Institut für Informatik

Fachbereich Mathematik und Informatik

Masterarbeit

Neuronale Netze zur Bestimmung römischer Kaiser auf Bildern antiker Münzen

Sebastian Gampe

28.10.2021

eingereicht bei:

Dr. Karsten Tolle

Datenbanken und Informationssysteme (DBIS)

Erklärung zur Abschlussarbeit

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

Hiermit erkläre ich Herr Sebastian Gampe

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 abgegebenen schriftlichen gebundenen Versionen meiner Masterarbeit mit der auf dem Datenträger abgegebenen elektronischen Version übereinstimmen.

Frankfurt am Main, den 28.10.2021

_____

Sebastian Gampe

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Das Thema Geld beschäftigt Forscher verschiedener Disziplinen wegen seiner besonderen Bedeutung in der Menschheitsgeschichte seit der Antike. Die Wissenschaft, die sich hauptsächlich der Primärquelle Münze widmet, wird Numismatik genannt. Anhand dieser Quelle untersucht die Numismatik in Zusammenarbeit mit anderen Disziplinen die Wirtschafts- und Kulturgeschichte verschiedener Regionen der Erde.

Antike Münzen als Gegenstand der Forschung unterliegen besonderen Herausforderungen. Da diese Münzen hauptsächlich aus Lesefunden und Grabungen auf uns gekommen sind, ist ihr Erhaltungszustand oft mangelhaft, was besonders durch metallbedingte Beschädigungen wie Korrosion und den Abrieb durch lange Umlaufszeiten bedingt ist. Oft macht dies eine Restauration nötig, damit Numismatiker sich der Bestimmung einer Münze und der Typisierung von Münzreihen widmen können. Im Zuge dieser Tätigkeiten werden besonders das abgebildete Motiv auf Vorder- und Rückseite, die Legende und das verwendete Metall analysiert. Ein weiteres wichtiges Ziel hierbei ist die Identifizierung der sogenannten Münzstempel, die zum Schlagen der Münze verwendet wurden und deren Kombination für Vorder- und Rückseite die Grundlage eines Typus ist. Wenn es gelingt die Zeitstellung eines solchen Typus z.B. durch andere Funde möglichst genau eingrenzen zu können, dann können die zugehörigen Münzen als sogenannter „*terminus post quem*“ (Zeitpunkt nach dem etwas passiert ist) auch zur Datierung anderer Objekte und Strukturen verwendet werden.

Seit einiger Zeit gibt es fächerübergreifende Kooperationen zwischen Numismatik und der Informatik. Diese betreffen vor allem die Analyse von größeren numismatischen Datensätzen durch neue Software und beziehen auch das in den letzten beiden Jahrzehnten immer mehr an Bedeutung gewinnende Gebiet des *Machine Learning* (ML) mit ein. Im Besonderen wird versucht Münzen durch künstliche neurale Netze bestimmen zu lassen. Dies ist für Numismatiker eine sehr zeitaufwändige Tätigkeit, da jede Münze einzeln von Hand geprüft werden muss. Hier gibt es das Potential diese Netze zu sehr guten Hilfstools für die Numismatik zu entwickeln. Im Zuge dessen ist auch die vorliegende Arbeit zur Untersuchung der Verwendung von neuronalen Netzen zur Bestimmung von Mitgliedern der römischen Kaiserhäuser auf antiken Münzen entstanden. Im Mittelpunkt steht hier das Porträt einer Münze, anhand dessen Ikonographie die Identifizierung durch das Netz ausgeführt werden soll. Die dem Training und dem Test dieser Netze zugrundeliegenden Münzbilder wurden mittels

den Datenbanken der Projekte *Online Coins of the Roman Empire* (OCRE) und *Corpus Nummorum* (CN) zusammengetragen.

1.2 Aufgabenstellung

Die Aufgabenstellung dieser Arbeit beinhaltet das Training und die Auswertung eines Convolutional Neural Networks (CNN) zur Identifizierung einer römischen Kaiserin oder eines Kaisers anhand des Porträts einer antiken Münze. In der Bachelorarbeit des Autors wurde mit Testreihen auf OCRE und CN Bildern festgestellt, dass ein solches VGG16 Netz, welches von A. Loyal in ihrer Masterarbeit auf Bildern vollständiger Münzen trainiert wurde, in mehr als 50% der Fälle zumindest einen Teil der Klassifikation anhand der Legende einer Münze trifft (siehe Abb. 1)[1]. Durch den Ansatz nur noch das Porträt einer Münze für das Training des CNNs zu nutzen ist die vorliegende Arbeit somit eine Fortsetzung der beiden vorhergehenden Abschlussarbeiten.

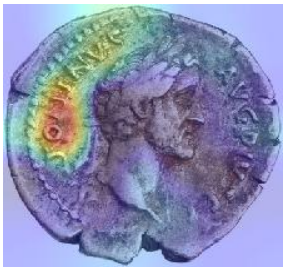

Abb. 1: Heatmap des Typus RIC III Antoninus Pius 54B aus der Bachelorarbeit des Autors, die den Fokus des von A. Loyal trainierten Modells auf die Münzlegende zeigt.

Die Zielsetzung gliedert sich im Einzelnen in die folgenden Punkte:

- Zuerst sollen verschiedenen Datensätze mit Kaiserporträts aus OCRE und CN, die sich in Anzahl und Format der unterscheiden als Grundlage für das Training der VGG16 Modelle erstellt werden.
- Das VGG16 CNN mit vortrainierten Gewichten, welches bereits von A. Loyal benutzt wurde, soll auf dem *Tensorflow Keras* Framework mit der Möglichkeit zur Modifikation der Hyperparameter implementiert werden. Außerdem sollen verschiedene Methoden zur Auswertung der Modelle umgesetzt werden.
- Es soll das Training mehrerer Modelle mit verschiedenen Datensätzen, Endsegmenten und Hyperparameter erfolgen, da diese Variablen großen Einfluss auf die Leitung eines

[1] Gampe 2019; Loyal 2018.

Modells nehmen können. Somit soll das am besten zur Porträtbestimmung geeignete Modell mittels der Performance-Metriken gefunden werden

- Abschließend soll eine Analyse des Vorhersageverhaltens des besten Modells auf den CN Porträts und Bildern von vollständigen Münzen aus der OCRE Datenbank erfolgen.

1.3 Vorgängerarbeiten

Die Möglichkeit mit *Machine Learning* Ansätzen Münzen zu bestimmen ist seit mehr als zwanzig Jahren in der Forschung erprobt worden. Frühe Ansätze nutzen aufgrund der damaligen Limitierung der Hardware meist noch Verfahren, die nicht auf einer Bilderkennung basierten wie beispielsweise die Arbeit von P. Davidsson, der 1998 Entscheidungsbäume zur Münzsortierung nutze[2]. Aber auch der Einsatz von künstlichen neuralen Netzwerken zur Münzerkennung ist kein neuer Ansatz mehr. Bereits 1992 konnten vier japanische Wissenschaftler zeigen, dass diese Art von Netzwerk zur Münzerkennung eingesetzt werden kann, obwohl ihr Modell nur zwei bestimmte Münzen unterscheiden kann[3]. Zum ersten Mal wurde ein *Convolutional Neural Networks* (*CNN*) 2015 von J. Kim und V. Pavlovic in der Numismatik eingesetzt. Ihr Ziel war es auf kaiserzeitlichen, römischen Münzen spezielle Orientierungspunkte und Regionen zu finden, mit welchen die einzelnen Klassen vorausgesagt werden können. Dazu gehörte auch eine Erkennung des auf den Münzen abgebildeten Kaisers. Das aus einer vorhergehenden Arbeit von drei Wissenschaftlern der Universität Toronto verwendete CNN, wurde auf dem ImageNet Datensatz vortrainiert und dann auf einer Bildmenge von ca. 4.500 Münzen weitertrainiert. Der mit der *Caffe* Bibliothek implementierte Ansatz konnte bei der Kaisererkennung eine *Top-1 Accuracy* von ca. 70% erreichen[4].

Neuere Arbeiten setzen dagegen meist auf die jetzt zur Verfügung stehende große Leistung von modernen Grafikkarten, die eine Bilderkennung auf wesentlich komplexeren neuronalen Netzen ermöglicht. Zu diesen gehört die Masterarbeit von A. Loyal „Maschinelles Lernen angewendet auf Bilder antiker Münzen“, die mehrere CNN Modelle für einen Einsatz zur Kaisererkennung und zur Münztypbestimmung getestet hat[5]. Letztendlich hat sie sich aufgrund ihrer Tests für ein vortrainiertes VGG16 Modell als Grundlage für beide Ziele entschieden. Ihre Ergebnisse zur Kaiserbestimmung auf den römischen Münzen fielen mit einer Top-1 Accuracy von ca. 91% und einer Top-5 Accuracy von ca. 92% bei 69 Klassen sehr vielversprechend aus.

[2] Davidsson 1998.
[3] Fukumi u. a. 1992.
[4] Kim –Pavlovic 2015; Das verwendete CNN Modell wurde hier vorgestellt: Krizhevsky u. a. 2012.
[5] Loyal 2018.

Auch die Typbestimmung konnte mit einer Top-1 Accuracy von 61% und einer Top-5 Accuracy von ca. 92% bei 195 Klassen vorhergehende Ansätze übertreffen[6].

In der Bachelorarbeit des Autors „Kombination maschineller Lernmethoden der Bild- und Texterkennung auf antiken Münzdaten“ wurde neben einer *Natural Language Processing* (*NLP*) Implementierung zur Auswertung von Münzbeschreibungen das von A. Loyal trainierte VGG16 Netz ausgewertet und sein Vorhersageverhalten überprüft[7]. Als Ergebnis konnte für die Bilderkennung die Aufdeckung von verschiedenen vier Fehlerklassen in einer Datenbank wie OCRE erprobt werden. Davon sind die ersten drei Klassen (CNN-1 – CNN-3) mit den Fähigkeiten des NLP Ansatzes gemein, während die Bilderkennung eine falsche Zuweisung einer Münze zu einem Typus (CNN-4) alleine erkennen kann[8]. Als größtes Problem der Kaisererkennung konnte das in Abschn. 1.2 beschriebene Problem der Heranziehung der Münzlegende zur Identifizierung des porträtierten Kaisers gezeigt werden. Auch ein durch Korrosion und starken Abrieb bedingter schlechter Erhaltungszustand einer Münze kann das Modell an die Grenze seiner Leistungsfähigkeit bringen[9].

