] bezeichnet wird. Die Dropout-Rate ist dabei die Menge an Neuronen die jeweils deaktiviert werden.

2.1.9 Mask R-CNN

Mask R-CNN ist das Machine Learning Modell, welches im weiteren Verlauf dieser Arbeit zur Motiverkennung verwendet wird. Es baut auf dem Modell des CNN auf. Es ist ein Modell, welches zum einen Objekte auf Bildern klassifiziert und zum anderen die klassifizierten Instanzen zusätzlich segmentiert.

Im Folgenden wird auf die Entwicklung von CNNs hin zu Mask R-CNNs eingegangen, sowie ein Einblick über die Funktionsweise der unterschiedlichen Modelle gegeben.

R-CNN:
Im ersten Entwicklungsprozess entstanden durch Weiterentwicklung von CNNs die *R-CNNs* [GDDM13]. Diese haben im Gegensatz zu herkömmlichen CNNs, welche ein Bild als Ganzes klassifiziert haben, zusätzlich das Objekt auf dem Bild lokalisiert und mittels einer *Bounding-Box* markiert.

Zunächst werden Regionen auf dem Eingabebild vorgeschlagen, auf welchen sich gegebenenfalls Objekte befinden könnten, was als *Region Proposals* bezeichnet wird. Dies geschieht mittels *Selective Search*. Bei Selective Search werden auf Pixelebene verwandte Nachbarn, welche beispielsweise ähnliche Farben oder Texturen besitzen, miteinander kombiniert und als Region zusammengefasst. Anschließend werden iterativ ähnliche Regionen zu größeren Regionen kombiniert [UGS13].

Abbildung 2.11: Region Proposals werden mittels Selective Search generiert.

Anschließend werden mittels CNNs Features auf diesen Regionen extrahiert und mittels einer SVM [MW01] klassifiziert. Zuletzt wird die Größe der Bounding-Box mit einem linearen Regressionsmodell optimiert.

Die Nachteile die sich aus dem R-CNN Modell ergeben sind eine lange Laufzeit: Jede Region Proposal muss klassifiziert werden (durschnittlich ca. 2000). Somit sind auch keine Echtzeitvoraussagen möglich.
Außerdem müssen drei unterschiedliche Modelle trainiert werden:

1. Das CNN, welches die Features erkennt.

2. Die SVM, die die Klasse des Objekts erkennt.

3. Das Regressionsmodell, welches die Bounding-Box optimiert.

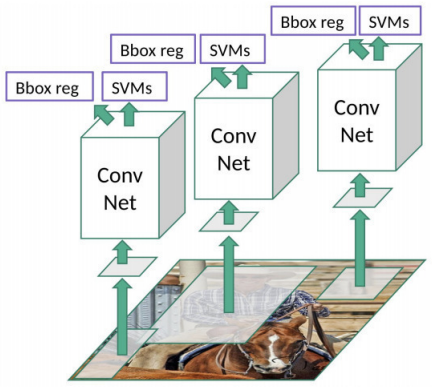


Abbildung 2.12: Illustration des Aufbaus eines R-CNN Modells

Fast R-CNN:
Fast R-CNN [Gir15] wurde vom selben Autor entwickelt, der auch schon das R-CNN entwickelte.

Bei Fast R-CNNs werden zunächst mittels Selective Search Region Proposals erstellt. Anschließend wird das Eingabebild einmal durch ein CNN geschickt, welches Feature-Maps generiert. Die Region Proposals werden daraufhin auf die Feature-Maps projiziert. Diese Projektionen, welche als Region of Interest (RoI) bezeichnet werden, laufen anschließend in ein *RoI-Pooling-Layer*, dessen Ausgabe dann durch ein Fully-Connected-Layer läuft, welches wiederum einen *RoI-Feature-Vector* erstellt. Dieser wird durch ein *Softmaxlayer* klassifiziert. Gleichzeitig werden die Koordinaten der Bounding-Boxen durch ein Regressions-Layer bestimmt.

Anstatt die Vielzahl an Region Proposals durch das CNN laufen zu lassen, wird das CNN bei Fast R-CNN nur einmal eingesetzt, um die Feature-Maps zu erstellen.

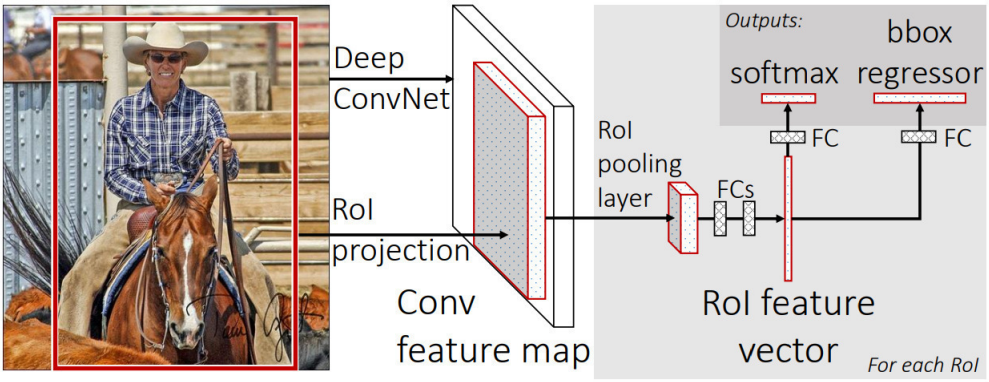


Abbildung 2.13: Schematische Darstellung des Fast R-CNN Modells

Im Gegensatz zum R-CNN Modell wird alles in einem Netzwerk trainiert. Die Geschwindigkeit des Modells wird durch die Erstellung der Region Proposals beschränkt, weshalb Fast R-CNN weiterentwickelt wurde und Faster R-CNN entstand.

Faster R-CNN:
Faster R-CNN verzichtet im Gegensatz zu seinen Vorgängern auf die Selective Search, was zu einem weiteren Geschwindigkeitsanstieg verhilft. Wie auch bei Fast R-CNN durchläuft das Eingabebild ein CNN, welches eine Feature-Map erstellt.

Anschließend wird ein separates CNN genutzt (Region Proposal Network) um die Region Proposals vorauszusagen.

Diese Region Proposals werden gleichermaßen auf die Feature-Maps projiziert und durchlaufen dann ebenfalls ein RoI-Pooling-Layer, dessen Ausgabe dann klassifiziert wird und in dem die Bounding-Box Koordinaten bestimmt werden.

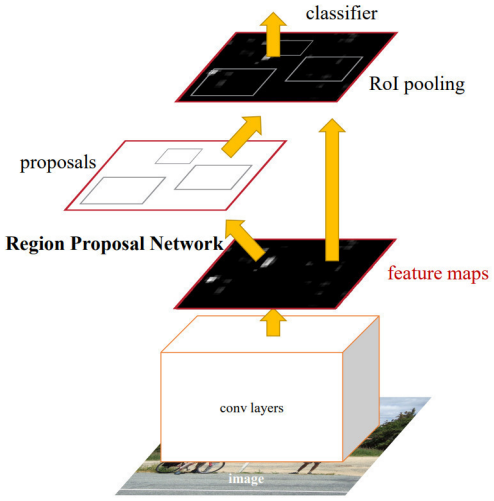


Abbildung 2.14: Schematische Darstellung des Faster R-CNN Modells

Mask RCNN:
Mask R-CNN [HGDG17] erweitert Faster R-CNN um eine pixelgenaue Klassifizierung (Image Segmentation).

Das Faster R-CNN Modell wird um ein *Fully Convolutional Network* [LSD15] erweitert, um eine Mask-Segmentation Ausgabe zu generieren, welche für einzelne Pixel entscheidet, ob es zu einem Objekt gehört oder nicht (*Binary Mask*).

Dieses Fully Convolutional Network [LSD15] läuft parallel zu der Klassifizierung und der Regression der Bounding-Box Koordinaten. Es erhält eine CNN Feature-Map als Eingabe und generiert daraus die Binary Mask als Ausgabe.

Da die Generierung von der Maske aufgrund der Genauigkeit auf Pixelebene präziser sein muss als die Generierung von Bounding-Boxen, wurde das RoI-Pooling aus dem Faster R-CNN Modell durch das *RoI-Align* verbessert.

Die Ungenauigkeit kommt zu Stande, da die Regions of Interest im Originalbild auf die kleineren Feature-Maps projiziert werden. Je nach Verhältnis der Dimensionen des Originalbilds zu den kleineren Feature-Maps werden die in Pixel gemessenen Längen des Region of Interest Fensters gerundet, sodass es zu geringen Ausrichtungsfehlern kommt.

RoI-Align nutzt bilineare Interpolation [Smi81] statt zu runden, was zu einer präziseren Ausrichtung des Region of Interest führt.

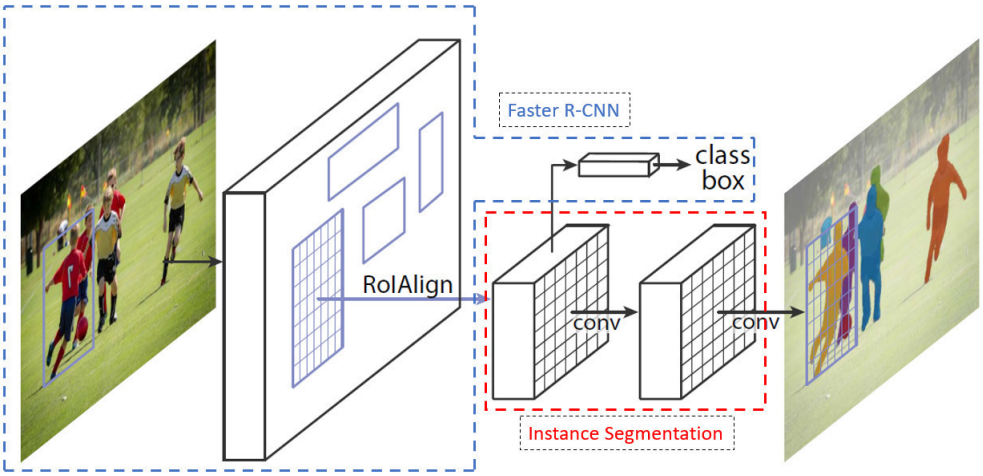


Abbildung 2.15: Schematische Darstellung des Mask R-CNN Modells

2.2 Prinzipien/Arten/Techniken der Bildannotation

Für die Trainingsphase des neuronalen Netzes bedarf es neben den Datensätzen, welche für das Training genutzt werden, notwendigerweise auch die dazugehörigen Ground Truth-Label.

Da die Problemstellung dieser Arbeit visuellen Charakters ist, besteht der Trainingsdatensatz aus Bildern. Für die Erstellung von Ground Truth-Labeln gibt es unterschiedliche Annotationstypen, sowie Annotationstechniken. Im Folgenden werden die unterschiedlichen Typen erläutert, sowie deren Anwendungsfälle besprochen.

2.2.1 Bildannotationstypen

Es gibt nach [CLO20] und [HB20] vier unterschiedliche Bildannotationstypen:

Classification:
Bei der Classification (Klassifizierung) wird dem Input als Ganzes eine Klasse zugewiesen oder eine Menge an Klassen. Es ist im Vergleich zu den anderen Bildannotationstypen die undetaillierteste. Ziel ist es lediglich die Präsenz von Klassen vorauszusagen.

Object Detection/Object Localization:
Im Vergleich zum vorherigen Typen, werden bei der Object Detection, zusätzlich zur Erkennung der auf dem Eingabebild vorhanden Klassen, eine Aussage zu ihrer Kardinalität getroffen. Des Weiteren werden Vorraussagen darüber getroffen, an welchen Positionen auf dem Eingabebild sich die klassifizierten Objekte befinden. Die Positionen werden mittels sogenannten Bounding-Boxen oder Centroids angegeben.

Semantic Segmentation:
Semantic Segmentation geht einen Schritt weiter und gibt für jeden Pixel eine Vorraussage seiner Klassenzugehörigkeit ab. Damit wird eine Voraussage über die Anwesenheit einer Klasse, sowie eine grobe Position, Größe und Form der Klasse möglich. Dabei wird in Kauf genommen, dass keine Aussage über die Menge von Objekten einer Klasse getroffen werden kann.
Diese Art der Annotation ist von Vorteil, wenn Objekte gruppiert erkannt werden sollen, und die Anzahl der Instanzen nicht von Relevanz ist.

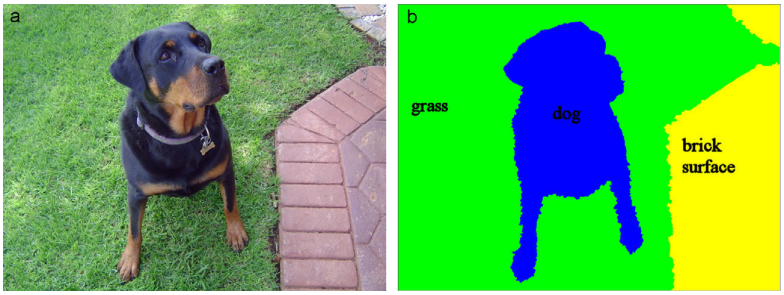


Abbildung 2.16: Semantic Segmentation mit drei unterschiedlichen Klassen

Instance Segmentation:

Beim letzten Annotationstypen werden, wie auch bei der Object Detection jeder Instanz einer Klasse, dessen Position sowie Klassenzugehörigkeit vorausgesagt. Somit kann eine Aussage getroffen werden, wie viele Instanzen einer Klasse auf dem Eingabebild vorhanden sind. Zusätzlich wird die Voraussage über Größe und Form der Instanz, durch die Zuordnung von Pixeln zu einer Klasse, wie auch bei der Semantic Segmentation, möglich. Instance Segmentation kann als eine Kombination aus Semantic Segmentation und Object Detection gesehen werden.