Die hier vorliegende Arbeit versucht u.a. die Lösung des beschriebenen Legendenproblems zu bewerkstelligen.

1.4 Struktur der Arbeit

Die vorliegende Arbeit ist in fünf weitere Kapitel mit mehreren Unterabschnitten eingeteilt. Das zweite Kapitel widmet sich dem Forschungsgebiet der römischen Münzen. Zuerst soll in Abschnitt 2.1 ihre Bedeutung im römischen Reich als Zahlungsmittel und Ausdruck der kaiserlichen Repräsentation beschrieben werden. Danach werden in Abschnitt 2.2 und 2.3 die beiden Projekte OCRE und CNT, welche das Finden der hier verwendeten Münzbilder ermöglichten, näher vorgestellt.

Kapitel 3 beschäftigt sich mit den theoretischen Grundlagen des Machine Learning. Abschnitt 3.1 soll dazu das Gebiet des Machine Learning und seine Herausforderungen beleuchten. Den

[6] Loyal 2018, 73–78 mit Tab. 6.1. 6.3.
[7] Gampe 2019. Die NLP Umsetzung wurde auf basiert folgender Arbeit evaluiert: Klinger 2018. Dieser Ansatz wurde schließlich mit Beteiligung des Autors veröffentlicht: Klinger u. a. 2018.
[8] Gampe 2019, 34–37. Die drei ersten Klassen sind: CNN-1 – Personen auf der Münze wurden nicht im Porträtfeld eingetragen, CNN-2 – Porträteintrag und Person in der Bildbeschreibung sind unterschiedlich, CNN-3 – Zusätzliche Einträge von Porträts, obwohl diese nicht in der Beschreibung vorkommen. Die Klasse CNN-1 kann von dem Modell aufgrund des beschriebenen Inschriftenproblems und der Tatsache, dass der in Betracht kommende Kaiser nicht zu den trainierten 69 Klassen gehört, nur als theoretisch erkennbar eingeordnet werden.
[9] Gampe 2019, 29–33. Die Ergebnisse dieser Arbeit wurden auf der Konferenz „CAA – Computer Application and Quantitative Methods in Archeology“ 2019 in Krakau in der Machine Learning Session präsentiert und werden bei Erscheinung des zugehörigen Konfernzbandes (zum diesem Zeitpunkt noch in Vorbereitung) publiziert.

neuronalen Netzen und ihrer Unterart Convolutional Neural Networks ist dann der Abschnitt 3.2 gewidmet. Anschließend soll ein Blick auf die Evaluationsmetriken des ML in Abschnitt 3.3 geworfen werden. Die Abschnitte 3.4 und 3.5 besprechen das Problem der Erklärbarkeit der Ergebnisse von neuronalen Netzen und stellen die Grad-CAM Methode näher vor.

Das vierte Kapitel soll die hier durchgeführte Implementation der Datensätze und der trainierten Netze erläutern. Dazu wird in Abschnitt 4.1 die als Vorarbeit zu diesem Thema ausgeführte Porträtbereichserkennung mittels Bounding Boxes vorgestellt. Der Abschnitt 4.2 ist dann der Erstellung der zum Training der Modelle genutzten Datensätze gewidmet. Nach der Betrachtung der verwendeten Arbeitsumgebung in Abschnitt 4.3 wird die CNN Architektur VGG16 und ihre Implementation mit den relevanten Hyperparametern in Abschnitt 4.4 erläutert. Das Kapitel wird dann mit der Vorstellung der hier angewandten Auswertungsfunktionen abgeschlossen.

Kapitel 5 beschreibt dann die drei durchgeführten Experimente zum Training des VGG16 Netzes und deren Auswertung. In Experiment 1 (Abschn. 5.1) wird geprüft, ob sich der originale VGG16 Aufbau mit seiner 1000 Klassen Ausgabe für die Kaisererkennung nutzen lässt. Das zweite Experiment (Abschn. 5.2) soll die verschiedenen Kombinationen aus VGG16 Endsegmenten, Bildmengen, Lernraten und Epochenanzahlen mittels der Metriken eines ML Modells erläutern und ein in der römischen Numismatik möglichst vielseitig einsetzbares Modell finden. Das Experiment 3 (Abschn. 5.3) wertet dieses Modell und sein Vorhersageverhalten aus und soll eventuell vorhandene Probleme und die Gründe dafür aufzeigen.

Im sechsten Kapitel sollen die erarbeiteten Ergebnisse dieser Arbeit noch einmal zusammengefasst werden und es wird ein möglicher Ausblick auf daran anschließende Arbeiten gegeben.

2. Datenbasis

Das vorliegende Kapitel widmet sich ganz dem Anwendungsgebiet der hier untersuchten Machine Learning Verfahren: den römischen Münzen. Eine Münze ist definiert als Zahlungsmittel, das aus Metall nach einem festgelegten Standard gefertigt wurde und ein bestimmtes Design trägt[10]. Im ganzen römischen Reich waren Münzen, die diese Definition erfüllen, das Hauptzahlungsmittel. Daher soll in Abschnitt 2.1 zuerst auf diese Verwendung im römischen Reich der Kaiserzeit näher eingegangen werden. Die beiden anschließenden Abschnitte beschreibt die beiden Datenbanken *Online Coins of the Roman Empire* (OCRE) und *Corpus Nummorum* (CN) näher, mit welchen die zum Training der Modelle verwendeten Bilder gefunden wurden.

2.1 Römische Münzen

Die ersten als Zahlungsmittel verwendeten Münzen stammen aus dem lydischen Reich des 7. Jhs. v. Chr. Seitdem ist dieser praktikable, handliche Wertmesser durchgehend in verschiedenen Teilen der Welt in Verwendung gewesen. Im römischen Reich dienten Münzen spätestens seit dem 3. Jh. v. Chr. bis zum Ende des weströmischen Reiches im 5. Jh. n. Chr. als Bezahlung für Güter und Dienstleistungen. Daher sind aus dieser langen Periode tausende Münztypen und noch mehr ihnen zugeordnete Exemplare auf uns gekommen.

Münzen waren seit den Tagen der römischen Republik Träger politischer Botschaften und diese Funktion wurde in der Kaiserzeit ununterbrochen beibehalten. An dieser Stelle soll sich daher auf die Münzen der Kaiserzeit konzentriert werden, da diese die Porträts der Kaiserinnen und Kaiser tragen. Lebende Personen wie Mitglieder eines Kaiserhauses auf Münzen zu porträtieren, war eine vergleichsweise neue Entwicklung zu Beginn der Kaiserzeit. Julius Caesar war der erste lebende Römer, der auf diese Weise geehrt wurde. Sein Nachfolger und erster Kaiser Augustus griff ebenfalls darauf zurück und es wurde somit ein Merkmal, das bis zum Ende der Kaiserzeit beibehalten wurde. Doch nicht nur der Kaiser selbst war ein mögliches Porträtmotiv, auch die Mitglieder der kaiserlichen Familie waren in ihren Rollen als Kaiserin, Kaisermutter oder designierter Nachfolger mögliche Kandidaten. Das kaiserliche Porträt auf Münzen diente neben den rundplastischen Büsten und Statuen auch als Mittel zur Verbreitung des Abbildes eines Kaisers und hatte somit wohl auch einen Einfluss auf den privaten Bereich der römischen Bürger, beispielsweise bei der Nachahmung von Frisuren der Mitglieder des

[10] Howgego 1995, 1.

Kaiserhauses. Es wurden aber auch andere Ausdrücke kaiserlicher Repräsentation wie die Münzlegende, die oft die kaiserliche Titulatur enthielt, etabliert. Sie verkündete beispielsweise die Errungenschaften eines Kaisers verkündete[11]. Aber auch die Rückseite einer Münze trägt politische Botschaften in einer bestimmten Ikonographie. Hierbei überwiegen Themen wie militärische Erfolge durch die Darstellung von Waffen, Tropaia oder der Verkörperung besiegter Völker. So zeigt das berühmte Münzbild eines an einer Palme angeketteten Krokodils aus Nîmes die augusteische Eroberung von Ägypten nach der Schlacht von Actium (31 v. Chr., siehe Abb. 2). Neben den militärischen Themen war auch die Darstellung von Göttern, mit denen sich die Kaiser verbunden fühlten, beliebt. Dazu gehörte beispielsweise auch die Abbildung von kaiserlichen Tugenden mittels einer Personifikation. Möglich war es ebenso das Porträt des Vorgängers im Kaiseramt im Sinne einer Vergöttlichung abzubilden und somit auch seine eigene göttliche Herkunft zu betonen. Letztendlich konnte sich der Kaiser auch selbst mit göttlichen Attributen wie der sog. Strahlenkrone abbilden lassen (siehe Abb. 39 Links)[12].

Abb. 2: Bronze As des Augustus vom Typus RIC I (second edition) Augustus 158 aus Nemausus.

Bestimmte Zielgruppen, an welche die einzelnen Münztypen adressiert waren, lassen sich meist jedoch nicht ausmachen. Daher ist es auch schwierig bei der Münzikonographie von einer kaiserlichen Propaganda zu sprechen. Die Motive sind zwar von politischer Natur, aber eine große Wirkung in diesem Sinne scheint nicht bestanden zu haben. Jedoch gehen sicherlich auch einzelne Motive über die Funktion als Zeichen einer Ehrung hinaus[13].