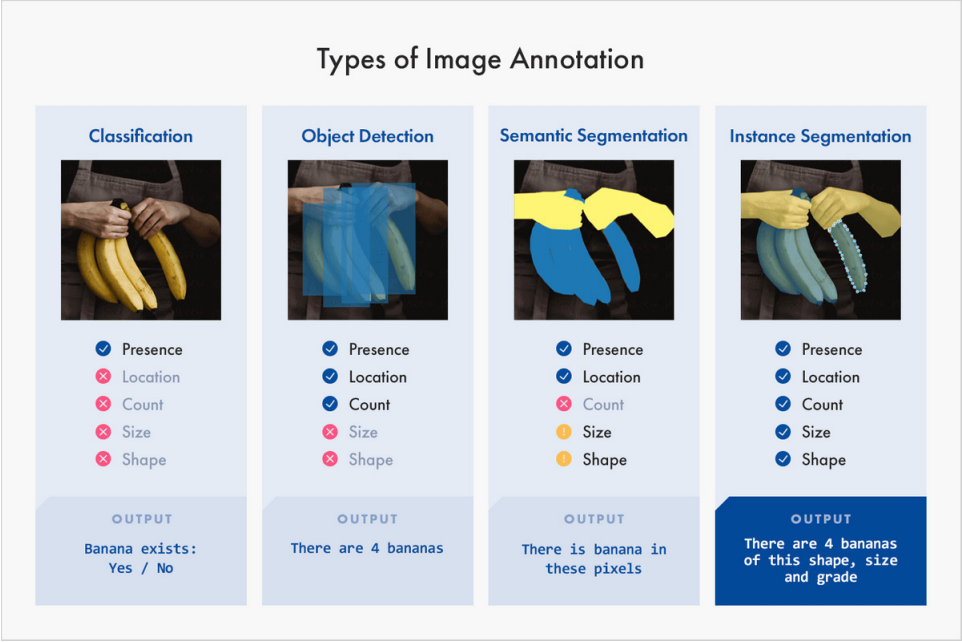


Abbildung 2.17: Annotationstypen im Vergleich

2.2.2 Bildannotationstechniken

Bounding-Box:
Wie der Name schon indiziert, werden bei Bounding-Boxen Objekte mittels Rechtecken gelabelt. Genutzt wird die Bounding-Box in Fällen, in denen die Form der Objekte nicht von Relevanz ist, sowie wenn Okklusion kein kritischer Faktor ist. Das dreidimensionale Gegenstück zu Bounding-Boxen sind *Cuboids*.

Landmarking:
Beim Landmarking werden Punkte in Objekte gesetzt, um die relative Anordnung von Bereichen innerhalb eines Objekts auszudrücken. Verwendet wird dies beispielsweise bei der Erkennung von Gesichtsausdrücken oder bei der Analyse von Bewegungsabläufen des Körpers im Sport.

Masking:
Masking wird überall da verwendet, wo Annotation auf Pixelebene nötig ist, wie bei der Image und Semantic Segmentation.

Polygon:
Bei der Annotation mit Polygonen werden die Ränder eines Objekts markiert. Genutzt werden Polygone wenn Objekte eine unregelmäßige Form aufweisen und Bounding-Boxen die Instanzen somit zu ungenau repräsentieren.

Polyline:
Für den Bereich des autonomen Fahrens spielt die Wahrnehmung der Umwelt eine sehr wichtige Rolle [TLL21]. Die Erkennung von Fahrbahnmarkierungen ist ein bedeutsamer Teil dieser Wahrnehmung. Für die Annotation von Trainingsbildern, welche für Machine-Learning Modelle genutzt werden, die Fahrbahnmarkierungen erkennen sollen, werden Polylines genutzt. Polylines sind über miteinander verbundene Punkte definiert, die allerdings im Gegensatz zum Polygonen keinen Zyklus bilden.

Es gibt noch eine Reihe anderer Annotationstechniken, welche für spezielle Zwecke genutzt werden. Im Rahmen dieser Arbeit werden aufgrund der Thematik allerdings nur die Bounding-Boxen sowie Polygone verwendet.

2.3 Testmetriken

Für die Beurteilung der Modelle wird auf unterschiedliche Metriken zurückgegriffen. Im Folgenden werden die genutzen Metriken genannt und erläutert.

2.3.1 Loss

Der `loss` ist ein Maß, welches die Fehlerhaftigkeit der Vorhersagen eines Modells beschreibt. Im Laufe des Trainings wird ein möglichst niedriger `loss`-Wert angestrebt.

Mask R-CNN protokolliert automatisch im Trainingprozess den `loss`, der sich nach jeder Epoche ergibt. Diese können nach Beendigung eines Trainings mittels *tensorboard*[1] betrachtet werden.
Der allgemeine `loss` setzt sich aus fünf unterschiedlichen `loss`-Werten zusammen. Zum einen aus den zwei `loss`-Werten des Region Proposal Networks, dem `rpn_bbox_loss` und dem `rpn_class_loss`, und zum anderen aus `mrcnn_class_loss`, `mrcnn_bbox_loss` und `mrcnn_mask_loss`[HGDG17].

2.3.2 Precision, Recall & Mean Average Precision

Für die Bewertung der Performanz von Klassifizierungsmodellen gibt es eine Vielzahl an Metriken die verwendet werden können. Im Rahmen dieser Arbeit wird die *Mean Average Precision* verwendet. Diese berücksichtigt neben den vorausgesagten Klassen auch die Bounding-Box und die Maskierung. Die Mean Average Precision wird oft im Rahmen der Messung der Performanz von Objekterkennungsmodellen genutzt.

Um die Mean Average Precision zu berechnen, müssen zunächst einige Begriffe für die Analyse einer binären Klassifikation definiert werden. Bei einer binären Klassifikation kann eine Klasse als die positive und die andere als die negative Klasse bezeichnet werden. Hat man beispielsweise ein Modell, welches Hunde und Katzen klassifizieren soll, wäre bei der Fragestellung, ob es sich auf einem Bild um einen Hund handelt, die Klasse Hund die positive und die Klasse Katze die negative. Die folgenden vier Kennzahlen [PNdS20] bezüglich der Vorhersage eines Modells sind für die weitere Analyse der Performanz von großer Bedeutung:

1. **True Positives:** Das Modell hat das Objekt als positive Klasse klassifiziert und es handelt sich wirklich um ein Objekt der positiven Klasse.

2. **True Negatives:** Objekte der negativen Klasse, die auch als negative klassifiziert wurden fallen unter True Negatives.

3. **False Positives:** Objekte der negativen Klasse, die als positiv klassifiziert wurden fallen unter False Positives.

4. **False Negatives:** Objekte der positiven Klasse, die als negativ klassifiziert wurden fallen unter False Negatives.

[1] Tensorboard, https://github.com/tensorflow/tensorboard

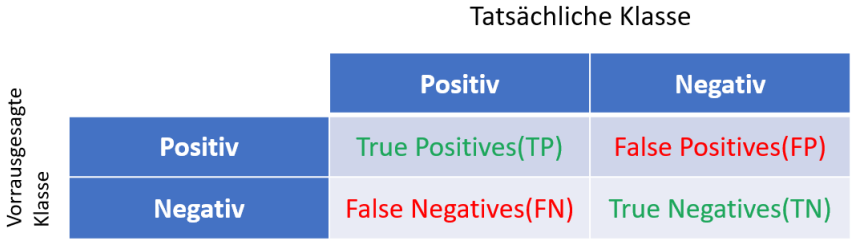


Abbildung 2.18: Klassifizierungskennzahlen

Die Metriken *Precision* und *Recall* werden für binäre Klassifikationen genutzt um die Güte des Modells zu messen. Um die Mean Average Precision für diese Arbeit nutzen zu können, muss das vorhandene Multiklassenproblem in mehrere binäre Klassifikationsprobleme unterteilt werden. Dies bedeutet, dass für jede Klasse ein Precision- und ein Recall-Wert berechnet wird. Dafür wird der *OVR* (One versus Rest) Ansatz genutzt. Jede Klasse nimmt dabei einmal die Rolle der positiven Klasse ein und alle anderen Klassen die Rolle der negativen Klasse.

Die Precision sagt aus, wie viele der als positiv klassifizierten Daten tatsächlich die positive Klasse repräsentieren, also wie gut das Modell eine bestimmte Klasse vorhersagen kann. Der Wert kann interpretiert werden, als die Wahrscheinlichkeit richtig zu liegen, wenn eine Klasse vorhergesagt wird. Sie drückt sich aus als Verhältnis zwischen True Positives und der Summe von True Positives und False Positives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

Die zweite Metrik ist der Recall. Er beschreibt wie viele der tatsächlich positiven Klasseninstanzen korrekt klassifiziert wurden. Der Wert gibt an, wie viel Prozent einer Klasse erkannt wurden. Für die Berechnung wird das Verhältnis zwischen den True Positives und der Summe der True Positives und False Negatives gebildet:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$

Die beiden Metriken können in einer $P \times R$ Kurve dargestellt werden, welche den Trade-Off zwischen beiden Metriken aufzeigt. Im Optimalfall sollte ein Modell möglichst alle Objekte richtig klassifizieren, was bedeutet, dass die Menge an FPs und FNs möglichst null sein soll. Ein möglichst hoher Flächeninhalt unter der $P \times R$ Kurve ist erstrebenswert. Die Fläche unter der Kurve wird als AP (*Average Precision*) bezeichnet. Sie wird nach [PNdS20] üblicherweise mittels *All-Point-Interpolation* oder *11-Point-Interpolation* berechnet, bei denen der Durchschnitt der maximalen Precision-Werte an den jeweiligen Recall-Leveln bestimmt

wird.

Für die *11-Point-Interpolation* gilt:

$$\mathrm{AP}_{11} = \frac{1}{11} \sum_{R \in \{0,0.1,\ldots,0.9,1\}} P_{\mathrm{interp}}(R) \tag{2.9}$$

mit

$$P_{\mathrm{interp}}(R) = \max_{\tilde{R}:\tilde{R} \geq R} P(\tilde{R}) \tag{2.10}$$

Für die *All-Point-Interpolation* gilt:

$$\mathrm{AP}_{\mathrm{all}} = \sum_{n} (R_{n+1} - R_n) P_{\mathrm{interp}}(R_{n+1}) \tag{2.11}$$

mit

$$P_{\mathrm{interp}}(R_{n+1}) = \max_{\tilde{R}:\tilde{R} \geq R_{n+1}} P(\tilde{R}) \tag{2.12}$$

Um nun die Mean Average Precision eines Modells zu bestimmen, wird der Durchschnitt der Average Precision über alle Klassen berechnet:

$$\mathrm{mAP} = \frac{1}{N} \sum_{N}^{i=1} \mathrm{AP}_i \tag{2.13}$$

Für die Berechnung der Mean Average Precision bedarf es der Einführung einer weiteren Metrik, der *Intersection over Union* (auch IoU). Die IoU ist eine Größe, welche die Überlappung zwischen vorausgesagter Bounding-Box und tatsächlicher Bounding-Box beschreibt. Je höher der Wert ist, desto genauer war die Voraussage.

Der Recall und die Precision sind abhängig von einem IoU-Threshold, welcher ein Grenzwert für die Klassifikation ist. Je nach Festlegung des Grenzwertes wird eine Voraussage als True Positive oder False Positive klassifiziert. Der Einfluss des IoU-Thresholds hat somit auch einen direkten Einfluss auf die Werte für Precision und Recall. Der Wert für die mAP kann je nach IoU-Threshold unterschiedlich ausfallen. mAP@0.5 beschreibt die Mean Average Precision bei Verwendung eines IoU-Thresholds von 0,5. Eine weitere Variante des mAP ist die Bildung eines Durchschnittswerts. Der Wert für mAP@[.5:.95] wird berechnet, indem die Mean Average Precision für unterschiedliche IoU-Thresholds bestimmt wird und dazu dann das arithmetische Mittel gebildet wird. Es werden Grenzwerte in einem Intervall von 0,5 bis 0,95 mit einer Schrittweite von 0,05 genutzt.

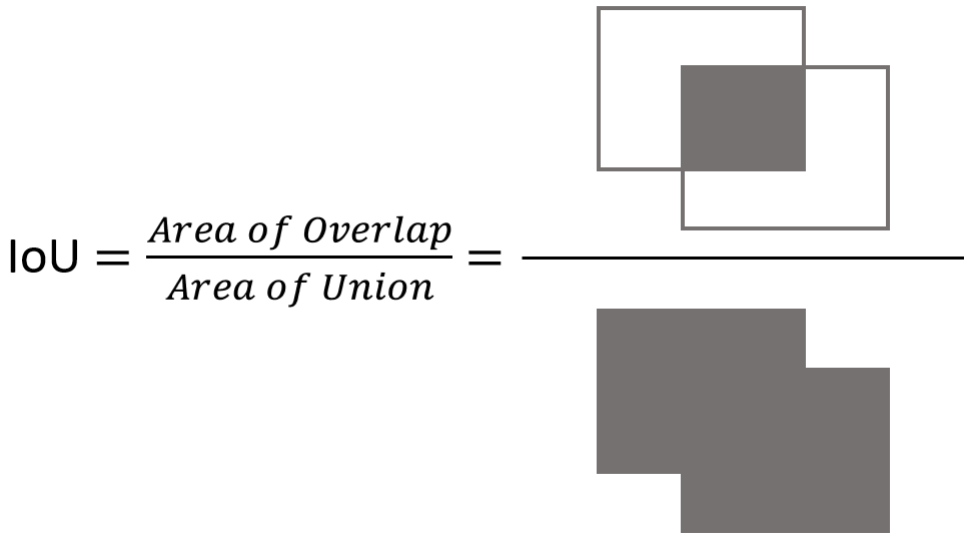


Abbildung 2.19: Intersection over Union

In dieser Arbeit wird für die Bestimmung der Mean Average Precision, ein IoU-Threshold von 0,5 gewählt.

Eine weitere Metrik die im späteren Verlauf dieser Arbeit für die Bewertung der Modelle zur Hilfe gezogen wird, ist der F1-Score. Der F1-Score kombiniert ebenfalls die zwei bereits erläuterten Metriken Recall und Precision. Er bildet das harmonische Mittel beider Werte.

Berechnet wird der F1-Score wie folgt:

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.14}$$

Um einen hohen F1-Score zu erreichen müssen sowohl Recall als auch Precision einen hohen Wert erreichen. Der F1-Score reagiert sensitiver wenn einer der beiden Werte niedrig ist.

3 Konzept und Implementierung

Der Ablauf der Arbeit wird im Folgenden Schaubild dargestellt.

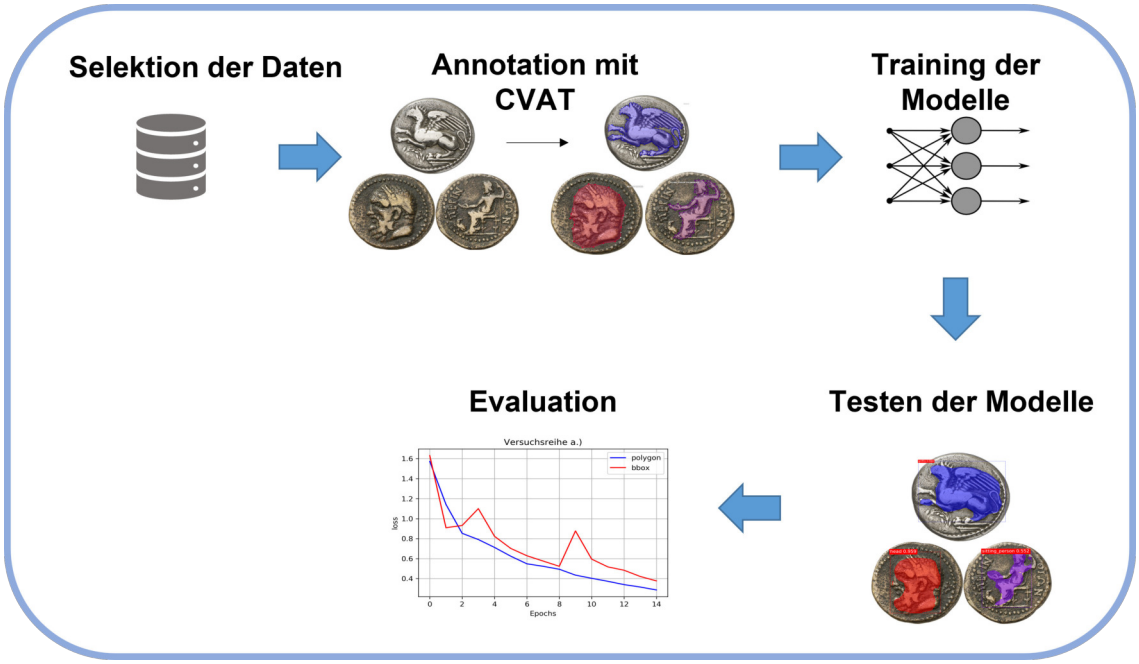


Abbildung 3.1: Ablaufdiagramm dieser Masterthesis

3.1 Vorbetrachtung

Im Folgenden Kapitel werden unterschiedliche Software und Tools aufgelistet, welche für die Umsetzung der Klassifikation benötigt werden.

In dieser Arbeit wird Mask R-CNN zur Klassifikation verwendet. Da das Modell in die Kategorie Supervised Learning fällt, werden annotierte Trainingsdaten benötigt. Im Rahmen dieser Arbeit waren der Großteil der verwendeten Daten ohne Annotationen. Da diese essentiell für den Trainingsprozess sind, bedarf es einer Anwendung, mit welcher Annotationen für die Trainingsdaten erstellt werden können.

Es wurden 25 unterschiedliche Annotationstools anhand folgender Kriterien verglichen: Aktivitätsstatus, letztes Update, Videounterstützung, Annotationsarten, Automatisierungsgrad und Exportformate. Eine Übersicht aller Tools befindet sich im Anhang.

Letztendlich wurde *CVAT* als geeigneter Kandidat gewählt.

3.2 Verwendete Hardware

In dieser Arbeit wird ein mit *Windows 10* operierender Rechner verwendet. Er besitzt folgende Spezifikationen:

1. CPU: Intel®Core™i5-6600 CPU @ 3.30Ghz 4-Kern Prozessor
2. GPU: NVIDIA GeForce GTX 970 (4GB RAM)
3. RAM: 8 GB

3.3 Verwendete Software/Tools

3.3.1 NVIDIA Treiber

Für das Training eines künstlichen neuronalen Netzes werden viele Matrix-Operationen durchgeführt, welche parallelisierbar sind [FSH04]. Um die Parallelisierbarkeit des Trainingsprozesses nutzen zu können und die Berechnungen auf der Grafikkarte laufen zu lassen, bedarf es der Installation von *NVIDIA CUDA* und *NVIDIA cuDNN*, da der verwendete Rechner eine NVIDIA-Grafikkarte besitzt. Die Versionen müssen zum einen in Abhängigkeit mit der genutzten Tensorflow Version und zum anderen in Abhängigkeit der eingesetzten Grafikkarte gewählt werden.

Für diese Arbeit wurde *NVIDIA CUDA* in der Version 10.0.130 verwendet, und *NVIDIA cuDNN* in der Version 7.4.1.5.

3.3.2 CVAT (Computer Vision Annotation Tool)

CVAT ist das in dieser Arbeit verwendete Tool für die Annotation der Münzbilder. Es wird von Intel angeboten und ist sowohl als online, sowie als offline Version erhältlich. Die Verwendung ist kostenlos und die Anwendung ist Open Source. Neben der Annotation von Bildern ist ebenfalls die Annotation von Videomaterial möglich. CVAT unterstützt die Annotation für verschiedenste Supervised Machine Learning Anwendungen wie zum Beispiel Object Detection, Image Classification und Image Segmentation.

Die Anwendung verfügt über eine Reihe von unterstützenden Funktionen, wie zum Beispiel eine semi-automatische Annotation mittels Deep-Learning Modellen. Es werden diverse Exportformate unterstützt.

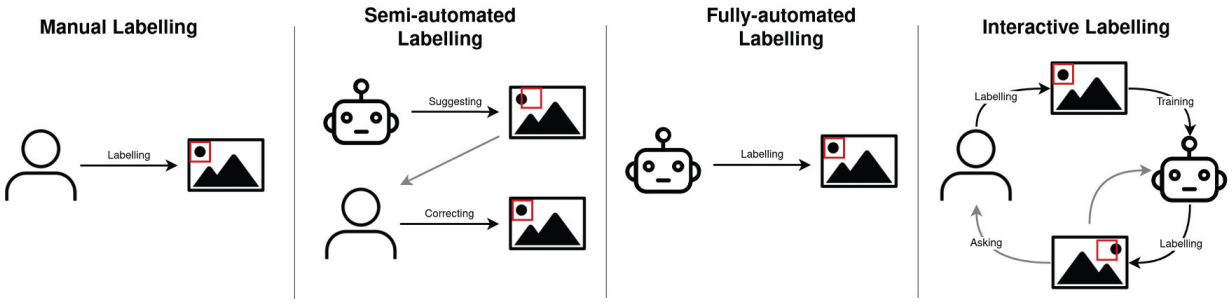


Abbildung 3.2: Verschiedene Automatisierungsgrade der Annotation

3.3.3 Docker

Docker ist eine Anwendung zur Containerisierung von Anwendungen, welche für diese Arbeit genutzt wurde, um das erstellte Machine Learning Modell zur Verfügung zu stellen.

Docker bietet den Vorteil, dass Software auf verschiedenen Host-Systemen laufen kann, ohne dass viele Anpassungen vorgenommen werden müssen. Zunächst wird ein Docker-Image erstellt, welches die Anwendung oder Software beinhaltet, die ausgeführt werden soll. Beim ausführen der Docker-Images werden sogenannte Docker-Container erstellt, in denen die Anwendung dann ausgeführt wird. Durch die Container soll sichergestellt werden, dass die Anwendung auf jedem Rechner oder Server uniform läuft und das gleiche Ergebnis liefert.

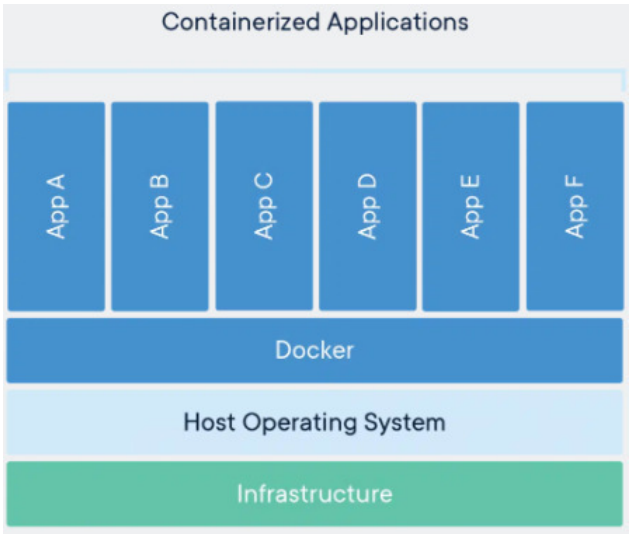


Abbildung 3.3: Mehrere Docker-Container auf einem Host-System

Anzumerken ist, dass der Docker-Container zum Zeitpunkt dieser Arbeit nicht auf Rechnern läuft die Apples M1 Chip nutzen. Offen ist, ob diese technische Einschränkung in Zukunft behoben wird.

3.3.4 Tensorflow

Tensorflow ist ein Machine Learning Framework welches von Google entwickelt wird [MAP+15]. Es steht unter der *Apache-2.0-Open-Source-Lizenz*.

Namensgebend für das Framework sind die sogenannten Tensoren. Tensoren sind multidimensionale Arrays.

In dieser Arbeit wird die Version 1.14.0 verwendet. Diese gibt es sowohl als CPU, als auch als GPU Variante. Bei entsprechend kompatibler Grafikkarte kann durch Verwendung der GPU-Version die Berechnungszeit deutlich verkürzt werden. An dem in dieser Arbeit verwendeten Rechner konnte ein Geschwindigkeitsunterschied vom Faktor zwölf erzielt werden.

3.3.5 Keras

Keras [Co15] ist eine in Python geschriebene Open-Source Bibliothek für Deep-Learning. Keras bietet Schnittstellen zu unterschiedlichen Machine Learning Frameworks, darunter *Tensorflow*, *Microsoft Cognitive Toolkit* und *Theano*. Seit der Tensorflow Version 1.4, ist es auch Bestandteil des Tensorflow Frameworks.

Keras besitzt ein höheres Level an Abstraktion, was eine einfachere Entwicklung von Machine Learning Anwendungen ermöglichen soll. Machine Learning Modelle können einfach durch Funktionen und Angaben unterschiedlicher Parameter des gewünschten Netzwerks generiert werden.

3.3.6 Mask R-CNN

Im Rahmen dieser Arbeit wird eine Mask R-CNN Implementierung von [Abd17] verwendet. Diese nutzt Keras und TensorFlow und baut auf einem ResNet101 [HZRS15] und einem FPN [LDG+16] auf.

Die Wahl fiel auf Mask R-CNN als Framework, da es eines der wenigen Computer Vision Frameworks ist, das mit Trainingsdaten arbeitet, welche wahlweise mit Bounding-Boxen oder Polygonen annotiert wurden. Außerdem bietet Mask R-CNN neben der Objekterkennung zusätzlich eine Instanzsegmentierung an.

3.4 Implementierung

Im Folgenden wird darauf eingegangen mit welchen Daten gearbeitet wurde, wie diese Daten annotiert wurden, sowie mit welchen unterschiedlichen Konfigurationen das Machine Learning Modell trainiert wurde. Alle Modelle werden auf vortrainierten Gewichten des *MS COCO* Datensatzes [LMB+14] trainiert.

Um ein Proof of Concept zu erstellen, welches eine Eignung vom Mask R-CNN Framework für die Motiverkennung auf antiken Münzen prüft, wurden in einem iterativen Prozess, unterschiedliche Modelle erstellt, die eine Eignung sukzessiv testen.

Im Folgenden werden die fünf wesentlichen Iterationen benannt, um auch in den späteren Kapiteln die Modelle indizieren zu können. Im anschließenden Kapitel wird dann genauer auf den verwendeten Datensatz eingegangen.

a.) Um einen ersten Eindruck der Eignung des Frameworks zur Erkennung von Motiven auf antiken Münzen zu bekommen, wurden aus den Datenbanken Motive rausgefiltert auf denen Tiere zu sehen sind. Die andere Kategorie *Objekte* wurde zunächst nicht betrachtet, da diese meist nur einen kleineren Teil eines Gesamtmotivs ausmachten, weshalb die Schwierigkeit der Erkennung des Objekts als höher eingestuft wird. Es wurden drei Tiere gewählt, die aus menschlicher und subjektiver Sicht unterschiedlich schwer zu Klassifizieren sind.

   Des Weiteren wurden zwei Modelle mit denselben Trainingsbildern trainiert, welche allerdings in einem Fall polygonal und im anderen Fall mit Bounding-Boxen annotiert wurden. Es sollte dadurch die Performanz beider Annotationsarten direkt evaluiert werden.

b.) In der nächsten Iteration wurde die Komplexität erhöht und es wurde geprüft, ob das Modell in der Lage ist, unterschiedliche Haltungen einer Person zu erkennen. Die Haltung einer Person wurde in drei Kategorien unterteilt: `stehend`, `sitzend` und `liegend`. Die Kategorie `liegend` konnte mangels Daten nicht berücksichtigt werden, weshalb das Modell nur auf die ersten zwei Kategorien trainiert wurde.

c.) Die nächste Iteration diente dazu sich ein Bild darüber zu machen, ob das Modell in der Lage ist Portraits zu unterscheiden, welche nach links, nach rechts oder nach vorne schauen.

d.) Ab der vierten Iteration lag der Fokus darauf, Daten zu betrachten die von einer Münzstätte stammen. Es wurden Klassen gewählt, von denen möglichst viele Bilder im Datensatz enthalten waren. Ziel war es möglichst viele Klassen erkennen zu können.
   Im zweiten Schritt wurde für jede Klasse die Menge an Trainingsbildern beschränkt. Es wurde daraufhin die Performanz mit dem Modell verglichen, bei dem alle vorhandenen Bilder verwendet wurden und keinerlei Beschränkung des Trainingsdatensatzes vorlag.

e.) In der letzten Iteration wurden für den Trainingsprozess Bilder einer weiteren Münzstätte verwendet, die ebenfalls viele Trainingsdaten enthielt. Die Menge an zu erkennenden Klassen wurde in dem Modell weiter erhöht.
   Es wurde zudem geprüft, ob Synergien zu Motiven anderer Münzstätten existieren, also

ob Motive anderer Münzstätten auch erkannt werden, obgleich das Modell nicht mit diesen trainiert wurde.