Obwohl das römische Münzwesen stark zentralisiert war, wurden Münzen jedoch nicht nur in der Münzstätte in Rom geschlagen, sondern auch in den Provinzen, da somit der lokale Münzbedarf besser gedeckt wurde. Auch manche Städte hatten unter römischer Herrschaft immer noch das Recht eigene Münzen zu prägen. Besonders in den östlichen, eher griechisch geprägten Reichsteilen wurde diese lokale Prägetradition wesentlich länger beibehalten als im

[11] Als Beispiel sei der Beiname „*Germanicus*“, abgekürzt in der Legende „GER“, als Sieger über den sog. Volkstamm der Germanen erwähnt.
[12] Howgego 1995, 22 f. 58. 68–71. 74 f. 78–83.
[13] Howgego 1995, 70–72.Dazu gehört auch das von A. Loyal in ihrer Arbeit vorgestellte Münze des Marcus Iunus Brutus mit dem Motiv der „Freiheitsmütze“ zwischen zwei Dolchen und der Nennung der Iden des März in der Legende. Loyal 2018, 23 f.

Westen. Diese Prägungen tragen auf der Vorderseite seit der Herrschaft des Augustus oft das kaiserliche Porträt, während die Rückseite lokale Themen aufweist[14].

Neben der Vorder- und Rückseite (auch „*Avers*“ bzw. „*Revers*“ genannt) besitzen römische Münzen noch folgende Bestandteile:

- Porträt
- Legende
- Rückseitenbild
- Beizeichen
- Münzstätte
- Münzmeister
- Denomination
- Material
- Gewicht
- Stempel

Das **Porträt** ist meist auf der Vorderseite einer Münze zu finden. Es gibt aber auch Münztypen die sowohl auf dem Avers wie auch auf dem Revers ein Abbild unterschiedlicher Individuen tragen. Mit der **Legende** ist die Inschrift einer Münze gemeint. Sie gibt meistens die kaiserliche Titulatur und den Namen des Porträtierten wieder. Es können aber auch der Münzmeister, der Auftraggeber der Prägung oder der Stadtname erwähnt werden. Die Legende lässt sich sowohl auf Vorder- wie auch der Rückseite einer Münze finden. Das **Rückseitenbild** trägt meist eine Darstellung unterschiedlichster Themen. Dies können Porträts, Bauwerke, Götter, Personifikationen oder auch militärische Sujets sein (s.o.). **Beizeichen** in verschiedenen Varianten stehen meistens für die **Münzstätte**, in welcher die Münze geschlagen wurde. Dazu können sie auch Hinweise auf den die Prägung überwachenden **Münzmeister** geben. Die **Denomination** einer Münze gibt ihren Wert an. Römische Münzen sind nach **Material** und **Gewicht** in verschiedene Denominationen unterteilt. Zu den bekanntesten Einheiten gehören: Asse (Kupfer), Sesterze (Messing), Denare (Silber) und Aurei (Gold). Des Weiteren kann auch der Feingehalt von Edelmetallen in den Münzen bestimmt werden, da beispielsweise der Silberanteil in Denaren im Laufe der Kaiserzeit immer weiter abnahm. Dieser Wert ermöglicht es auch die Datierung einer Münze zu bestimmen[15]. Münzen wurden mittels einer Kombination von zwei unterschiedlichen **Stempeln**, die die Motive trugen, mittels eines Hammers

[14] Howgego 1995, 84 f. 100–102.
[15] Vgl. Loyal 2018, 24 f.; Howgego 1995, 115–121.

geschlagen. Diese wurden von den Stempelschneidern im Auftrag des Münzmeisters angefertigt. Diese Kombination der Stempel bildet in der Numismatik die Grundlage der Typisierung.

2.2 Online Coins of the Roman Empire (OCRE)

Als Grundlage der in dieser Arbeit trainierten ML Modelle kamen die reichsrömischen Prägungen der *„Online Coins of the Roman Empire"* (OCRE) Datenbank zum Einsatz. OCRE ist ein Gemeinschaftsprojekt der *American Numismatic Soci*ety (ANS) und dem I*nstitute for the Study of the Ancient World* (ISAW) der *New York University*. Das Projekt wurde mit dem Ziel gestartet ein digitaler Corpus für alle jemals publizierten reichsrömischen Münzen zu werden. Der Begriff „reichsrömisch" bezieht sich dabei auf Münzen, welche hauptsächlich in Rom und anderen wichtigen, offiziellen Münzstätten des Reiches geprägt worden sind (bspw. Lyon oder Trier). Diese sind durch den Münzumlauf in alle Teile des römischen Reiches gelangt und wurden somit auch in verschiedenen Ländern gefunden. OCRE soll dabei helfen die von mehr als 20 verschiedenen Institutionen zur Verfügung gestellten Münzen zu erforschen, zu identifizieren und zu katalogisieren. Das Projekt ist auf die Reihe des *Roman Imperial Coinage* (RIC), einem Standardwerk der antiken Numismatik, aufgebaut. Somit geben die RIC Bände auch die Zeitspanne der für OCRE in Frage kommenden Münzen vor: Diese reicht von der Gründung des römischen Kaiserreiches im Jahr 31 v. Chr. bis kurz nach dem Ende des weströmischen Kaiserreiches im Jahr 491 n. Chr. Daneben wurde ebenfalls die Typisierung der RIC in OCRE übernommen, die aus der RIC Bandnummer, einer Zuweisung an Prägeherr/-ort oder porträtierte Person und eine Nummer aufweist (z.B. RIC I (second edition) Augustus 155). Um für die aktuelle numismatische Forschung gut einsetzbar zu sein, kommen hier die im Nomisma Projekt festgelegten Standards zum Einsatz. Dazu gehören vor allem die nach der *Linked Open Data* (*LOD*) Methodologie des *Semantic Webs* erzeugten stabilen *Unified Ressource Identifiers* (*URI*) für jede Münze[16]. Um mit der Datenbank arbeiten zu können stehen ein *Ressource Description Framework (RDF)* Dump, der mittels der Sprache SPARQL abgefragt werden kann, und eine browserbasiertes Suchwerkzeug auf der Webseite zur Verfügung[17]. In dieser Arbeit werden die mittels OCRE gefundenen Bilder der

[16] Nomisma, <http://nomisma.org/> (25.06.2021); Beispiel für eine solche URI mit der oben genannten RIC Bezeichnung ist: RIC I (second edition) Augustus 155, <http://numismatics.org/ocre/id/ric.1(2).aug.155> (25.07.2021).
[17] Online Coins of the Roman Empire, <http://numismatics.org/ocre/> (25.06.2021); vgl. Gampe 2019, 3.

Münzvorderseiten mit den Kaiserporträts als primäre Quelle für das Training der verschiedenen Modelle benutzt.

2.3 Corpus Nummorum (CN)

Die zweite Quelle für Münzen mit römischen Kaiserporträts ist die Datenbank des *Corpus Nummorum* (CN), vormals *Corpus Nummorum Thracorum* (CNT), Projektes. CN ist ein Gemeinschaftsprojekt der Berlin-Brandenburgischen Akademie der Wissenschaften, des Münzkabinetts der staatlichen Museen zu Berlin und dem Big Data Lab der Goethe Universität Frankfurt am Main. Anders als OCRE konzentriert sich CN jedoch auf einen kleineren geographischen Raum mit jeweils eigener Städteprägung. Es sind die antiken Landschaften Thrakien, Moesia Inferior, Mysien und die Troas. Dieser Raum verteilt sich heutzutage vorwiegend auf das Balkangebiet (Südosteuropa) und die nordwestliche Türkei. Das Projekt dient der Erschließung der lokalen Münzprägung für die numismatische Forschung. Im Gegensatz zu den OCRE Münzen wurden die CN Exemplare in Münzstätten dieser Region geschlagen und haben auch hauptsächlich dort Verwendung gefunden. Somit wurden die Münzen auch vorwiegend in den oben genannten Ländern gefunden, verteilen sich aber heute auf verschiedene numismatische Institutionen weltweit. Der größte Unterschied zu den reichsrömischen Münzen ist die Benutzung des Altgriechischen für die Münzlegenden und die Abweichungen von den offiziellen stadtrömischen Kaiserporträts durch die lokalen Stempelschneider. Die Zeitspanne der Herstellung der im CN Projekt gesammelten Münzen umfasst die Spätarchaik bis zur späten römischen Kaiserzeit (2. Hälfte 6. Jh. v. Chr. bis 3. Jh. n. Chr.). Eine der Hauptaufgaben des Projektes ist es eine neue Typologie dieser Münzen zu erstellen. Diese Typologie orientiert sich dabei an den bereits vorhandenen lokalen Typologien und verbindet diese zu einem größerem Ganzen[18]. Die Namen jeder erfassten Münze und jedes Typs enthalten zurzeit die Bezeichnung „CN Coin“ oder „CN Type“ und eine fortlaufende Nummer. Die Namensgebung ist jedoch noch nicht endgültig. Daneben findet auch eine Übertragung der CN Daten in das RDF Format statt, so dass die Ergebnisse des Projektes auch auf dem Nomisma Portal veröffentlicht werden können. Bis dies abgeschlossen ist, sind die Münzen und ihre Typen auf der CN Webseite mit einer browserbasierten Suche abrufbar[19].

[18] Solche Typologien wurden bisher vor allem zu lokalen Städteprägungen erarbeitet. Als Beispiel sei hier die Typologie der Stadt Byzantion angeführt: Schönert-Geiß, Griechisches Münzwerk. Die Münzprägung von Byzantion 1-2 (Berlin 1970–1972).

[19] Corpus Nummorum, <https://www.corpus-nummorum.eu/about> (30.06.2021); vgl. Gampe 2019, 3 f.

3. Grundlagen

In diesem Kapitel soll ein Überblick über den theoretischen Hintergrund der in dieser Arbeit verwendeten *Machine Learning* (ML) Verfahren, Metriken und Frameworks gegeben werden. Zuerst wird das maschinelle Lernverfahren der *Artifical Neural Networks* (ANN) und das davon abgeleitete *Convolutional Neural Network* (CNN) behandelt. Darauffolgend werden die Metriken zur Auswertung des Lernerfolgs dieser Netze näher erläutert. Im nächsten Abschnitt wird auf die Erklärbarkeit der Ergebnisse von CNNs und die dazu verwendete Technik Grad-CAM eingegangen. Abschließend sollen die eingesetzten Frameworks näher erläutert werden.