3.4.1 Erhebung und Wahl der Daten

Die Daten mit denen in dieser Arbeit gearbeitet wird, stammen aus diversen Quellen. Zum einen wurden unterschiedliche Münzen aus dem Webportal *Corpus Nummorum*[1] verwendet, welche mittels eines Skripts in Kombination mit einer RDF-Datenbank und einer SQL-Datenbank bezogen wurden, welche von Dr. Karsten Tolle zur Verfügung gestellt wurden. Des Weiteren wurden Münzbilder von Sebastian Gampe zur Verfügung gestellt. Diese enthielten zum einen Münzen aus unterschiedlichen Münzstätten und zum anderen Münzen mit Portraits und ihre dazugehörigen Annotationen im XML-Format.
Mittels eines Python-Skripts wurden aus den Münz-IDs in der Datenbank URIs generiert mithilfe derer die Münzbilder heruntergeladen wurden.

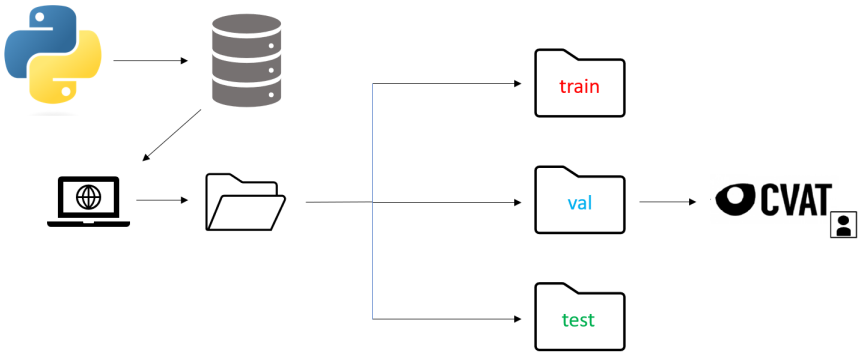


Abbildung 3.4: Erhebung der Daten

Im Folgenden werden die Begriffe *Münzdesign* und *Motiv* im Rahmen dieser Arbeit genauer definiert.

Als Münzdesign wird die Prägung einer Seite einer Münze definiert (die komplette visuelle Sicht einer Münzseite).

Als Motiv wird ein Bestandteil eines Münzdesigns definiert. Wenn beispielsweise auf einer Münze zwei unterschiedliche Tiere geprägt sind, besteht das Münzdesign aus zwei unterschiedlichen Motiven des jeweiligen Tieres.

Die Daten, welche für die vorher erwähnten Iterationen verwendet wurden, sahen wie folgt aus:

[1] Corpus Nummorum, https://www.corpus-nummorum.eu/, besucht am 05.11.2022

a.) Nachstehend sind die aus der Datenbank enthaltenen Motive aufgelistet:

Tier	**Anzahl**
Horse	797
Eagle	775
Serpent	751
Lion	652
Griffin	490
Bull	314
Tunny	308
Ram	258
Goat	255
Dolphin	248

Tabelle 3.1: Die zehn häufigsten Tiere, die im Webportal *Corpus Nummorum* aufgelistet sind.

Auch wenn die Menge an Trainingsdaten einen wichtigen Faktor für die Performanz eines Machine Learning Modells darstellt, wurden die Klassen, welche für das Training des ersten Modells gewählt wurden, auch anhand des Schwierigkeitsgrads ihrer Identifikation aus menschlicher Sicht ausgewählt.
Es wurden hierfür drei unterschiedliche Schwierigkeitsklassen definiert, welche durch jeweils eine Klasse an Motiven aus dem Datensatz repräsentiert wurden.

1. **Einfach identifizierbare Tiere:** Unter diese Kategorie fallen Tiermotive, deren Darstellung sich über alle Münzen hinweg stark ähneln. Es gibt wenig bis keine Varianten der Motive. Ihre Größe sowie Position auf der Münze unterscheidet sich kaum.
   Das Motiv des ***Greifs*** wurde für diese Kategorie gewählt.
2. **Schwer identifizierbare Tiere:** Unter diesen Kategorien finden sich Motive, deren Darstellung sich auf unterschiedlichen Münzen teilweise stark unterschieden. Des Weiteren treten die Motive teilweise mit anderen Motiven zusammen auf einem Münzdesign auf und überlappen sich zum Teil.
   Das Motiv der ***Schlange*** wurde für diese Kategorie gewählt.
3. **Tiere mit teils schwer und teils einfach identifizierbaren Motiven:** Die letzte Kategorie ist eine Kombination aus beiden zuvor genannten Kategorien. Es existieren Motive die sich stark ähneln und ohne andere Motive auf einer Münze auftreten, aber auch Motive die in anderen Motiven stecken oder auch in mehreren Varianten vorhanden sind.
   Die letzte Kategorie wird durch das Motiv des ***Löwen*** repräsentiert.

Die Datenbank enthält sowohl Münzen aus Metall als auch Gipsabdrücke unterschied-

licher Münzen. Für das Training der Modelle wurden sowohl Gipsabdrücke als auch Münzen aus Metall berücksichtigt.
Zunächst wurden pro Kategorie 100 Bilder ausgewählt, die jeweils im Verhältnis 6:2:2 in Trainings-, Validierungs-, und Testdatenset eingeteilt wurden.

b.) Im zweiten Experiment wurde getestet, ob das Modell in der Lage ist eine sitzende Person von einer stehenden zu unterscheiden.

	Training	**Validation**	**Test**
Standing	104	51	45
Sitting	60	30	34

Tabelle 3.2: Datensatz der stehenden und sitzenden Person

c.) Für das nächste Experiment wurden Bilder von Münzen mit Portraits samt Annotation von Sebastian Gampe zur Verfügung gestellt. Die Annotationen lagen im XML-Format vor. Dabei hatte jedes Bild eine dedizierte Annotationsdatei. Für das Mask R-CNN Framework müssen alle Annotationen in einer JSON-Datei gesammelt werden.

Des Weiteren musste die Angabe der Bounding-Box konvertiert werden. Im bereitgestellten Format ist die Bounding-Box über zwei diagonal zueinander liegende Punkte der Bounding-Box definiert. Im *COCO*-Format wird die Bounding-Box über die Koordinate der linken oberen Ecke der Bounding-Box, sowie über ihre Höhe und Breite definiert. Für die Konvertierung wurde ein Python-Skript erstellt, welches diese Aufgabe automatisiert. Es musste allerdings noch zusätzlich die Blickrichtung des Portraits manuell angegeben werden.

	Training	**Validation**	**Test**
Links	171	56	56
Rechts	19	6	6
Frontal	4	2	0

Tabelle 3.3: Datensatz mit drei unterschiedlichen Portraitarten

d.) Für das vierte Experiment wurden ebenfalls von Sebastian Gampe zur Verfügung gestellte Daten verwendet. Diese sind nach 120 unterschiedlichen Münzstätten aufgeteilt. Es wurde *Pergamon* gewählt, da diese Münzstätte die größte Anzahl an Münzbildern aufweist. Innerhalb des Datensatzes wurden dann wiederum diejenigen Klassen ausgewählt, welche innerhalb des Datensatzes ebenfalls die meisten Bilder vorweisen konnten. Die Klassen sind in dem Fall: `Head`, `Owl`, `Serpent`, `Serpent-Box` und `Sitting Person`. Es wurden mit den Klassen zwei unterschiedliche Modelle trainiert. Eines welches mit

allen Bildern aller betrachteten Klassen trainiert wurde und eines welches mit jeweils 30 Bildern pro Klasse trainiert wurde. Anzumerken ist, dass die Bilder dieses Datensatzes immer zwei Bilder einer Münze enthalten (Vorder- und Rückseite). Aufgrund dessen war es auch nicht vermeidbar, dass die Klasse `Head` mehr Klasseninstanzen im Trainingsdatensatz enthielt als die anderen Klassen. Auf den Münzen mit dem Motiv `Owl` und `Sitting Person` auf der Rückseite war immer jeweils ein Motiv der Klasse `Head`.

	Training	**Validation**	**Test**
Head	300	100	100
Owl	90	30	30
Serpent	180	60	60
Serpent-Box	180	60	60
Sitting-Person	222	74	74

Tabelle 3.4: Datensatz aus der Münzstätte Pergamon

	Training	**Validation**	**Test**
Head	60	20	20
Owl	30	10	10
Serpent	30	10	10
Serpent-Box	30	10	10
Sitting-Person	30	10	10

Tabelle 3.5: Limitierter Datensatz

e.) Im letzten Experiment wurden Bilder einer weiteren Münzstätte verwendet. Es wurden Bilder der Münzstätte *Perinthos* gewählt. Trainiert wurden 13 Klassen mit jeweils 20 Bildern. Es wurden vier Klassen paarweise zusammengefasst.

Zum einen wurde die Klasse `Bull` mit der Klasse des mythologischen *Apis* unter der Klasse `Bull` zusammengefasst. *Apis* war ein heiliger Stier, weshalb die Motive folglich stark dem Motiv des *Stiers* ähneln. Sowohl der Kopf mit den beiden Hörnern als auch der Körper, welcher auf manchen Münzen abgebildet ist, sind Merkmale anhand derer die beiden Klassen kaum zu unterscheiden sind.
Die anderen zwei Klassen die zusammengefasst worden sind, sind `Quadriga` und `Biga`, welche zur Klasse `Quadriga` zusammengefasst wurden. Beides sind zweirädrige Streitwägen, welche in dem Fall der *Biga* von zwei und im Fall der *Quadriga* von vier nebeneinander gehenden Zugtieren gezogen werden. Die Unterscheidung beider Klassen anhand der Anzahl der Zugpferde gelang in ersten Experimenten nicht zuverlässig. Mit Hinblick auf das Ziel, die Unterscheidung nicht zu feingranular zu gestalten, wurden beide Klassen zu einer zusammengefasst.

Alle Daten wurden händisch sortiert und ausgewählt.

	Training	**Validation**	**Test**
Head	230	6	6
Apis/Bull	20	6	6
Quadriga/Biga	20	6	6
Double Horse	20	6	6
Club	20	6	6
Laurel Wreath	20	6	6
Podium	20	6	6
Price Crown	20	6	6
Ship	20	6	6
Sitting Person	20	6	6
Standing Person	20	6	6
Table	20	6	6
Pot	20	6	6

Tabelle 3.6: Datensatz aus der Münzstätte Perinthos

3.4.2 Annotation

Die Annotation der Münzen wurde mit CVAT durchgeführt. CVAT bietet eine Vielzahl an Exportformaten, unter anderem die gängigsten: *Pascal VOC*, *COCO*, *VGG* und *YOLO*. Für diese Arbeit wurde als Exportformat *COCO 1.0* gewählt. Das Exportformat nutzt das JSON-Dateiformat und ist wie folgt strukturiert:

```
{
        "categories": [
                {
                        "id": ,
                        "name": "",
                        "supercategory": ""
                },
        ],
        "images": [
                {
                        "id": ,
                        "width": ,
                        "height":,
                        "file_name": "",
                }
        ],
        "annotations": [
                {
                        "id": ,
                        "image_id": ,
                        "category_id": ,
                        "segmentation": [
                                [Koordinaten der Maskierungspunkte
                                ...
                                ]
                        ],
                        "bbox": [minimal Koordinate, Höhe, Breite der Bounding-Box
                        ],
                }
        ]
}
```

Dabei ist anzumerken, dass in dem JSON-File noch weitere Meta-Daten wie beispielsweise die Lizenz, der Urheber und sonstige Angaben zu finden sind. Zur Veranschaulichung wurden in dem obigen Schaubild nur die für die Annotation relevanten Daten aufgeführt.

Es wurden zwei Annotationsarten verwendet. Zum einen die Bounding-Box, die von vielen Bilderkennungs Frameworks wie zum Beispiel *Yolo* [RDGF15] verwendet wird. Zum anderen wurden die Bilder polygonal annotiert.

Aufgrund der im Vergleich zu Bounding-Boxen höheren Komplexität der Polygone, ist diese Annotationsart deutlich aufwendiger zu erstellen. Allerdings bilden die Polygone die tatsächlichen Motive auf den Münzen detailierter und formgetreuer ab. Bei vorhandener po-

lygonaler Annotation, bedarf es keines zusätzlichen manuellen Aufwands für die Bounding-Box Annotation, da sich das COCO-Exportformat die vier Bounding-Box Koordinaten aus den Maxima und Minima der Koordinaten des Polygons generiert. Ob sich dadurch eine bessere Performanz in Hinblick auf die korrekte Erkennung der Motive ergibt, wird in den folgenden Kapiteln evaluiert.

Die Annotation der gesammelten Bilder war ein sehr großer und zeitaufwändiger Teil dieser Arbeit. Es wurden insgesamt 2801 Bilder annotiert, was ca. 46 Stunden Arbeitsaufwand entspricht (bei optimistischer Einschätzung von einer Minute Bearbeitungszeit pro Motiv).

3.4.3 Mask R-CNN

Die Implementierung des Frameworks orientiert sich an den Arbeiten von [Sha20] und [Sin22]. Anpassungen bedarf es hinsichtlich der Anzahl, sowie Entität der in dieser Arbeit zu erkennenden Klassen. Des Weiteren bedarf es einer Anpassung des Einlesens der Inputdaten, welche dem Framework übergeben werden. Beide zuvor genannten Arbeiten nutzten das VIA (VGG Image Annotator) Format, wohingegen in dieser Arbeit das COCO-Format verwendet wird, was eine Anpassung des Codes nötig macht.

Die Modelle in dieser Arbeit basieren alle auf einem bereits vortrainierten Netz. Dieses wurde auf dem COCO-Datensatz vortrainiert. Jedes Modell was im Rahmen dieser Arbeit trainiert wird, nutzt somit das vortrainierte Netz, was Trainingsiterationen spart. Dies wird auch als *Transfer learning* [ZQD+21] bezeichnet. Bereits erlernte Muster können angewendet werden, ohne dass diese nochmals trainiert werden müssen.

Die Hyperparameter, welche für das Training der Modelle genutzt wurden, werden nachfolgend aufgelistet:

1. **Epochs:** Die Zahl der Trainingsdurchläufe wurde im Rahmen dieser Arbeit variiert und getestet. Letztlich wurde für das Training des letzten Modells eine Epochenzahl von 30 als geeignet bewertet.

2. **Batchsize:** Die Größe der Batches ist durch den Speicher der Grafikkarte limitiert. Da in dieser Arbeit der Fokus nicht auf dem Hyperparameterttuning liegt, wurde sich für das *Stochastic Gradient Descent*-Verfahren entschieden, bei welchem die Batchsize auf eins festgelegt ist.

3. **Lernverfahren:** Das Framework nutzt als Lernverfahren Stochastic Gradient Descent. Das herkömmliche *Gradient Descent*-Verfahren, welches im zweiten Kapitel erläutert wurde, verwendet alle Datenpunkte eines Trainingsdatensatzes, um die Gewichte des Netzes anzupassen. Dies ist bei großen Datensätzen sehr rechenaufwändig.

   Im Gegensatz dazu, wird bei Stochastic Gradient Descent in jeder Iteration stets ein Datenpunkt betrachtet und die Gewichte anhand des Datenpunktes angepasst.

Es wurde die vom Framework standardmäßig gesetzte Lernrate von 0,001 übernommen. Im Paper [HGDG17] wurde eine Lernrate von 0,02 verwendet, was aber in der Implementierung mit Tensorflow zu Problemen führte.

4. **Kostenfunktion:** Es werden unterschiedliche Kostenfunktionen in dem Framework verwendet. Für die Erkennung der Maske wird *Binary Crossentropy* verwendet, da die Erkennung der Maske ein binäres Klassifikationsproblem ist. Es muss entschieden werden, ob ein Pixel zur Maske gehört oder nicht.

   Für die Erkennung der Klassen wird *Sparse Categorial Crossentropy* verwendet. Der Unterschied zum herkömmlichen *Categorial Crossentropy* besteht lediglich in der Abbildung der Klassenlabel. Sparse Categorial Crossentropy bildet die Klassen auf Integer-Werte ab, während Categorial Crossentropy die *One-Hot-Kodierung* nutzt, um die Klassen abzubilden.

Es wurden in dieser Arbeit abseits der Hyperparameter, eine Reihe von Konfigurationen des Modells trainiert und dessen Ergebnisse evaluiert. Zunächst wurde die Performanz getestet in Hinblick auf polygonaler und Bounding-Box Annotation.
In der nächsten Versuchsreihe wurde die Performanz der Modelle anhand der Datenmengen verglichen, mit denen sie trainiert wurden.
In der letzten Versuchsreihe wurden dann mehr Klassen hinzugenommen, auf denen das Netzwerk trainiert wird. Statt auf drei Klassen wurde das Modell auf 13 Klassen trainiert.

4 Evaluation und Resultate

4.1 Testergebnisse

Im Folgenden werden die Ergebnisse der einzelnen Versuchsreihen aufgelistet. Es werden die jeweiligen `loss`-Werte der Modelle aufgezeigt, sowie deren Mean Average Precision.

a.) Für den Vergleich des Modells, welches mit Bounding-Boxen trainiert wurde und dem Modell, welches mit Polygonen trainiert wurde, wird zunächst der `loss` beider Modelle verglichen. Wie bereits in Kapitel 2.3.1 erwähnt, setzt sich der `loss` aus der Summe von fünf verschiedenen `loss`-Werten zusammen.

Da für die Annotation mittels Bounding-Boxen die Maskierung keine Rolle spielt, wird diese Metrik für den Vergleich der `loss`-Werte nicht berücksichtigt und vom Gesamt-`loss` subtrahiert.
Folgender Graph zeigt die `loss`-Werte beider Modelle im Verlauf von 15 Trainigsepochen auf:

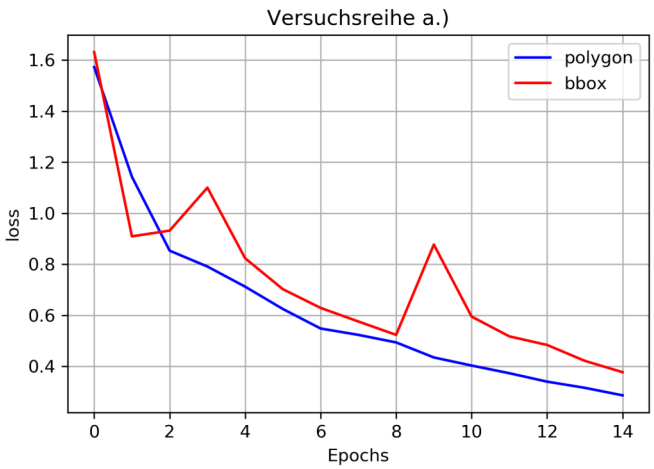


Abbildung 4.1: Loss per Epoch in Versuch a.)

Nach 15 Epochen weißt das Modell, welches mit Polygon-Annotation trainiert wurde einen `loss`-Wert von 0,29 auf, während das Modell, das mit Bounding-Box Annotation trainiert wurde einen `loss`-Wert von 0,38 aufweist. Im Verlauf der Epochen erkennt man, dass das Modell, welches die Polygon-Annotation verwendete, ab der zweiten Epoche stets bessere `loss`-Werte erzielt.

Auf dem Testset erreicht das Modell mit Bounding-Box Annotation ein mAP@0.5 von 0,44 und das Modell mit Polygon-Annotation einen mAP@0.5 Wert von 0,40.

	loss	**mAP@0.5**
Bounding-Box	0,38	0,40
Polygon	0,29	0,44

Tabelle 4.1: Loss und mAP Werte über 15 Epochen

Beide Metriken zeigen eine Indikation dahingehend, dass das Modell mit Polygon-Annotation etwas besser abschneidet.

Betrachtet man die Klassifizierungsergebnisse der einzelnen Klassen des Polygon-Modells genauer, haben sich die Erwartungen der Klassen hinsichtlich ihrer Komplexität bewahrheitet.

Polygon - Modell Versuch a)				
	Precision	Recall	F1-Score	Support
				25
Griffin	0,54	0,95	0,69	20
Serpent	0,50	0,18	0,27	22
Lion	0,39	0,41	0,40	22

Tabelle 4.2: Precision, Recall und F1-Score der einzelnen Klassen bei 60 Testbildern

Die Tabelle 4.2 zeigt, dass die Klasse `Griffin` in allen drei Metriken (Precision, Recall und F1-Score) besser abgeschnitten hat, als die anderen Klassen. Somit deckt sich das Ergebnis des Modells mit der subjektiven Einschätzung, dass der Greif aus menschlicher Sicht einfacher zu klassifizieren ist als die anderen beiden Klassen.

Der sehr hohe Recall-Score zeigt, dass das Modell sehr gut darin war alle Greifmotive zu erkennen. Der deutlich niedrigere Precision-Score deutet an, dass das Modell allerdings dazu tendiert viele Motive der Klasse `Griffin` zuzuordnen, die ein anderes Motiv enthalten.

Der sehr niedrige Recall-Score der Klasse `Serpent` zeigt deutlich, dass das Modell Schwierigkeiten hat das Motiv zu erkennen. Dies deckt sich auch mit der anfänglichen Einschätzung.

Abb 4.2 macht deutlich, wieso das Modell Probleme mit der Erkennung der Klasse `Serpent` haben könnte. Das Motiv in (a) auf Abb 4.2 wurde nicht erkannt, war aber auch deutlich komplexer als das Motiv in (b) welches klassifiziert werden konnte. Die Schlange auf Motiv (a) umwindet eine Person und wird teilweise von ihrer Hand bedeckt, während die Schlange in (b) als einziges Motiv deutlich erkennbar und größer auf der Münze abgebildet ist.

(a) CN Coin 14499 (b) CN Coin 5095

Abbildung 4.2: Im Gegensatz zur Münze (a), auf dem kein Motiv entdeckt wurde, wurde die Schlange auf Münze (b) korrekt klassifiziert, auch wenn die Maske das Motiv nicht optimal abdeckt.

Die Klasse `Lion` befindet sich von der Erkennungsgüte wie erwartet, zwischen den bereits genannten Klassen.

(a) CN Coin 7451 (b) CN Coin 39638

Abbildung 4.3: Zwei unterschiedliche Motive des Löwen

Abb 4.3 zeigt zwei verschiedene Darstellungsformen des Löwen, die jeweils erkannt wurden. Beide Motive sind allerdings im Vergleich zu den Schlangenmotiven auch weniger komplex. Beide Münzen enthalten nur ein Motiv ohne sich mit einem zweiten zu kreuzen.

b.) Der nächste Versuch sollte zeigen wie gut das Modell eine stehende von einer sitzenden Person unterscheiden kann. Da der vorherige Versuch bereits zeigte, dass die polygonale Annotation etwas besser abschneidet, wurde als Annotationsart der in dem Versuch genutzten Münzen ebenfalls die polygonale Annotation verwendet.

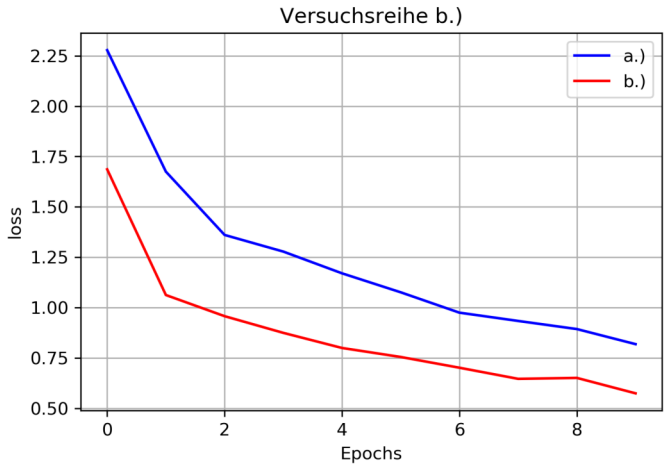


Abbildung 4.4: Vergleich der Modelle aus Versuchsreihe a.) und b.)

	loss	**mAP@0.5**
Versuch a	0,81	0,44
Versuch b	0,57	0,5

Tabelle 4.3: `loss` und mAP Werte der Modelle a.) und b.)

Der `loss`-Wert des Polygon-Modells stimmt nicht mit der Angabe der Tabelle in a.) überein, da zum einen der Mask-`loss` mit einberechnet wurde und zum anderen der `loss`-Wert beider Versuche bei der 10. Epoche betrachtet wurde, weshalb der Wert ebenfalls etwas höher liegt im Vergleich zum vorherigen Versuch.

Modell Versuch b.)				
	Precision	Recall	F1-Score	Support
Standing-Person	0,92	0,79	0,85	56
Sitting-Person	0,68	0,88	0,77	32

Tabelle 4.4: Precision, Recall und F1-Score der beiden Klassen `Standing Person` und `Sitting Person`.

Betrachtet man die Precision- und Recall-Werte, sieht man ein ausgeglicheneres Verhältnis als im vorherigen Versuch. Der F1-Score ist auch in beiden Fällen höher als bei den Klassen im vorherigen Modell.

Dies könnte aber auch mit zwei Tatsachen zusammenhängen. Erstens existiert in dieser Versuchsreihe eine Klasse weniger, welche das Modell erkennen muss. Dadurch reduziert sich die Komplexität des Klassifikationsproblems.

Zweitens unterscheiden sich die Motive auch markant an der Form und Position der Beine. Außerdem enthielt der Datensatz kaum Motive einer Klasse die sich signifikant

voneinander unterschieden haben, wie es beispielsweise bei der Klasse `Lion` der Fall war.

(a) CN Coin 5843 (b) CN Coin 9796

Abbildung 4.5: Beispiele der Klassen `Sitting-Person` und `Standing-Person`

Auffällig war in dieser Versuchsreihe, dass die Beine der Motive mit der sitzenden Person oft falsch maskiert wurden. Auf vielen Bildern sitzt die Person auf einem Stuhl, dessen Stuhlbeine man auf der Münze auch sehen kann. In manchen Bildern, wie auch in Abb 4.5 Bild (b), wurde eins der Stuhlbeine fälschlicherweise als Bein der Person maskiert.
Alles in Allem war sowohl die Klassifizierung als auch die Maskierung dennoch zufriedenstellend.

c.) Im nächsten Versuch wurde geprüft, ob es möglich ist ein Modell zu trainieren, welches gegebenenfalls die Blickrichtung von Portraits erkennen kann. Zwar gelang in den meisten Fällen eine Klassifizierung der Portraits, allerdings wurden alle Portraits stets als nach rechts blickend klassifiziert. Dies ist wahrscheinlich dem geschuldet, dass der Trainingsdatensatz zum einen sehr klein und zum anderen stark unbalanciert ist und deutlich mehr Portraits vorhanden sind, die nach rechts blicken. Aufgrund des schlechten Ergebnisses werden im Folgenden nur die Klassifizierungsergebnisse betrachtet. Auf eine genauere Betrachtung des `loss` und mAP wird verzichtet.

Modell Versuch b)				
	Precision	Recall	F1-Score	Score
Left	0,00	0,00	0,00	9
Right	0,89	1.00	0,94	85
Front	0,00	0,00	0,00	2

Tabelle 4.5: Precision, Recall und F1-Score der Klassen `Left`, `Right` und `Front`.

Der hohe Recall-Wert von 1,00 für die Klasse `Right` kommt dadurch zustande, dass das Modell stets die Klasse `Right` vorraussagt. Somit erkennt sie automatisch alle Instanzen dieser Klasse. Allerdings werden die anderen Klassen nicht klassifiziert, wodurch die Metrik nicht aussagekräftig ist.

d.) Versuch d.) besteht aus mehreren Modellen, deren Performanz direkt miteinander verglichen werden. Alle Modelle wurden mit Münzen aus der Münzstätte Pergamon trainiert und sind auf die Klassifikation von fünf unterschiedlichen Klassen ausgelegt.

Es wurde ein Modell in dem Versuch als Referenz gewählt. Dieses Modell wurde mit allen verfügbaren Bildern in 10 Epochen trainiert. Die Menge an Epochen wurde anhand dessen ausgewählt, dass der `loss` sich nach 10 Epochen nicht mehr sonderlich verbessert hat.

Das zuerst verglichene Modell wurde hinsichtlich der Trainingsdaten beschränkt, welche auf 30 Bilder pro Klasse limitiert wurde.

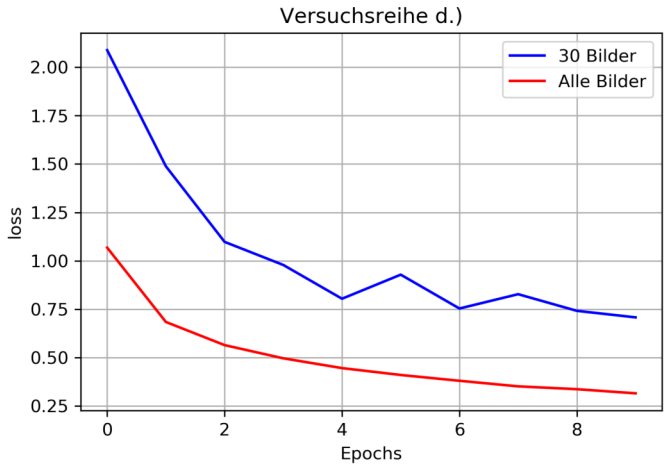


Abbildung 4.6: Loss per Epoch Versuch d.)