3.1 Machine Learning

Machine Learning (ML) ist ein Teilgebiet der *Künstlichen Intelligenz* (KI). ML Algorithmen sind in der Lage aus vorhandenen Daten selbstständig zu lernen und sich somit zu verbessern. Dabei können diese Algorithmen Probleme bearbeiten, die normalerweise mit klassischen, explizit programmierten Applikationen kaum gelöst werden können.

Die Aufgaben, die ein ML System zu lösen hat, werden in die Kategorien *überwachtes* und *unüberwachtes Lernen* eingeteilt. Beide Arten werden anhand ihrer Erfahrung mit den Daten während des Lernens eingeteilt. Zum überwachten Lernen, bei dem der Algorithmus auch die Ziel- oder Ausgabewerte zur Verfügung gestellt bekommt, gehört auch die in dieser Arbeit angewandte Methode der *Klassifikation*. Dabei soll das ML System vorhandene Daten wie das Porträt eines römischen Kaisers in eine von mehreren Klassen wie beispielsweise „Augustus“ einteilen. Diese Klassen müssen dem System schon während dem Training bekannt gemacht werden, indem den Trainingsbildern die zugehörige Klasse als sog. *Groundtruth* beigegeben wird. Ein weiteres Beispiel für überwachtes Lernen ist die *Regression*. Diese Methode versucht anhand mehrerer Eingabevariablen einen numerischen Wert als Ausgabe zu finden. Dabei setzt der Algorithmus die Eingaben zueinander in Beziehung um den Zielwert zu ermitteln. Als Beispiel dafür kann eine Vorhersage zukünftiger Preise einer Ware anhand ihrer Herstellungskosten wie Material, Arbeitszeit usw. dienen. Im Gegensatz dazu steht das unüberwachte Lernen, bei dem es keine Vorgabe zur Einteilung der vorhandenen Daten oder Zielwerte gibt. Die klassische Aufgabe des unüberwachten Lernens ist das sog. *Clustering*. Hierbei werden die Daten von dem System selbstständig in verschiedene Cluster unterteilt. Das Ziel ist es, dass sich einander sehr ähnliche Datenpunkte in einem Cluster befinden. Die verschiedenen Cluster sollen aber möglichst gut unterscheidbar sein. Das Ziel ist es hier die gesamte Wahrscheinlichkeitsverteilung eines Datensatzes zu lernen.

Für die Evaluation eines ML Systems gibt die Möglichkeit seine Leistung oder Performance mittels verschiedener Metriken zu messen. Diese beziehen sich auf die Fähigkeit eines trainierten Modells richtige Vorhersagen zu treffen und somit seine Güte zu ermitteln. In dieser Arbeit werden solche Metriken zum Ermitteln der Fähigkeit eines Modells die richtige Kaiserin oder den richtigen Kaiser anhand seines Porträts zu ermitteln eingesetzt (siehe Abschn. 3.3).

Das wichtigste Ziel eines ML Systems ist eine möglichst gute Generalisierung zu erreichen. Das bedeutet, dass ein trainiertes Modell den generellen Trend in den Trainingsdaten so gut es geht erfassen soll. Wenn dies aber nicht gelingt, spricht man von der sog. *Über*- oder *Unteranpassung*. Diese beiden Probleme hängen eng mit den Trainings- und Testfehlern zusammen. Wenn ein Modell auf neuen und ungesehen Daten Fehler macht spricht man von Testfehlern. Analog dazu werden die Trainingsfehler auf den Trainingsdaten gemacht und dienen dazu dem Modell einen Lernerfolg zu ermöglichen. Im Laufe eines Trainingsvorgangs sollte die Trainings- und auch die Testfehleranzahl sinken. Wenn jedoch die Trainingsfehlerrate und auch die Testfehlerrate während des Trainings hoch bleibt, spricht man von einer Unteranpassung (*Underfitting*). Das Modell konnte mit den vorhandenen Daten nicht genug Lernen, um korrekte Vorhersagen zu machen. Wenn jedoch die Trainingsfehleranzahl sinkt, aber gleichzeitig die Testfehlerzahl hoch bleibt oder sogar steigt, wird eine Überanpassung (*Overfitting*) an die Trainingsdaten vorliegen. Das Modell hat sich möglichst genau an jeden einzelnen Datenpunkt der Trainingsdaten angepasst, so dass neue ungesehene Daten nicht mehr korrekt vorhergesagt werden können. Diese beiden Probleme zu vermeiden ist eine der zentralen Aufgaben auf dem Weg zu einem guten ML Modell und der ganze Vorgang bildet mit seinen Bestandteilen aus Daten, Algorithmen und Parametern ein Optimierungsproblem[20].

3.2 Convolutional Neural Networks

Artificial Neural Networks Die zu dem Oberbegriff *Deep Learning* (DL) gehörenden künstlichen neuronalen Netze sind heutzutage ein State-of-the-Art Verfahren des Machine Learnings mit zahlreichen Anwendungsgebieten wie Bild- oder Spracherkennung. Die Grundlage dieser Netze bildet das sog. *Perzeptron*, welches nach dem Vorbild der Reizverarbeitungsneuronen im menschlichen Gehirn entwickelt wurde. Das Prinzip des Perzeptrons funktioniert folgendermaßen: Verschiedene Eingangsvariablen werden zuerst mit zugeordneten Gewichtswerten verrechnet. Anschließend werden diese zu einer gewichteten Summe addiert und mittels einer Aktivierungsfunktion wird geprüft, ob diese Summe einen

[20] Goodfellow u. a. 2016, 97–99. 101–103. 108–110; Loyal 2018, 3–6.

Schwellenwert erreicht und somit das Perzeptron „feuert“ (einen Wert an ein anderes Perzeptron weitergibt) oder nicht[21]. Eine Schicht eines neuralen Netzwerks besteht aus mehreren Perzeptronen, die jeweils mit der vorhergehenden und der nachfolgenden Schicht verbunden sind (siehe Abb. 3).

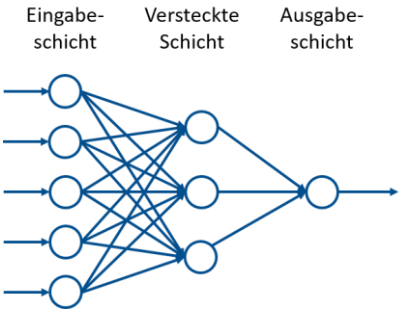

Abb. 3: Schematischer Aufbau eines Neuralen Netzwerks mit drei Schichten von Perzeptronen. Die mittlere Schicht ist jeweils mit der vorgehenden und der nachfolgenden Schicht voll verbunden.

Die einzelnen Schichten werden folgendermaßen unterteilt: Die Eingabeschicht nimmt die Eingabe in Form von Variablen (z.B. die Pixelwerte eines Bildes) in Empfang. Die Ausgabeschicht ist hat die Aufgabe die Vorhersage eines Netzes auszugeben. In den dazwischenliegenden, versteckten Schichten erfolgt die Umwandlung der Eingabe in die Ausgabe mittels des oben beschriebenen Perzeptron-Prinzips. Solche Netzwerke werden auch als *Feed-foward Neural Networks* bezeichnet, da die Ausgabe durch das rein Vorwärtsfließen der Eingabewerte durch das Netz (ohne Feedbackverbindungen) berechnet wird. Dieses Prinzip und die gleichzeitig große Skalierbarkeit solcher Netze führt auch dazu, dass das Verfahren der Fehlerrückführung (Gradienabstiegsmethode) zur Anpassung der Gewichte während des Trainings benutzt wird (siehe Abschn. 4.4.b)[22].

Convolutional Neural Networks Das in dieser Arbeit verwendete *Convolutional Neural Network* (CNN) ist eine Unterart der künstlichen neuronalen Netze. Ein CNN nutzt die namensgebende *Convolution* Operation (auch Faltungsoperation genannt) um die Kanten in einem Bild hervorzuheben. Ein CNN besteht grundsätzlich den bereits oben beschriebenen drei Schichten. Jedoch ist das erste Segment der versteckten Schichten aus Convolution gefolgt von Pooling Schichten aufgebaut. Am Ende des Convolutional/Pooling Abschnittes folgt ein Segment mit vollständig verbundenen Schichten (oder auch „fully connected layers“ genannt), wobei jedes Neuron in solch einer Schicht mit jedem anderen Neuron der nachfolgenden Schicht verbunden ist (siehe Abb. 4). Die Convolution/Pooling Schichten sind dagegen nicht

[21] Eine solche Aktivierungsfunktion ist die sog. Rectified Linear Unit (Relu), die auch in dem hier verwendeten VGG16 Netz benutzt wird (siehe Abschn. 4.4.a). Eine schematische Darstellung zu einem Perzeptron findet sich bei: Loyal 2018, 8 Abb. 2.3.

[22] [22] Goodfellow u. a. 2016, 164 f.; Loyal 2018, 7–9.

vollständig verbunden. Diese lokale Konnektivität stellt sicher, dass nachfolgende Neuronen nur auf lokale Reize ihrer Vorgänger reagieren (auch rezeptives Feld genannt). Die Faltungsoperation wird durch einen Filterkernel (oder auch Faltungsmatrix genannt) mit einer bestimmten, über alle Convolution Schichten gleichbleibenden Größe durchgeführt. Der Kernel wandert in Schritten über alle Pixel des Eingabebildes. Dabei wird ein inneres Produkt des Kernels mit den Pixelwerten errechnet und an die nachfolgende Schicht weitergereicht. Jedes Neuron der Convolution verwendet außerdem eine Aktivierungsfunktion (meist die sog. ReLU, siehe Abschn. 4.4.a). In einer Schicht werden jeweils mehrere unterschiedliche Filter auf die Eingabe angewandt, so dass sich deren Ausgabe immer leicht unterscheidet. Da der Filterkernel am Anfang aufgrund seiner Größe nur kleine Teile des Bildes auf einmal erfassen kann, werden somit auch nur einzelne Kanten der im Bild vorhandenen Strukturen entdeckt.