Nun wurden die Anzahl der Epochen sukzessiv erhöht bis der `loss` der Modelle sich anglich. Dies wurde nach 30 Epochen erreicht.

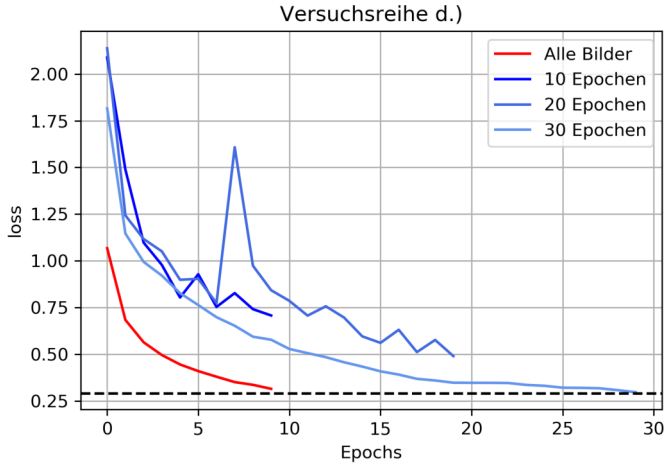


Abbildung 4.7: Loss zu unterschiedlichen Epochen

Epochen	10	20	30
loss-Modell d.) loss-Referenzmodell	0,57 0,31	0,34	0,29

Tabelle 4.6: `loss`-Werte an unterschiedlichen Epochen

Zwecks des Annotationsaufwands wurde ein weiteres Modell erstellt und verglichen. Bei dem Modell wurde die Menge der Trainingsdaten weiter auf 20 restriktiert, und in 30 Epochen trainiert. Da beide Modelle anhand der Metriken ähnliche Werte zeigten, wurde die Menge an Trainingsbildern und die Anzahl an Trainingsepochen für die nächste Versuchsreihe als Referenzwerte genommen.

Modell	loss	mAP@0,5
30 Bilder	0,29	0,68
20 Bilder	0,34	0,65

Tabelle 4.7: `loss`-Werte bei unterschiedlicher Anzahl an Trainingsbildern

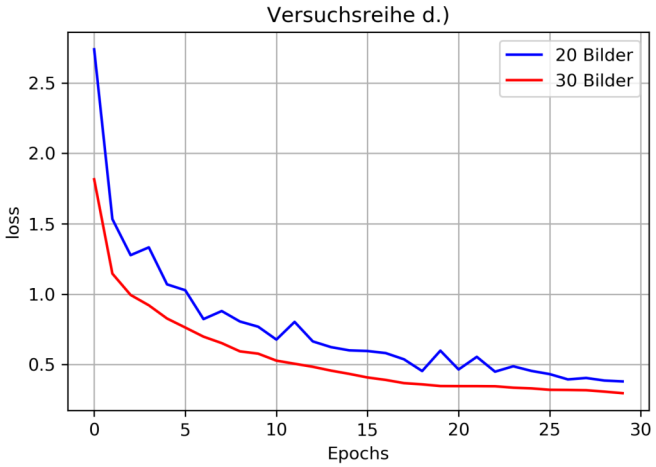


Abbildung 4.8: Loss bei 30 Epochen und unterschiedlicher Menge an Trainingsdaten

Da gezeigt wurde, dass das Modell auch mit weniger Trainingsbildern scheinbar eine hinreichende Performanz zeigt, wurden für das Testen des Modells vier weitere Klassen mit jeweils 20 Trainingsbildern in den Trainingsdatensatz aufgenommen. Es sollten zusätzlich die Klassen `Eagle`, `Podium`, `Bull` und `Bow` klassifiziert werden.

Modell Versuch d.) mit 9 Klassen				
	Precision	Recall	F1-Score	Score
Sitting-Person	1,00	0,90	0,95	10
Head	0,76	1,00	0,86	47
Serpentbox	1,00	1,00	1,00	10
Owl	1,00	0,90	0,95	10
Serpents	0,83	1,00	0,91	10
Eagle	1,00	0,88	0,93	8
Podium	0,91	0,50	0,65	20
Bull	0,78	0,37	0,50	19
Bow	0,73	0,73	0,73	11

Tabelle 4.8: Precision, Recall und F1-Score der 9 Klassen die aus dem Datensatz der Münzstätte Pergamon stammen.

Im Folgenden wird auf die wichtigsten Erkenntnisse aus dem Test des vierten Modells eingegangen. Wie auch in dem vorherigen Versuch, schneiden die Klassen `Sitting Person` und `Head` gut ab. Beide erhielten in dem Test einen F1-Score von über 80 Prozent. Positiv hervorzuheben ist, dass das Modell auch Motive auf Gipsabdrücken erkannt hat, von denen ein Teil abgebrochen ist.

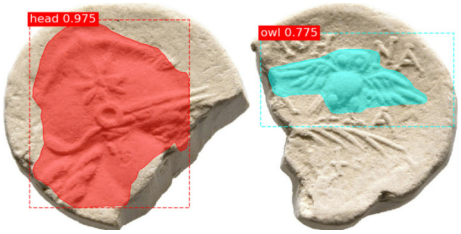


Abbildung 4.9: cnt-coin-39983-p-rez-CN12525

In Abb 4.9 ist solch ein beschädigter Gipsabdruck zu sehen. Beide Klassen wurden dennoch erkannt. Das Motiv der Klasse `Owl` ist zwar intakt, weshalb eine erfolgreiche Klassifizierung nicht verwundert, allerdings ist durch die Beschädigung des Gipsabdrucks ein signifikanter Teil des Gesichts der Klasse `Head` abgeschnitten. Trotzdem hat das Modell das Motiv mit einer Konfidenz von ca. 98% korrekt klassifiziert.

Auf die Klassen `Serpentbox`, `Serpents` und `Eagle` wird nicht im Detail eingegangen, da alle drei Klassen einen hohen F1-Score aufweisen, sowie sonst keine besonderen Auffälligkeiten festgestellt wurden.

Die Klasse `Podium` hat einen verhältnismäßig hohen Precision-Wert im Vergleich zu seinem Recall-Wert. Betrachtet man die klassifizierten Bilder, erhält man eine Indikation der Ursache, die zu der Diskrepanz der beiden Metriken und dem somit, im Vergleich zu den anderen Klassen, niedrigen F1-Score führt.

Die Betrachtung der Testbilder zeigt, dass das Modell Probleme hat einzelne Entitäten

Abbildung 4.10: Unterschiedliche Varianten der Klasse `Podium`

der Klasse `Podium` zu erkennen, sobald mehr als eine Entität auf einem Bild zu finden ist. Betrachtet man Abb 4.10 erkennt man, dass das einzelne Motiv rechts im Bild noch erkannt und größtenteils korrekt maskiert wird. Die mittige Münze mit zwei Entitäten wird als eine einzige Instanz klassifiziert. Die drei Instanzen auf dem Gipsabdruck links auf der Abbildung konnten allesamt nicht klassifiziert werden. Dieses Verhalten trat auch auf anderen Testbildern mit Motiven dieser Klasse auf.

Die Klasse `Bull` hat ebenfalls Münzen auf denen sich mehr als eine Instanz der Klasse befindet. Außerdem existieren unterschiedliche Variationen der Klasseninstanz. Sowohl vom Optischen als auch von der Blickrichtung unterscheiden sich einige Motive.

Abbildung 4.11: Unterschiedliche Varianten der Klasse `Bull`

Bei Betrachtung der Testbilder lässt sich allerdings erkennen, dass der im Klassenvergleich schlechtere F1-Score andere Ursachen hat als jene, die den schlechteren F1-Score bei der Klasse `Podium` verursacht haben.

Abb 4.11 veranschaulicht, dass im Gegensatz zu der Klasse `Podium`, in einigen Fällen der Klasse `Bull` auch mehrere Instanzen korrekt klassifiziert und auch maskiert wurden.

Die linke und mittlere Münze in Abb 4.11 zeigt, dass auch unterschiedliche Varianten erkannt werden können. Die beiden Motive sind nicht nur von der Blickrichtung gespiegelt, sondern sind von der Darstellungsform unterschiedlich, wie beispielsweise die Länge des Horns.

Der vergleichweise niedrige F1-Score, ergibt sich dadurch, dass weniger Motive jeder Variante erkannt werden. Es werden zum einen Motive nicht erkannt die einzeln auf einer Münze auftauchen und zum anderen wird teilweise nur eine Instanz klassifiziert,

obwohl sich zwei Instanzen auf der Münze befinden. Im Falle, dass die Gesamtmenge eines Motivs auf einem Eingabebild größer als eins war, wurde im Gegensatz zur Klasse `Podium`, zumindest eine Teilmenge aller abgebildeten Instanzen erkannt und klassifiziert und auch korrekt maskiert. In Abb 4.11 war bei der mittleren Münze zu sehen, dass eine Instanz klassifiziert wurde, die Maskierung allerdings über beide Instanzen abgebildet war.

Die letzte Klasse `Bow` liegt mit ihrem F1-Score von 0,73 im Vergleich zu den anderen Klassen im Mittelfeld. Betrachtet man die Testbilder, ist positiv hervorzuheben, dass das Motiv erkannt wurde, obwohl es in unterschiedlichen Rotationen vorliegt. Des Weiteren ist die Maskierung in den meisten Fällen auch sehr akkurat.

Abbildung 4.12: Unterschiedliche Varianten der Klasse `Bow`

e.) Im letzten Versuch wurde ein Modell erstellt, welches auf Münzen der Münzstätte Perinthos trainiert wurde. Das Modell wurde darauf trainiert 13 unterschiedliche Klassen zu erkennen. Es wurde mit Parametern trainiert, welche aus den Erkenntnissen der vorherigen Versuche hervorgegangen sind. Es wurden jeweils 20 Bilder für jede Klasse in den Trainingsdatensatz aufgenommen. Das Modell wurde über 30 Epochen trainiert.

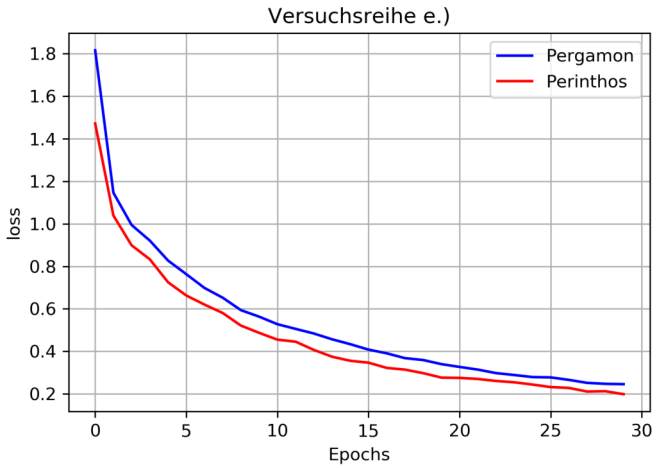


Abbildung 4.13: Loss per Epoch der beiden Modelle zu den Münzstätten Pergamon und Perinthos im Vergleich

Modell	**loss**	**mAP@0,5**
Pergamon	0,29	0,68
Perinthos	0,34	0,65

Tabelle 4.9: `loss` und mAP Werte der beiden Modelle zur Erkennung der Münzstätten Pergamon und Perinthos

Das Modell, welches auf Münzen von der Münzstätte Perinthos trainiert wurde, zeigt ähnliche Ergebnisse bezüglich des `loss`- und mAP-Wertes (Tabelle 4.9), obwohl das Modell auf 13 statt der vorherigen 9 Klassen trainiert wurde (≈ 50 % Steigerung der Anzahl an Klassen).

Perinthos			
	Precision	Recall	F1-Score
Sitting Person	0,64	0,64	0,64
Head	0,91	0,99	0,95
Podium	0,90	0,69	0,78
Bull	0,43	0,30	0,35
Quadriga	0,57	0,40	0,67
Double Horse	0,75	0,55	0,63
Club	1,00	0,10	0,18
Laurel Wreath	0,75	0,60	0,67
Price Crown	0,83	0,38	0,53
Ship	0,88	0,64	0,74
Table	1,00	0,78	0,88
Pot	0,50	0,40	0,44
Standing Person	0,88	0,70	0,78

Tabelle 4.10: Precision, Recall und F1-Score der 13 Klassen die aus dem Datensatz der Münzstätte Perinthos stammen

Im Folgenden werden die einzelnen Klassen genauer betrachtet. Es wird sich allerdings nur auf die Klassen fokussiert, welche Besonderheiten aufzeigten, oder signifikant schlechter erkannt wurden als der Durchschnitt.

Auffällig ist die drastische Verschlechterung der Klasse `Sitting Person`. Der F1-Score ist im Vergleich zum Pergamon-Modell von 0,95 auf 0,64 gefallen. Die Betrachtung der Trainings- und Test-Daten legt zwei Vermutungen nahe, die diese Verschlechterung bewirkt haben könnten: Zum einen befanden sich im Trainingsdatensatz des Versuchs e.) mehr Gipsabdrücke als im Trainingsdatensatz aus Versuch d.). Zusätzlich befanden sich viele Gipsabdrücke unter den Trainingsdaten, welche von der Qualität deutlich schlechter waren, als die Gipsabdrücke aus Versuch d.)(siehe Abb 4.14).

Abbildung 4.14: Zwei Gipsabdrücke mit der Klasse `Sitting Person` die deutliche Qualitätsunterschiede aufweisen

Der zweite Faktor der gegebenenfalls die Klassifikationsgüte beeinflusst haben könnte, sind die Hintergründe auf den Fotoaufnahmen. Unter den Trainingsdaten wurden ungefähr die Hälfte der Bilder auf einem grauen Hintergrund aufgenommen. Im Trainingsdatensatz aus Versuch d.) waren auch vereinzelt Münzen auf anderen Hintergründen fotografiert worden, allerdings waren das Ausnahmen ($< 10\%$).