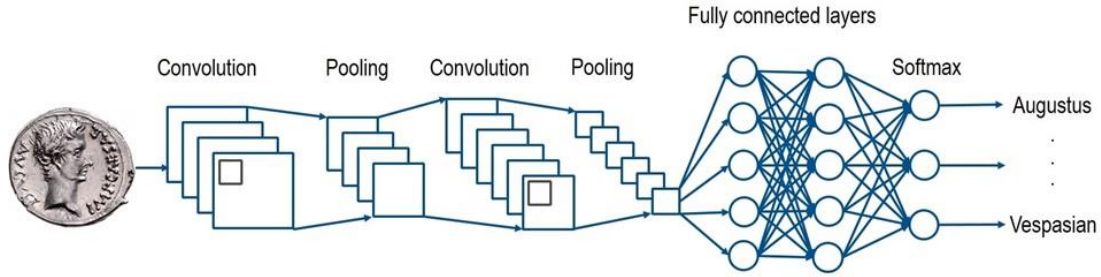

Abb. 4: Aufbau eines Convolutional Neural Networks. Der gleichbleibend große Filterkernel wird als schwarzer Kasten in den Convolution Schichten dargestellt.

Um größere Strukturen entdecken zu können, ist es nötig die Ausgabe einer Convolution Schicht zu verkleinern. Dies geschieht durch die sog. *Max Pooling* Operation. Hierbei wird wieder ein Filter einer bestimmten Größe über die Ausgabe einer Convolutional Schicht gelegt. Dieser Filter verändert die Werte dieser Ausgabe jedoch nicht, sondern sucht den größten Wert innerhalb des Feldes und verwirft alle anderen (siehe Abb. 5). Dieser Filter wandert jetzt über die ganze Ausgabe der vorhergehenden Faltungsschicht und am Ende entsteht ein kleineres Bild, in welchem die Kanten und ihre ungefähre Position erhalten bleiben. Die Pooling Operation sorgt ebenfalls dafür, dass durch die verkleinerte Ausgabe die Berechnungs-geschwindigkeit in den nächsten Schichten steigt. Anschließend wird in der nächsten Schicht wieder ein Convolution Filter mit derselben Größe wie zuvor auf die neue Eingabe aus der Pooling Schicht angewandt. Dieser Filter deckt jetzt auf dem kleineren Bild einen größeren Bereich als zuvor ab und kann somit nicht nur Kanten, sondern auch schon größere Strukturen wie Auge oder Nase eines Porträts hervorheben. Am Ende des Convolution/Pooling Segmentes wird solch ein Filter dann in wenigen Schritten die Strukturen des gesamten Porträts erfassen können. Anschließen werden die Ausgaben der letzten Pooling Schicht mittels einer Dimensionsreduktion zur Eingaben der ersten vollständig verbundene Schicht transformiert.

Innerhalb dieser Schichten findet jetzt die Aktivierung der einzelnen Neuronen nach dem Perzeptron Prinzip durch die Features dieser Eingabe statt. In der letzten Schicht des Netzwerks wird durch eine *Softmax* Funktion aus den ankommenden Aktivierungen die Wahrscheinlichkeitsverteilung der einzelnen Klassen errechnet und ausgegeben[23].

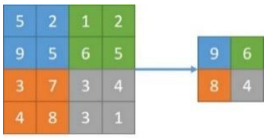

*Abb. 5: Ergebnis eines Max Poolings mit einer Filtergröße von 2*2 Pixeln und der Schrittgröße 2.*

Regional Convolutional Neural Networks Eine Unterart der Convolutional Neural Networks ist ein sog. *Regional Convolutional Neural Networks*, das in dieser Arbeit für die Porträtbereichserkennung verwendet wurde. In einem solchen Netzwerk werden im ersten Schritt mittels Bounding Boxes verschiedene, unterschiedliche große Regionen des Bildes vorgeschlagen. Danach wird mittels eines CNNs ein 4096 Einträge großer Feature Vektor für jede vorgeschlagene Region des Bildes extrahiert. Zuletzt werden die in den Boxen enthaltenen Objekte mittels des Vektors und klassenspezifischen *Support Vector Machine* (SVM) Modellen klassifiziert. Anschließend werden die Regionen, die mit einer hohen Wahrscheinlichkeit die gesuchte Klasse enthalten, als Bounding Box auf dem Eingabebild ausgegeben[24].

3.3 Evaluationsmetriken von ML Modellen

Als Verfahren zur Evaluierung eines zur Klassifizierung angewandten Machine Learning Models stehen verschiedene Metriken zur Verfügung. Diese werden in dieser Arbeit während und nach dem Training der einzelnen Modelle angewandt. Die hier angewandten Metriken sind:

- Confusion Matrix
- Accuracy
- Loss
- Precision
- Recall
- F1-Score
- Top-X Error

Um die Güte eines Modells bewerten zu können müssen die Ergebnisse seiner Vorhersage mit der sog. ***Groundtruth*** (GT) verglichen werden. Die ***Groundtruth*** gibt das richtige, vorher

[23] Goodfellow u. a. 2016, 326 f. 329–331. 335–339; Loyal 2018, 11–16.
[24] Girshick u. a. 2014, 1–4.

bereits bekannte Ergebnis an. Dieser Vergleich wird während des Trainings mit den Trainings- und Testmengen ausgeführt. Es wird gemessen wie gut das Modell die schon bekannten Daten der Trainingsmenge und die noch nicht gesehenen Daten der Testmenge vorhersagen kann. Die Ergebnisse der Klassifizierungen werden in vier Kategorien unterteilt:

1. ***Richtig Positiv (RP)***: Modell und Groundtruth geben Positiv aus.
2. ***Richtig Negativ (RN)***: Modell und Groundtruth geben Negativ aus.
3. ***Falsch Positiv (FP)***: Modell ermittelt Positiv, aber Groundtruth ist Negativ.
4. ***Falsch Negativ (FN)***: Modell ermittelt Negativ, aber Groundtruth ist Positiv.

Confusion Matrix: RP und RN beschreiben eine korrekte und FP und FN eine falsche Klassifikation. Gleichzeitig sind sie die Grundlage der sog. *Confusion Matrix* eines binären Klassifikators, welche die Anzahl der Ergebnisse der vier Kategorien in ihren Feldern enthält und die Vorhersage des Modells der Groundtruth gegenüberstellt (siehe Abb. 6).

		Modell	
		Positiv	Negativ
Groundtruth	Positiv	Richtig Positiv (RP)	Falsch Negativ (FN)
	Negativ	Falsch Positiv (FP)	Richtig Negativ (RN)

Abb. 6: Confusion Matrix eines binären Klassifikators.

Eine *Confusion Matrix* mit mehreren Klassen stellt dagegen alle Klassen in der Groundtruth gegen die Vorhersage des Modells auf. Die einzelnen Felder zählen die entsprechenden Überschneidungen. Diese können als Zahlenwerte oder auch als Farben in einer *Heatmap* angezeigt werden. Wenn eine Spalte solch einer Matrix durchgehend höhere Werte aufweist, ist das ein Anzeichen für ein mögliches Overfitting bei dieser Klasse (siehe Abb. 7).

		Modell		
		Klasse 1	Klasse 2	Klasse 3
Groundtruth	Klasse 1	10	10	0
	Klasse 2	2	15	3
	Klasse 3	0	8	12

Abb. 7: Confusion Matrix eines mehrklassigen Modells. In der Spalte der Klasse 2 ist eine Überanpassung durch die durchgehend hohen Werte gegeben.

Accuracy: Einer der wichtigsten Gütewerte eines trainierten ML Modells ist die sog. *Accuracy* (Genauigkeit der Klassifikation). Dieser Wert wird sowohl auf der Trainings- wie auch der Testmenge bestimmt, wobei der Wert der Testmenge aufgrund der dem Modell unbekannten Bilder der Wichtigere ist. Die *Accuracy* gibt das Verhältnis von richtigen zu falschen Klassifizierungen an. Die Berechnung für einen binären Klassifikator ist in Formel 1 zu sehen. Bei einem mehrklassigen Modell wird die Anzahl der korrekten Klassifizierungen durch die Anzahl aller Klassifizierungen geteilt. Dieses Maß ist aber bei verschiedenen Probleme wie

einem zu großen Ungleichgewicht bei der Trainingsdatenverteilung der Klassen alleine nicht aussagekräftig genug, daher berechnet man in diesem Fall auch die Werte *Recall* und *Precision*.

$$Accuracy = \frac{RP+RN}{RP+RN+FP+FN}$$

Formel 1: Berechnung der Accuracy eines binären Klassifikators

Loss: Die *Loss* Metrik (auch *Error* Funktion oder *Kostenfunktion* genannt) ist ein Maß für falsche Vorhersagen eines Modells. Die Funktion gibt generell an wie gut oder schlecht die Vorhersage für einen einzelnen Datenpunkt war. Im Zuge einer Klassifikationsaufgabe gibt die Funktion bei einer richtigen Vorhersage den Wert Null aus. Eine falsche Vorhersage erhält einen Wert höher als Null. Das Ziel des Trainings eines ML Modells ist ein möglichst niedriger durchschnittlicher Loss Wert.

Precision: Mit dem *Precision* Wert kann die Fähigkeit des Modells bewertet werden eine Klasse korrekt zu vorhersagen. Dazu wird der Anteil der richtigen klassenspezifischen Vorhersagen an den Gesamtvorhersagen des Modells für diese Klasse bestimmt. Die Berechnung für einen binären Klassifikator kann in Formel 2 abgelesen werden.

$$Precision = \frac{RP}{RP + FP}$$

Formel 2: Berechnung der Precison eines binären Klassifikators

Im Falle eines Modells mit mehr als zwei Klassen wird die Berechnung klassenweise für jede Spalte der *Confusion Matrix* durchgeführt. Dabei wird die Anzahl der richtigen Vorhersagen einer Klasse durch die Anzahl aller (richtigen und falschen) Vorhersagen dieser Klasse geteilt. Die *Precision* für das Modell wird dann als Durchschnitt über die *Precison* aller Klassen ausgegeben.

Recall: Der Anteil der richtigen Vorhersagen einer Klasse an dem Gesamtaufkommen der Datenpunkte dieser Klasse in der Groundtruth kann mit dem *Recall* Maß bestimmt werden (siehe Formel 3). Bei einem Modell mit mehreren Klassen wird die Berechnung klassenweise für jede Zeile der *Confusion Matrix* ausgeführt. Man teilt hier die richtigen Vorhersagen der Klasse durch alle vorhandenen Vertreter der Klasse in der Groundtruth. Für das Gesamtmodell wir der *Recall* dann als Durchschnittswert der *Recall* Werte aller Klassen ermittelt.

$$Recall = \frac{RP}{RP + FN}$$

Formel 3: Berechnung des Recalls eines binären Klassifikators

F1-Score: Wenn ein Ausgleich zwischen *Precision* und *Recall* gewünscht ist, kann der sog. *F1-Score* als harmonisches Mittel berechnet werden (siehe Formel 4).

$$F1 - Score \ = 2 * \frac{(Precision * Recall)}{(Precision + \ Recall)}$$

Formel 4: Berechnung des F-1 Scores aus Precision und Recall

Top-X Error: Der *Top-X Error* gibt die Fehlerrate für die wahrscheinlichsten X Klassen an. Er ist damit der Gegenwert zu der *Top-X Accuracy*. Das bedeutet, wenn ein Modell eine *Top-1 Accuracy* von 0,91 hat, beträgt der *Top-1 Error* damit 0,09. In dieser Arbeit werden *Top-1 Error* und *Top-5 Error* Werte angegeben[25].

3.4 Erklärbarkeit maschineller Lernmethoden

In den letzten Jahren erlebt das Forschungsgebiet der maschinellen Lernmethoden einen großen Aufschwung. Zur Bewältigung großer Datenmengen, die händisch kaum mehr zu bearbeiten sind, können ML basierte Modelle einen großen Anteil leisten. Dafür werden aber wiederrum große Datenmengen benötigt, um diese Modelle trainieren und testen zu können. Bevor es zu einem Einsatz als Hilfstool kommt, müssen aber den Personen, die keine Domain Experten im Gebiet ML sind, die eingesetzten Modelle, deren Ergebnisse und die komplexen Algorithmen, die sich dahinter verbergen, verständlich gemacht werden. Diese zu verstehen fällt auch Experten auf dem Gebiet des ML nicht immer leicht, da solche Modelle eine sog. „Black Box", in welche man keine Einsicht hat, bilden. Dafür kann die Komplexität der sehr großen Datenmengen und die zahlreichen Berechnungen während des Lernvorgangs verantwortlich gemacht werden. Wie ein Modell letztendlich seine Entscheidungen trifft, kann ohne eine Möglichkeit diesen Prozess sichtbar zu machen, einem ML Laien kaum erklärt werden. Damit die Modelle aber zur Unterstützung verschiedener Zielgruppen eingesetzt werden können, muss Vertrauen in deren Entscheidungen vorhanden sein. Letztendlich profitiert auch die ML Forschung selbst von transparenten Modellen, da sie unerwünschtes Verhalten und damit verbundene Risiken erklärbar machen und somit zu besseren Ergebnissen führen können. Beispielsweise sollte im dem Gebiet des autonomen Fahrens die Fehlerquote eines solchen Modelles durch die Nachvollziehbarkeit seiner Entscheidungen bestenfalls auf null gesetzt werden können, damit ein Endkunde sein Leben einem solchen Fahrzeug anvertrauen möchte[26].

In letzter Zeit hat Ziel der Erklärbarkeit maschineller Lernmethoden ein größeres Forschungsinteresse auf sich gezogen. Viele neue Ansätze dazu werden veröffentlicht und eingesetzt. Das Ziel ist hierbei, das Modell und sein Entscheidungsverhalten anhand von

[25] Loyal 2018 17 f.; Gampe 2019, 5 f.; Saleh 2018.
[26] Vgl. Gampe 2019, 6.

mehreren Beispielen auch einem Laien verständlich zu machen. Letztendlich kann dies auch zu einer verbesserten Zusammenarbeit von ML Experten und Domain Experten aus anderen Anwendungsgebieten führen. Es soll außerdem verhindert werden, dass Modelle Vorhersagen aufgrund völlig ungeeigneter Kriterien treffen, beispielsweise durch einen Bias in den zugrundeliegenden Trainingsdaten. Gleichzeitig kann die Forschung neue Erkenntnisse über den Trainingsvorgang selbst gewinnen, was zu besseren Ergebnissen bei einem erneuten Training mit neuen Klassen führen kann. Dies kann auch mit einer Kostensenkung verbunden sein. Letztendlich haben auch Geldgeber und Anwender von ML Modellen das Ziel möglichst gut verwertbare Ergebnisse durch nachvollziehbare Entscheidungen zu erhalten[27].

Dass eine gesteigerte Nachvollziehbarkeit der Entscheidungsfindung das Vertrauen von Anwendern in ein ML Modell erhöhen kann, konnte bei einem Experiment mit einem Random-Forrest Algorithmus gezeigt werden[28]. Dieses wurde im Anwendungsgebiet der Bioinformatik mit nicht-ML Experten durchgeführt und konnte durch seinen benutzerzentrierten Ansatz zur Transparenz des Modells das Verständnis und letztlich auch das Vertrauen in die Entscheidungen deutlich steigern[29].

3.5 Grad-CAM

Um die Erklärbarkeit der in dieser Arbeit trainierten Modelle zu steigern, wurde eine Methode verwendet, die auch schon in der Masterarbeit von A. Loyal und der Bachelorarbeit des Autors zur Anwendung kam: *Gradient-weighted Class Activation Mapping* (Grad-CAM)[30].

Dieses Verfahren wurde entwickelt um auf Convolutional Neural Networks mit vollständig verbundenen Schichten wie das in dieser Arbeit verwendete VGG16 benutzt zu werden. Die Arbeitsweise von Grad-CAM basiert darauf, dass eine Heatmap der Pixel eines durch das Netz propagierten Bildes erstellt wird, die durch ihren Farbverlauf anzeigt, welche Pixel die Neuronen des Netzes besonders stark aktiviert haben. Die Farben der Heatmap rangieren von blau nach rot, wobei blau keine Aktivierung und rot eine besonders starke Aktivierung anzeigt. Damit lassen sich die Bereiche ausmachen, welche für die getroffene Klassifikation

[27] Petkovic u. a. 2018, 204 f.; Selvaraju u. a. 2017, 1; DARPA. Explainable Artificial Intelligence, <https://www.darpa.mil/program/explainable-artificialintelligence> (24.07.2021); Vgl. Gampe 2019, 6 f.
[28] Random Forrest ist eine ML Lernmethode, welche durch mehrere Entscheidungsbäume für die Klassen (den sog. Wald) eine Mehrheitsentscheidung zur Klassifizierung durchführt. Petkovic u. a. 2018, 205.
[29] Ein weiterer Vorteil der Transparenz war, dass sogar die Anzahl der verwendeten Features bei kaum absinkender Genauigkeit reduziert werden konnte. Petkovic u. a. 2018, 212–214; vgl. Gampe 2019, 7.
[30] Selvaraju u. a. 2017, 1; Loyal 2018, 20; Gampe 2019, 7–9; Die Grad-CAM Implementation in dieser Arbeit wurde nach der folgenden Quelle implementiert: Keras. Grad-CAM class activation visualization, <https://keras.io/examples/vision/grad_cam> (27.07.2021).

verantwortlich sind. Nach dem Erstellen der Heatmap wird diese über das ursprüngliche Bild gelegt und man kann die Aktivierungszonen direkt auf dem Bild sehen. Das Beispiel auf Abb. 8 zeigt, dass das Modell in diesem Fall die Klasse „Augustus“ hauptsächlich an Stirn- und Haarbereich des Porträts erkannt hat. Der Übergang von Hals- zu Schulterbereich hat bei der Entscheidung auch eine kleinere Rolle gespielt. Der Haarbereich über dem Ohr ist dabei durch die Rotfärbung als besonders wichtig angesehen worden[31].

Abb. 8: Grad-CAM Heatmap einer Münze des Typs RIC I (second edition) Augustus 43.

Da die Methode auch klassendifferenziert arbeitet, kann auch man eine Heatmap für die weniger wahrscheinlichen Klassen erstellen. Das bedeutet, dass bis zu 66 Heatmaps entsprechend der hier trainierten Klassen erstellt werden können, wobei sich aber auf die wahrscheinlichste Klasse (Top-1) beschränkt wird.

Die Bestimmung der wichtigen Pixel mit der Grad-CAM Technik wird auf den Gradienten-Informationen der letzten Convolution Schicht des VGG16 Netzwerkes ausgeführt (siehe Abschn. 3.2). Zuerst wird das Bild durch das Netz propagiert und zur Erstellung der Heatmap als Ziel die wahrscheinlichste Klasse der Ausgabe der Softmax Schicht angegeben. Da nur diese Klasse von Interesse ist, wird deren Gradient auf eins gesetzt und der aller anderen Klassen auf null. Damit wird das Signal durch das Netzwerk zu den Input Neuronen zurückverfolgt. Im nächsten Schritt wird die Heatmap entsprechen der letzten Convolution Schicht als 14 * 14 großes Bild erstellt. Anschließend wird diese an das ursprüngliche Bildformat angeglichen und dann entsprechend der Verläufe der einzelnen Wärmebereiche mit einer weiteren Funktion zur besseren Visualisierung der wichtigen Bildflächen weichgezeichnet (siehe Abb. 8)[32].