Die nächste auffällige Klasse war `Bull`. Der F1-Score war auch hier schlechter als im Versuch d.). Auch hier liegt die Vermutung nahe, dass die Qualität der Trainingsdaten der Grund für die schlechte Performanz ist.

Abbildung 4.15: Gipsabdruck mit der Klasse `Bull`

Auf Abb 4.15 ist eine Münze mit dem Motiv `Bull` exemplarisch dargestellt. Das Motiv ist kaum zu erkennen. Der Großteil der Bilder dieser Klasse hatte eine ähnlich schlechte Qualität.

Die nächsten zwei Klassen, welche Auffälligkeiten zeigten, sind zum einen die Klasse `Quadriga` und zum anderen die Klasse `Double Horse`. Die beiden Klassen haben eine Gemeinsamkeit. Beide Klassen beinhalten als Motiv Pferde. Die Klasse `Double Horse` zeigt zwei miteinander verbundene Pferde. Die Klasse `Quadriga` beinhaltet zwei bis vier Pferde, die als Zugtiere genutzt werden. Im Zuge der Klassifikation von Münzen beider Klassen ist zu sehen, dass das Modell in manchen Fällen die beiden Klassen vertauscht. Es werden sowohl Motive der Klasse `Quadriga` als `Double Horse` klassifiziert, als auch umgekehrt Motive der Klasse `Double Horse` als `Quadriga`.

Abbildung 4.16: Motiv der Klasse `Quadriga` in der die Klasse `Double Horse` fälschlicherweise klassifiziert wurde

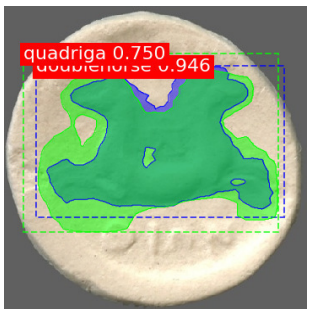


Abbildung 4.17: Das Modell findet keine eindeutige Klasse

Besonders auf der linken Münze in Abb 4.16 sieht man, dass die Person nicht maskiert wird, sondern Hauptsächlich die Pferde.

Die Klasse, bei der die Klassifikation am schlechtesten abschnitt, war die Klasse `Club`. Mit einem F1-Score von 0,18 liegt die Performanz deutlich unter den anderen Klassen.

Abbildung 4.18: Münze mit dem Motiv der Klasse `Club`

Das Modell hatte große Probleme diese Klasse zu erkennen. Vergleicht man das Motiv der Klasse mit den anderen Motiven, wird ersichtlich, dass das Motiv kaum markante Merkmale aufweist (Abb 4.18).

Ebenfalls im Vergleich zu den Klassen schlechter abgeschnitten, hat die Klasse `Pot`. Die Motive ähneln der Klasse `Club` und weisen keine markanten Merkmale auf.

Abbildung 4.19: Zwei Varianten der Klasse `Pot`

Bis auf die Pflanzen die aus dem Topf herausragen, ähnelt der Topf dem Motiv `Club`.

Die letzte Klasse des Modells, welche Auffälligkeiten zeigte, ist die Klasse `Ship`. Wie die Precision schon andeutet, waren die meisten Vorraussagen der Klasse `Ship` auch korrekt.

Die Auffälligkeiten dieser Klasse beziehen sich auf die Transferleistung, die im Optimalfall hätte geleistet werden können. So sind auf einigen Münzen, welche die Motive der Klasse `Ship` beeinhalten, ebenfalls noch weitere Motive enthalten, auf deren Klasse, das Modell trainiert wurde. So existieren Münzen auf denen sich nicht nur das Motiv der Klasse `Ship` befindet, sondern es finden sich in einigen Fällen zusätzlich das Motiv der Klasse `Standing Person` oder `Podium` auf der selben Münze (siehe Abb 4.20).

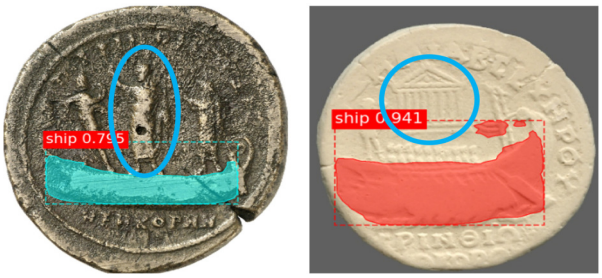


Abbildung 4.20: Münzen der Klasse `Ship` mit weiteren Klassen

In keinen der Testfälle war das Modell im Stande die zweite Klasse auf der Münze zu klassifizieren. Nichtsdestotrotz war es dem Modell zumindest möglich das Schiff korrekt zu klassifizieren, obwohl auch bei diesem Motiv eine Reihe an Varianten existierte. Neben den zusätzlichen Klassen gab es Motive auf denen das Schiff mit einem Segel abgebildet war aber auch Motive auf denen das Schiff kein Segel besaß.

Abbildung 4.21: Variante der Klasse `Ship` mit einem Segel

In Abb 4.21 sieht man zwei Beispiele auf denen das Motiv ein Segel besitzt. In beiden Fällen konnte das Modell das Segel korrekt identifizieren, auch wenn die Maskierung nicht perfekt ist.

Nun wurde im letzten Schritt noch geprüft, ob das Modell in der Lage ist Motive zu erkennen, die von einer anderen Münzstätte stammen. Dazu wurde die Münzstätte Pergamon aus der Versuchsreihe d.) gewählt. Die Schnittmenge der Klassen beider Versuchsreihen bestand aus den Klassen: `Head`, `Sitting Person`, `Podium` und `Bull`.

	Head	Sitting Person	Podium	Bull
#Vorrausgesagt	25	9	2	1
#Vorgegeben	25	10	10	10

Tabelle 4.11: Klassifizierungen des Modells e.) auf Bildern der Münzstätte Pergamon

Tabelle 4.11 zeigt, dass die beiden Klassen `Head` und `Sitting Person` deutlich besser abschneiden, als die Klassen `Podium` und `Bull`. Beim direkten Vergleich der beiden Trainingsdatensätze lässt sich eine Vermutung aufstellen, wieso die Klassen so unterschiedlich im Test abgeschnitten haben.

Bei den Klassen `Head` und `Sitting Person` ist zu erkennen, dass in den Trainingsdaten beider Münzstätten eine große Varianz der Motive innerhalb der Klassen existiert. Besonders bei den Portraits gibt es sehr viele unterschiedliche Motive. Bei der Klasse `Bull` ist auffällig, dass die Motive hautpsächlich den Kopf eines Bullen in verschiedenen Variationen abbilden. Münzen der selben Klasse aus der Münzstätte Perinthos enthalten hauptsächlich Motive von Bullen, die mit dem ganzen Körper abgebildet sind.

Abbildung 4.22: Zwei Münzen der Klasse `Bull`. Die linke Münze stammt aus Pergamon und die rechte aus Perinthos. Die roten Teilbereiche könnten sich für eine Zuordnung der Münzen eignen.

Abb 4.22 zeigt eine Münze aus der jeweiligen Münzstätte. Die rot markierten Teilbereiche der Motive eignen sich gegebenenfalls als markante Ausschnitte für die Bewertung der Klassifikation, in welcher beide Motive der selben Klasse zugeordnet werden würden. Leider konnte diese Transferleistung nicht von dem Modell erbracht werden.

Abbildung 4.23: Zwei Münzen der Klasse `Podium`. Die linke Münze stammt aus Pergamon und die rechte aus Perinthos. Die roten Markierungen zeigen die Unterschiede beider Motive hinsichtlich der Pfeiler.

Abb 4.23 zeigt den Unterschied der beiden Münzstätten im Hinblick auf die Klasse

`Podium`. Während die Bauwerke in der Münzstätte Pergamon meistens mit jeweils zwei Pfeilern links und zwei Pfeilern rechts dargestellt wird, werden die Bauwerke auf den Münzen aus Perinthos hauptsächlich mit durchgängigen Pfeilern abgebildet.

5 Diskussion

Im Folgenden Abschnitt werden Auffälligkeiten, die während der Bearbeitung dieser Arbeit aufgetreten sind besprochen. Bei der Annotation der Datensätze traten im Annotationsprozess einige Fragestellungen auf, die es zu klären gab. Die Art und Weise wie detailliert die Motive annotiert werden, ist eine dieser Fragen.

Abbildung 5.1: Unterschiedliche Möglichkeiten der Annotation des Flügels

Abb 5.1 zeigt das Motiv der Klasse `Griffin` und zwei Annotationen mit unterschiedlichem Detaillierungsgrad. Auf der linken Münze ist zu sehen, dass die Annotation die einzelnen Federn des Flügels mit einbezieht. Die Annotation der rechten Münze umrandet dagegen nur den Flügel als Ganzes. Im Rahmen dieser Arbeit wurde die rechte Variante als die umzusetzende Annotation gewählt. Ein aussschlaggebender Grund für die Entscheidung war die Zeitersparnis. Um die Menge an zu erstellenden Annotationen zu bewältigen, wurde der Detaillierungsgrad der rechten Variante gewählt. Ein weiterer Grund für die Entscheidung waren die Varianten innerhalb der Klassen.

Abbildung 5.2: Greifmotiv dessen Abbildung keine Federn an den Flügeln enthält

Abb 5.2 zeigt eine Variante des Motivs der Klasse `Griffin`, welche keine Federn aufweist. Mit der Wahl für die unpräzisere Variante der Annotation wurde versucht, eine einheitliche Repräsentation der Klasseninstanzen zu erreichen.

Ein weiteres Problem, welches bei der Annotation der Daten auftrat, ist die Tatsache, dass Motive, welche Elemente enthielten die sich überlappen, auf zweierlei Arten annotiert werden konnten. Zum einen ist es möglich die Form entsprechend der Realität abzutasten, zum anderen ist es möglich die Silhouette des Motivs abzutasten, ohne dass sich die Linien des Polygons kreuzen.

(a) Möglichkeit A (b) Möglichkeit B

Abbildung 5.3: Zwei Varianten der Annotation von einem Motiv, das sich überkreuzende Elemente beinhaltet.

Abb 5.3 soll dies verdeutlichen. An der Stelle, an denen sich die Beine der sitzenden Person kreuzen, sieht man den Unterschied zwischen den beiden Annotationen. Möglichkeit A zeigt wie sich die Linien des Polygons überschneiden. Bei Möglichkeit B orientiert sich das Polygon an der Silhouette der sitzenden Person. Ob und wie sich die zwei Arten der Annotation auf die Güte des Modells auswirken, ist ein interessanter Aspekt, welcher in zukünftigen Forschungen betrachtet werden sollte.

Ein weiterer nennenswerter Faktor dieser Arbeit war die Varianz der Dimensionen der verwendeten Bilder. Beginnend von den kleinsten Bildern in der Auflösung von circa 100x100 Pixel, reichte die Größe der Bilder bis hin zu einer Auflösung die 2000x2000 Pixel übersteigt. Im Rahmen dieser Arbeit wurde, die vom Modell vorgegebene Eingabegröße verwendet. Beim Einlesen der Daten werden die Bilder auf eine Größe von 1024x1024 Pixel skaliert. Ob durch eine einheitliche Größe der Eingabebilder eine Steigerung der Klassifikationsgüte erreicht werden kann, bleibt zu prüfen.

In Versuch e.) wurden die Pferde des Motivs, welche der Klasse `Quadriga` angehörten, der

Klasse `Double Horse` zugeordnet. Einerseits war dies nicht das gewünschte Ergebnis, da die Kombination aus Pferden, Reitern und Zugwagen als Motiv gesucht war. Andererseits bietet diese Teilklassifikation des Motivs auch Möglichkeiten. Je nach Problemstellung könnte dieses Verhalten des Modells auch wünschenswert sein. Beispielsweise wenn gezielt nach Pferdemotiven auf Münzen gesucht oder nach diesen geclustert werden sollen. Denkbar wäre eine Verallgemeinerung, sodass nicht nur nach Tieren auf den Motiven gesucht werden kann, sondern auch nach unterschiedlichen Objekten die auf den Münzen vorkommen.

Bei der Klasse `Ship` wurden die Teilmotive der Klasse `Podium` sowie `Standing Person` nicht erkannt. Für die erwähnte Teilklassifikation der Motive, könnte versucht werden alle einzelnen Klassen eines Motivs zu annotieren. In dieser Arbeit wurde darauf verzichtet, wenn die Teilmotive nur einen sehr kleinen Bereich des Bildes ausmachten.

Der letzte erwähnenswerte Aspekt ist der unterschiedliche Hintergrund. In dieser Arbeit wurde angenommen, dass das Abstraktionslevel des Modells in der Lage ist, den Hintergrund nicht in die Klassifikation miteinfließen zu lassen. Für weitere Versuche könnte evaluiert werden, ob das Entfernen beziehungsweise Vereinheitlichen des Hintergrundes der Eingabebilder zu einer Steigerung der Performanz beitragen würde.

6 Resümee und Fazit

Die Arbeit hatte zum Ziel mittels eines geeigneten Machine Learning Modells eine automatisierte Erkennung von unterschiedlichen Motiven auf diversen antiken Münzen zu entwickeln. Mit der Erkennung sollte es ermöglicht werden, Münzstätten mit ihrer Großzahl an verschiedenen Münzen und Motiven, aufzuteilen. Diese Aufteilung sollte der Feingranularität der Klassifizierung der Münzen entgegenwirken, um den Trainingsprozess eines Bildklassifikators und somit auch den Bildklassifikator selbst, zu verbessern.

Unter diesem Hintergrund wurde geprüft, ob sich Mask R-CNN als Modell eignet um diese Aufgabe zu lösen.

Zunächst mussten für erste Versuche geeignete Trainingsdaten, bestehend aus Münzbildern, gesammelt werden. Diese stammten aus unterschiedlichen Quellen. Die Bilder waren zum Teil annotiert und zum Teil ohne Annotationen. Ein Großteil der Münzen waren nach Münzstätten aufgeteilt.

Nach abgeschlossener Annotation wurde in einem ersten Versuch geprüft, ob das Modell generell in der Lage ist Motive auf Münzen zu erkennen. Dazu wurden drei unterschiedlich komplexe Motive gewählt und ein Modell trainiert. Die Unterschiede in der Klassifikationsgüte der einzelnen Klassen entsprachen den Erwartungen.