Die neu erstellten Bilder mit den über das ursprüngliche Bild gelegten Heatmaps können jetzt beispielsweise Domain Experten vorgelegt werden, welche sich mit der Ikonographie des antiken Kaiserporträts auskennen. Da solche Porträts als Träger von Botschaften selten die reale Physiognomie des Kaisers wiedergaben, ist es ein Blick auf die verschiedenen Merkmale , an denen das Netz den abgebildeten Kaiser erkennt (z.B. Frisur, Kopfschmuck, Teile des Gesichts

[31] Selvaraju u. a. 2017, 1; Vgl. Gampe 2019, 7.
[32] Selvaraju u. a. 2017, 3 f.; Siehe dazu auch: Selvaraju u. a. 2017, 4 Abb. 2; Allerdings wird dort noch Guided Grad-CAM, das einem weiterentwickelten Ansatz entspricht, eingesetzt.

usw.), lohnend[33]. Für diesen Zweck wird eine größere Reihe an Heatmaps verschiedener Kaiser angelegt. Besonders interessant sind in diesem Fall die Klassen bei denen das Netzwerk eine besonders schlechte Erkennungsrate aufweist. Hier kann der Ansatz möglicherweise weitere Erkenntnisse liefern[34].

3.6 Frameworks: Tensorflow und Keras

Aufgrund des bereits erwähnten Aufschwungs der Anwendung der Neuronalen Netze in den letzten Jahren sind eine Reihe von Frameworks und Bibliotheken entstanden, die es ermöglichen verschiedene Netzstrukturen für diverse Anwendungszwecke zu erstellen. Dazu gehören die in dieser Arbeit verwendeten Bibliotheken *TensorFlow* und *Keras*[35]. A. Loyal verwendete in ihrer Arbeit ausschließlich das *Caffe* Framework, da sie durch den zur Verfügung stehenden Rechner mit vorinstalliertem *Caffe* darauf beschränkt war[36]. Zur Aufgabenstellung dieser Arbeit gehört es, dass die damalige Netzstruktur VGG16 und die verwendete Trainingsmenge auf *TensorFlow* (TF) übertragen werden sollte. Hierbei bot sich die *Keras – Python Deep Learning API* an, da diese ein Teil der *TensorFlow* API ist und bereits über eine Implementation des VGG16 Netzes und seiner trainierten Gewichte verfügt.

TensorFlow Die *TensorFlow* (TF) API wurde von Google's Brain Team als Open-Source Plattform entwickelt, um schnell und einfach Machine Learning basierte Anwendungen zu erstellen. Die Plattform ist im Netz frei verfügbar und der Code der zugehörigen Bibliotheken und Module kann unter GitHub eingesehen werden[37]. TF hat eine einfach zu benutzende Python Schnittstelle und kann wie jedes andere Modul auch nach der Installation in ein Jupyter Notebook importiert und sofort verwendet werden. Mittlerweile ist die Version 2.6.0 erschienen, aber in dieser Arbeit wurde aus Gründen der Komptabilität zu anderen Projekten die Version 1.14.0 eingesetzt.

Das Prinzip des TF Frameworks ist die Verwendung der namensgebenden Tensoren, welches ein anderer Begriff für multidimensionale Arrays ist. Diese Tensoren werden in Datenflussgraphen (oder auch „*Flowchart*" genannt) als Ein- und Ausgabe benutzt. Die in den

[33] Beispielsweise ließ sich der Kaiser Augustus noch in hohem Alter in den bekannten, kaum dem realen Alter angepassten Porträttypen darstellen. Hölscher 2016, 58 f.
[34] Vgl. Loyal 2018, 20; Vgl. Gampe 2019, 8 f.
[35] TensorFlow, <https://www.tensorflow.org> (27.07.2021); Keras. The Python deep learning API, <https://keras.io> (27.07.2021).
[36] Loyal 2018, 20 f.; Jia u. a. 2014. Caffe. Deep Learning Framework, <https://caffe.berkeleyvision.org> (27.07.2021).
[37] GitHub. Tensorflow/tensorflow: An Open Source Machine Learning Framework for Everyone, <https://github.com/tensorflow/tensorflow> (28.07.2021).

Graphen benutzten Knoten verkörpern die Operatoren, welche die Tensoren manipulieren und dann wiederrum als Tensor über eine Kante an den nächsten Knoten weiterschicken. Mit diesen Graphenprinzip können auch komplizierte mathematische Funktionen umgesetzt werden. Die Bestandteile eines Tensors sind Name, Struktur bzw. Dimension und der verwendete Datentyp. Das Framework kann auf CPUs wie auch GPUs ausgeführt werden. TF versucht selbst eine vorhandene Graphikkarte bei der Ausführung eines Graphen, der sog. Session, zu nutzen, da dies eine starke Beschleunigung der des Berechnungsvorgangs bewirken kann[38]. Weil ein Convolutional Neural ebenfalls auf einem Graphenprinzip aufbaut, kann diese ML Architektur in TF recht einfach umgesetzt werden. Der Eingabedatensatz muss in Tensoren umgewandelt werden und die einzelnen Schichten des CNNs werden als Knoten des Flowcharts, die über ihre Verbindungskanten die Eingabetensoren bearbeiten und dann an die nächste Schicht weiterleiten, definiert[39].

Keras Die *Keras – Python Deep Learning API* ist seit der TF Version 1.4 Teil des TensorFlow Frameworks. Ursprünglich wurde Keras als Ergebnis eines Forschungsprojektes 2015 von F. Chollet als Open Source Bibliothek speziell für *Deep Learning* veröffentlicht. Die TF eigene Keras Implementierung bringt den Vorteil mit, dass man die Daten mit den Möglichkeiten von TF zur Datenbearbeitungen nutzen kann. Keras selbst stellt dabei die High Level Bausteine für die Entwicklung von ML basierten Anwendungen bereit und bietet die Möglichkeit diese auf den verschiedenen Backends (TF, Theano, CNTK) laufen zu lassen. Es wird sowohl die Ausführung der Netze auf GPU als auch CPU unterstützt[40].

Aufgrund des TensorFlow Backends kann Keras ebenfalls mit Tensoren als Eingabe- und Ausgabedaten umgehen. Um eine Netzarchitektur zu definieren ist es ausreichen die einzelnen Schichten (bspw. Convloution-, Max Pooling und vollständig verbundene Schichten) mit ihren jeweiligen Größen in einem Jupyter Notebook als sequentielles Modell zu definieren. Anschließend können Verlustfunktionen und Metriken zur Überwachung des Trainings ausgewählt werden. Die fertige Architektur wird dann zu einem Modell kompiliert, welches im Anschluss mit den Tensor-Trainingsdaten den Lernvorgang ausführt. Trainierte Modelle können in einer Datei gesichert und zur Vorhersage neuer Daten wieder geladen werden. Keras bietet außerdem die Möglichkeit die Architektur des hier verwendeten VGG16 Modells und seine bereits trainierten Gewichte zu laden und diese dann weiter zu trainieren[41].

[38] Shukla – Fricklas 2018, 29. 31. 34–37.
[39] Shukla – Fricklas 2018, 172 f. 182–186.
[40] Chollet 2018, 3.2.0 f.
[41] Chollet 2018, 3.2.2.

4. Implementierung

An dieser Stelle soll ein Überblick über die Vorarbeiten und anschließende Implementierung des VGG16 Models in TensorFlow Keras gegeben werden. Zuerst werden die benutzten Bildmengen unterschiedlicher Größe und ihre Erstellung betrachtet. In Zuge dessen soll auch auf die Unterschiede zur Arbeit von A. Loyal eingegangen werden. Anschließend wird die Architektur des VGG16 Modells und die verwendeten Hyperparameter vorgestellt. Zuletzt sollen die verschiedenen Auswertungsimplementierungen beschrieben werden. In jedem der hier erwähnten Abschnitte werden außerdem die Unterschiede zu der Implementierung durch A. Loyal aufgezeigt[42]. Der geplante Workflow ist in Abb. 9 zu sehen: Nach der Generierung der Datensätze werden die Bildmengen mit der Porträtbereichserkennung auf der Arbeitsumgebung behandelt. Die ausgeschnittenen Bilder dienen als Eingabe für das VGG16 Modell. Das Modell und seine Hyperparameter sind der Ausgangspunkt der im nächsten Kapitel beschriebenen Experimente auf der beschriebenen Arbeitsumgebung. Die implementierten Auswertungsfunktionen sollen deren Evaluation ermöglichen.

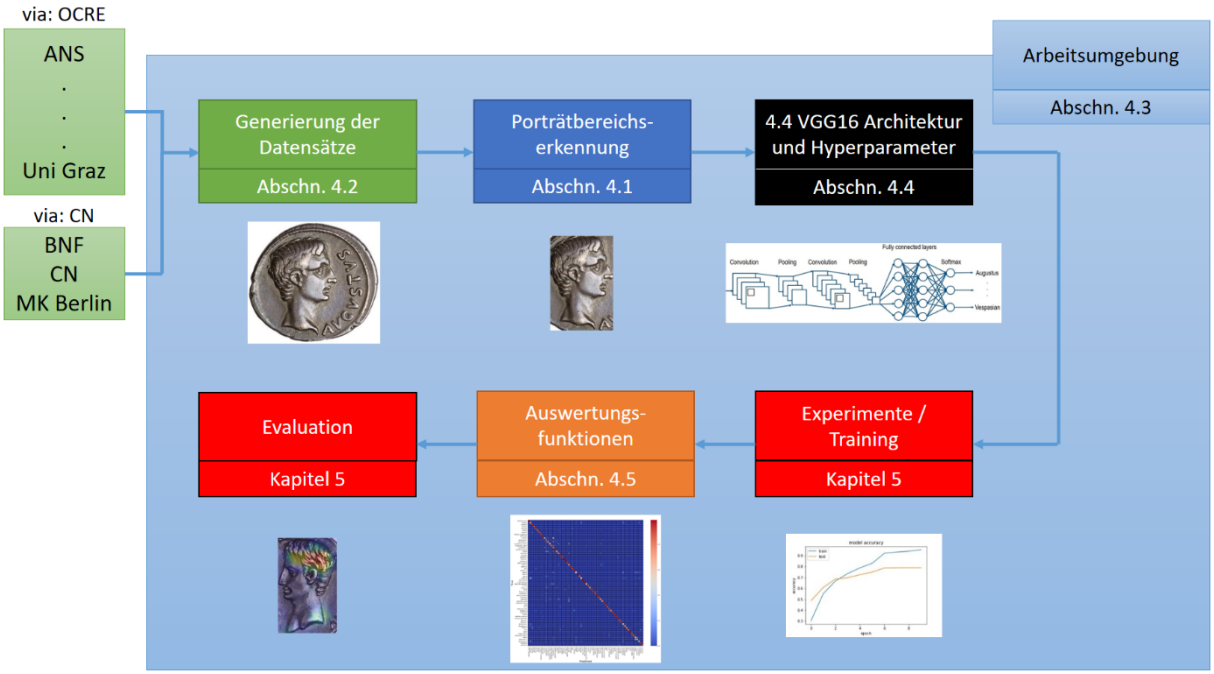

Abb. 9: Geplanter Workflow der in diesem Kapitel beschriebenen Implementierungen und der Verknüpfung zu Kapitel 5.