Im nächsten Versuch wurde getestet, ob das Modell in der Lage ist eine stehende Person von einer sitzenden zu unterscheiden. Das Modell konnte die unterschiedliche Haltung der Person erkennen und übertraf das vorherige Modell in jeder Metrik.

Mangels Daten brachte der dritte Versuch keine relevanten Ergebnisse.

In Versuch d.) wurden unterschiedliche Modelle geprüft, welche Bilder der Münzstätte Pergamon als Trainingsdaten verwendeten. Die Modelle unterschieden sich hinsichtlich der Menge an Trainingsdaten, sowie der Anzahl der Trainingsdurchläufe (Epochen). Die Erkenntnisse aus der Versuchsreihe wurden genutzt um die untersuchten Parameter in der nächsten Versuchsreihe anzuwenden.

Im letzten Versuch wurde ein Modell trainiert, welches Münzen der Münzstätte *Perinthos* klassifizieren sollte. Die Menge an Trainingsdaten und die Anzahl an Trainingsepochen wurde anhand der Ergebnisse des vorherigen Versuchs festgelegt.

Zudem wurde im letzten Versuch geprüft, ob das Modell im Stande ist Motive auf Münzen einer anderen Münzstätte zu klassifizieren. Es wurden dazu Münzen ausgewählt, deren Motivklassen in beiden Münzstätten vorkommen.

Erkenntnisse, die aus dieser Arbeit gewonnen wurden sind:

1. Modelle die mit polygonal annotierten Daten trainiert wurden, zeigten eine marginal bessere Performanz im Vergleich zu Modellen, die mit Bounding-Boxen trainiert wurden.

2. Die polygonale Annotation eines Motivs kann, je nach Vorgehensweise, unterschiedlich ausfallen, sodass beim kollaborativen annotieren die Vorgehensweise im Vorfeld festgelegt werden sollte.

3. Das Modell ist im Stande mehrere Instanzen einer Klasse auf einer Münze zu klassifizieren.

4. Je nach Klasse wurden auch unterschiedliche Varianten eines Motivs korrekt klassifiziert.

5. Modelle, die auf Daten einer Münzstätte trainiert wurden, sind im Stande Motive auf Münzen anderer Münzstätten zu klassifizieren, wenn die Varianz der Bilder innerhalb einer Klasse in den Trainingsdaten hinreichend groß war.

Die Ergebnisse dieser Arbeit haben verdeutlicht, dass die ersten Versuche der Motiverkennung mittels Mask R-CNN bereits gute Ergebnisse liefern konnte, obwohl die Menge an Trainingsdaten für den Bereich der Bilderkennung verhältnismäßig klein ausgefallen ist.
Für den Trainingsprozess sollte dann allerdings auf leistungsstärkere Hardware zurückgegriffen werden, da der Trainingsprozess bereits in dieser Arbeit eine beachtliche Zeit in Anspruch genommen hat. Der Trainingsprozess des letzten Modells aus der Versuchsreihe e.) dauerte ca. sieben Stunden.
Ein weiterer Faktor, welcher in zukünftigen Arbeiten betrachtet werden sollte, ist, ob die marginale Verbesserung des neuronalen Netzes bei Nutzung von Bildern mit polygonaler Annotation im Trainingsprozess den Mehraufwand im Annotationsprozess im Vergleich zur Annotation der Bilder mit Bounding-Boxen rechtfertigt. Eine Kosten-Nutzen-Analyse von Polygon versus Bounding-Box Annotation ist besonders bei größeren Datenmengen interessant. Da sich die Performanz neuronaler Netze mit wachsender Datenmenge in der Regel verbessert, wäre ein Vergleich der relativen Verbesserung der Modelle in Abhängigkeit der Annotationsart spannend.

6.1 Ausblick

Die Ergebnisse dieser Arbeit haben gezeigt, dass das Potenzial zur Nutzung von Mask R-CNN für die Erkennung von Münzmotiven vorhanden ist.

Die Verwendung von größeren Datenmengen, sowie weiteren Optimierungen hinsichtlich der Hyperparameter machen eine Verbesserung der automatisierten Erkennung denkbar.

Die Annotation der Trainingsdaten ist von großer Bedeutung für den Trainingsprozess und somit auch essentiell für die Verwendung von Mask R-CNN. Ein weiterer Ausblick der sich damit ergibt, ist die weitere Verbesserung von Annotationssoftware und der automatisierten Annotation, die einen schnelleren und qualitativ hochwertigeren Annotationprozess ermöglichen kann.

Für zukünftige Arbeiten wäre es denkbar größere Datenmengen mittels Data-Augmentation zu generieren und das neuronale Netz dann mit mehr Daten zu trainineren, um gegebenenfalls weitere Performanzverbesserungen zu erzielen. Entsprechende Tools wie *imgaug*[1] ermöglichen nicht nur die Augmentation der Bilder, sondern projizieren die Augmentation ebenfalls auf die dazugehörigen Annotationen. Damit entfällt der manuelle Annotationsaufwand. Ob die Augmentation von Trainingsdaten die Effektivität der Modelle steigern kann, müsste in nachfolgenden Arbeiten erörtert werden.

[1] https://imgaug.readthedocs.io/en/latest/

Anhang

Name	Link	Aktiv	Letztes Update	Video	Bounding Box	Polygon	Andere Annotationsarten	Automatisierungsgrad	Exportformate
Alturos.ImageAnnotation	https://github.com/AlturosDe	Ja	07.02.2020	Nein	Ja	Nein		Manuell	YOLO
Anno-Mage	https://github.com/virajmavani	Ja	12.04.2020	Nein	Ja	Nein		Manuell + semi-automatisiert	CSV, XML
Annotorious	https://github.com/recogito/a	Ja	31.08.2021	Nein	Ja	Ja		Manuell + Semi-automatisiert	Selber Programmieren über API
CATMAID	https://github.com/catmaid/C	Ja	21.09.2020	Nein					JSON
COCO Annotator	https://github.com/jsbroks/co	Ja	11.06.2021	Nein	Ja	Ja	Point	Manuell, semi-automatisiert	COCO
CVAT	https://github.com/openvinot	Ja	26.08.2021	Ja	Ja	Ja	Point, Cuboid, Polyline, Class Labels	Manuell, semi- und vollautomatisiert	VOC, COCO, YOLO + 12 weitere
deeplabel	https://github.com/ivelichmic	Ja	13.08.2021	Nein	Ja	Nein	Nein	Manuell, Model inference(pre-trained model trained on a small subset)	VOC, YOLO, COCO, KITTI, Google Cloud Platform
ImageTagger	https://github.com/bbl-bots/in	Ja	22.06.2021	Nein	Ja	Ja	Line, Point, Ball	Manuell, (semi-automatisiert,in progress)	unklar
imglab	https://github.com/NaturalIntl	Ja	02.12.2020	Nein	Ja	Ja	Circle, Point	Manuell, semi-automatisiert?	VOC, COCO + 2 weitere
labelImg	https://github.com/tzutalin/la	Ja	25.07.2021	Nein	Ja	Nein		Manuell	VOC, YOLO,
LabelMe	https://github.com/wkentaro/	Ja	27.07.2021	Ja	Ja	Ja	Circle, Polyline, Point, Class Labels	Manuell	VOC, COCO
LOST	https://github.com/l3lo-cv/los	Ja	19.03.2021	Nein	Ja	Ja	Line, Point	Manuell	CSV
make-sense	https://github.com/SkalskiP/	Ja	31.08.2021	Nein	Ja	Ja	Point, Line, Label	Manuell + Semi-automatisiert	CSV, YOLO, VOC-XML, VGG JSON, COCO JSON
MedTagger	https://github.com/medtagger	Ja	06.01.2021	Nein	Ja	Ja	Point	Manuell	
OpenLabeling	https://github.com/Cartucho/	Ja	07.05.2020	Ja	Ja	Nein		Manuell, Semi-automatisiert(ABER nur Edgedetector)	VOC, YOLO
OpenLabeler	https://github.com/kinhong/C	Ja	07.03.2021	Nein	Ja	Ja		Manuell + Annotation Tipps	VOC, COCO
pixelAnnotationTool	https://github.com/abreheret	Ja	02.06.2021	Nein	Nein	Nein	Brush + OpenCV watershed marked	Manuell, semi-automatisiert	PNG mit Maske
VIA	https://www.robots.ox.ac.uk/	Ja	04.03.2021	Ja	Ja	Ja	Circle, Point	Manuell	COCO, JSON, CSV
VoTT	https://github.com/microso	Ja	13.04.2021	Ja	Ja	Ja		Manuell	VOC,CSV, JSON,CNTK und weitere
Pixie	https://github.com/buni-rock/	Nein	10.01.2019						
Sloth	https://github.com/cvhciKIT/s	Nein	02.06.2017						
turktool	https://github.com/jaxony/tur	Nein	21.01.2018						
Web Annotation	https://github.com/mozanbe	Nein	15.08.2018						
yolo_mark	https://github.com/AlexeyAB/	Nein	24.4.2019						
Labelbox	KOSTENPFLICHTIG								

Das gesamte Projekt mit Quellcode wird unter folgendem Link hochgeladen:
https://hessenbox-a10.rz.uni-frankfurt.de/getlink/fiM1d4wqX6iqE2bvDtvpd/

Literaturverzeichnis

[Abd17] Waleed Abdulla. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow, 2017. Publication Title: GitHub repository.

[Bal15] Stephen Balaban. Deep learning and face recognition: the state of the art. *Biometric and surveillance technology for human and activity identification XII*, 9457:68–75, 2015. Publisher: SPIE.

[CLO20] CLOUDFACTORY. Image Annotation for Computer Vision: A Guide to Labeling Visual Data for Your Machine Learning Project. `https://www.cloudfactory.com/image-annotation-guide`, 2020. Besucht am 20.12.2022.

[Co15] Francois Chollet and others. Keras, 2015. Publisher: GitHub.

[DCK17] Sri Harsha Dumpala, Rupayan Chakraborty, and Sunil Kumar Kopparapu. k-FFNN: A priori knowledge infused Feed-forward Neural Networks, 2017.

[FHY19] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244–252, 2019.

[FSH04] K. Fatahalian, J. Sugerman, and P. Hanrahan. Understanding the Efficiency of GPU Algorithms for Matrix-Matrix Multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '04, pages 133–137, New York, NY, USA, 2004. Association for Computing Machinery. event-place: Grenoble, France.

[Gan18] Rohith Gandhi. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. `https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e`, 2018. Besucht am 20.12.2022.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GC09] Thiago S. Guzella and Walmir M. Caminhas. A review of machine learning approaches to Spam filtering. *Expert Systems with Applications*, 36(7):10206–10222, 2009.

[GDDM13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.

[Gir15] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. arXiv: 1504.08083.

[Han08] Allan Hanbury. A survey of methods for image annotation. *Journal of Visual Languages & Computing*, 19(5):617–627, 2008.

[HB20] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: state of the art. *International journal of multimedia information retrieval*, 9(3):171–189, 2020. Publisher: Springer.

[HE20] Bert Heinrichs and Simon B Eickhoff. Your evidence? Machine learning algorithms for medical diagnosis and prediction. *Human brain mapping*, 41(6):1435–1444, 2020. Publisher: Wiley Online Library.

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015.

[KBB+15] Rudolf Kruse, Christian Borgelt, Christian Braune, Frank Klawonn, Christian Moewes, and Matthias Steinbrecher. *Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze*. Springer Fachmedien Wiesbaden, Wiesbaden, 2015.

[Lan11] Adrian Lange. Diagramm einer McCulloch-Pitts-Zelle nach Minsk. `https://upload.wikimedia.org/wikipedia/commons/a/a7/Diagram_of_a_McCulloch-Pitts-cell.svg`, 2011. Besucht am 20.12.2022.

[LDG+16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection, 2016.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[MAP+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[Mou] Adil Moujahid. A Practical Introduction to Deep Learning with Caffe and Python // Adil Moujahid // Data Analytics and more. `http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/`. Besucht am 20.12.2022.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. Publisher: Springer.

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[MW01] David Meyer and FT Wien. Support vector machines. *R News*, 1(3):23–26, 2001. Publisher: R-News Austria.

[noa] How do Convolutional Neural Networks work? `https://e2eml.school/how_convolutional_neural_networks_work.html`. Besucht am 20.12.2022.

[noa21] What is a Container? - Docker. `https://www.docker.com/resources/what-container/`, November 2021.

[NRS+10] Nazri Mohd Nawi, R. S. Ransing, Mohd Najib Mohd Salleh, Rozaida Ghazali, and Norhamreeza Abdul Hamid. An Improved Back Propagation Neural Network Algorithm on Classification Problems. In Yanchun Zhang, Alfredo Cuzzocrea, Jianhua Ma, Kyo-il Chung, Tughrul Arslan, and Xiaofeng Song, editors, *Database Theory and Application, Bio-Science and Bio-Technology*, pages 177–188, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[PNdS20] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[RDGF15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. Publisher: American Psychological Association.

[Sch14] Juergen Schmidhuber. Deep Learning in Neural Networks: An Overview. April 2014.

[Sha20] Dhruvil Shah. Mask RCNN implementation on a custom dataset! `https://towardsdatascience.com/mask-rcnn-implementation-on-a-custom-dataset-fd9a878123d4`, December 2020. Besucht am 20.12.2022.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[Sin22] Aarohi Singla. Mask-R-CNN-on-Custom-Dataset, September 2022. original-date: 2021-01-16T16:23:21Z.

[SJZ21] Christoph Sager, Christian Janiesch, and Patrick Zschech. A survey of image labelling for computer vision applications. *Journal of Business Analytics*, 4(2):91–110, 2021. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/2573234X.2021.1908861.

[Smi81] P. R. Smith. Bilinear interpolation of digital images. *Ultramicroscopy*, 6(2):201–204, 1981.

[TLL21] Jigang Tang, Songbin Li, and Peng Liu. A review of lane detection methods based on deep learning. *Pattern Recognition*, 111:107623, 2021.

[UGS13] Jasper Uijlings, T. Gevers, and A.W.M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104:154–171, September 2013.

[Wu17] Jianxin Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[ZKS08] Sebastian Zambanini, Martin Kampel, and Mario Schlapke. On the Use of Computer Vision for Numismatic Research. pages 17–24, January 2008.

[ZKZ07] Maia Zaharieva, M. Kampel, and Sebastian Zambanini. Image Based Recognition of Ancient Coins. In *CAIP*, 2007.

[ZQD+21] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.