[42] Loyal 2018, 58–72.

4.1 Vorarbeiten: Porträtbereichserkennung mittels eines RCNN

Ein wichtiger Teilschritt vor der eigentlichen Implementierung war das Trennen des Porträtbereiches der Münze von ihrer Legende, um bei einem ML basierten Training zur Bestimmung des Kaisers oder der Kaiserin einen unerwünschten Fokus der Modelle auf diesen Bestandteil der Münze zu vermeiden (siehe Abschn. 1.2).

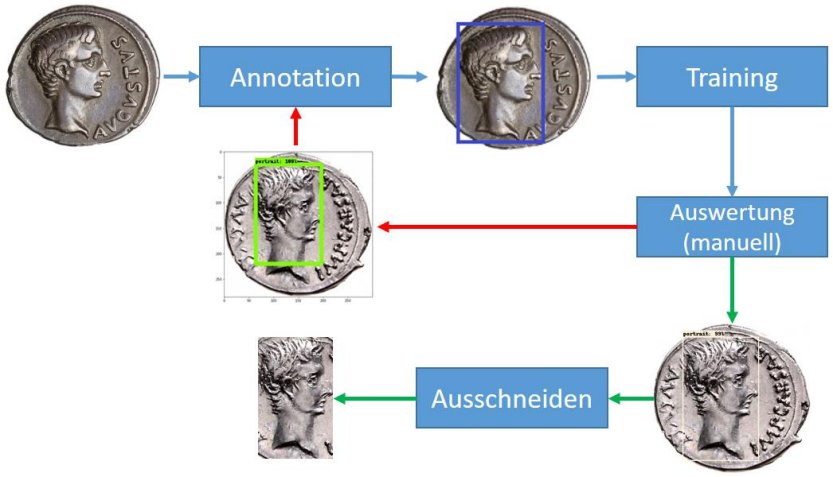

Abb. 10: Ablaufdiagramm der Porträtbereichserkennung. Legende: Rote Pfeile – Probleme bei der Auswertung entdeckt, grüne Pfeile – keine Probleme bei der Auswertung entdeckt.

Im Rahmen der Bachelor Arbeit des Autors wurde mittels Grad-CAM Heatmaps (siehe Abschn. 3.5) festgestellt, dass das von A. Loyal trainierte VGG16 Model zur Kaisererkennung bei über 50% der getesteten Münzen seine Entscheidung hauptsächlich oder zu einem großen Teil anhand der Münzlegende trifft[43]. Um dieses Problem anzugehen, wurde im Rahmen eines universitären Praktikums in Kooperation mit Patrick Bonack ein ML Model basierend auf einem *Regional Convolutional Neural Network* (*RCNN*) (siehe Abschn. 3.2) trainiert, welches den Porträtbereich einer Münze identifizieren soll[44]. Dieser sollte mittels einer sog. Bounding Box, die das Porträt vollständig umschließt und dabei möglichst wenige Bestandteile der Münzlegende miteinbezieht, ausgegeben werden (siehe Abb. 10 und Abb. 11).

Dazu wurden 200 zufällig ausgewählte Münzen mit Porträts verschiedener Kaiserinnen und Kaiser ausgewählt und mit dem Tool „LabelImg“ annotiert[45]. Für das Training wurde die *TensorFlow Object Detection API* und das auf einem RCNN basierende

[43] Loyal 2018; Gampe 2019 31–33.
[44] Das gemeinsame Projekt wurde 2019 auf dem 9. Workshop der Arbeitsgemeinschaft „Computer Anwendungen und Quantitativen Methoden in der Archäologie“ (CAA) einem Fachpublikum präsentiert. CAA. Programm des 9. Workshops der AG CAA Deutschland, <https://ag-caa.de/workshop2019/programm> (14.09.2021).
[45] GitHub, LabelImg, <https://github.com/tzutalin/labelImg> (27.07.2021).

„faster_rcnn_resnet101_coco_2018_01_28“ Model gewählt[46]. Dieses Model wurde bereits auf dem CoCo Datensatz vortrainiert und eignete sich deswegen für den beabsichtigten *Transfer Learning* (TL) Ansatz[47].

Abb. 11: Links: Mit LabelImg annotierte Münze des Typs RIC I (second edition) Augustus 43B. Rechts: Dieselbe Münze wurde nach dem Ausschneiden noch mit schwarzen Balken in ein quadratisches Format gebracht.

TL bezeichnet die Methode ein bereits trainiertes Model mit neuen Daten einem weiteren Training zu unterziehen. Dabei birgt den Vorteil, dass die auf eine bestimmte Problemstellung trainierten Gewichten des neuronalen Netzes bereits einen bestimmten Lernfortschritt repräsentieren. Dieser wird sich bei einem erneuten Training zunutze gemacht, indem neue Klassen aus einer ähnlichen Problemstellung durch neue, annotierte Daten gelernt werden können. Somit muss ein Modell nicht von Grund auf neu trainiert werden und es werden somit auch weniger Daten für das erfolgreiche Detektieren von weiteren Klassen benötigt. Es kann also zeit- und ressourcensparender gearbeitet werden[48].

Im nächsten Schritt wurden die im XML-Format vorliegenden Annotationen zusammen mit den Bildern in das TensorFlow eigene „Record“-Format per Script zu überführt[49]. Anschließend konnte das Training mit einem Train / Test Split von 160 zu 40 Bildern gestartet werden. Der Vorgang nahm auf dem in Abschn. 4.3 beschriebenen System bei ca. 200.000 Wiederholungen etwas mehr als 14 Stunden in Anspruch. Der *Localisation Loss*, welcher angibt wie weit die gesetzte Bounding Box von der Annotation der Testbilder abweicht, betrug am Ende 0,055 (siehe Abb. 12 Links).

Nach dem Training erfolgte noch eine Auswertung von Hand durch den Autor. Im Zuge dessen wurde festgestellt, dass das Model bereits für unsere Zwecke überzeugende Ergebnisse liefert. Lediglich der Hals- und Nasenbereich der Kaiserporträts wurden nicht immer ganz erfasst

[46] TensorFlow Object Detection API, <https://github.com/tensorflow/models/tree/master/research/object_detection> (27.07.2021); Model Download, <http:// download.tensorflow.org/models/object_detection/faster_rcnn_resnet101_coco_2018_01_28.tar.gz> (27.07.2021).
[47] Der *Common Objects in Context (CoCo)* Datensatz ist eine Sammlung von Bildern verschiedenster Objekte, die speziell für das Thema Object Detection erstellt wurde: CoCo, <https://cocodataset.org> (27.07.2021).
[48] Géron 2019.
[49] Quelle: Raccoon Detector Dataset, <https://github.com/datitran/raccoon_dataset> (27.07.2021).

(siehe Abb. 12 Rechts). Diese Probleme wurden durch eine großzügigere Einfassung der bereits annotierten Bilder und ein erneutes Training behoben. Der letzte Schritt umfasste das Hinzufügen einer automatischen Ausschneidefunktion, die den Porträtbereich innerhalb der von der Bounding Box vorgegebenen Eckpunkte des Bildes freistellt (siehe Abb. 15).

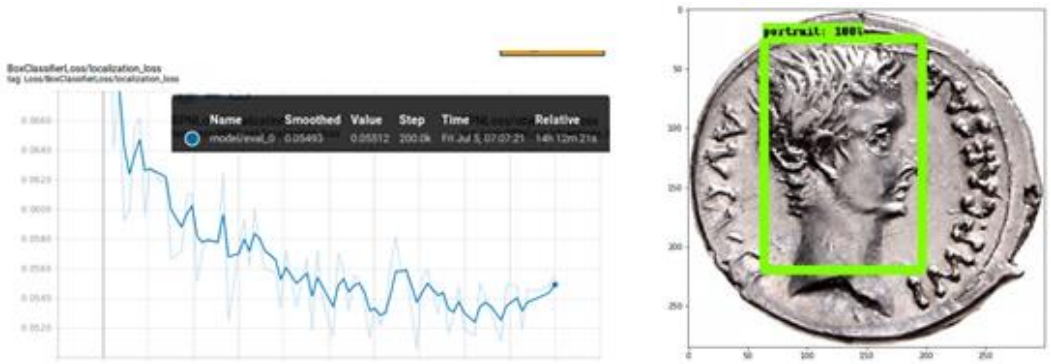

Abb. 12: Links: Localisation Loss des Trainings; Rechts: Ein Ergebnis des ersten Trainings.

4.2 Generierung der Datensätze

Der nächste Schritt der Implementierung umfasste nun die Generierung verschiedener Bilddatensätze für das Training, die anschließende Evaluierung und die Vorverarbeitung für die verschiedenen Experimente (siehe Abb. 13 und Tab. 3).

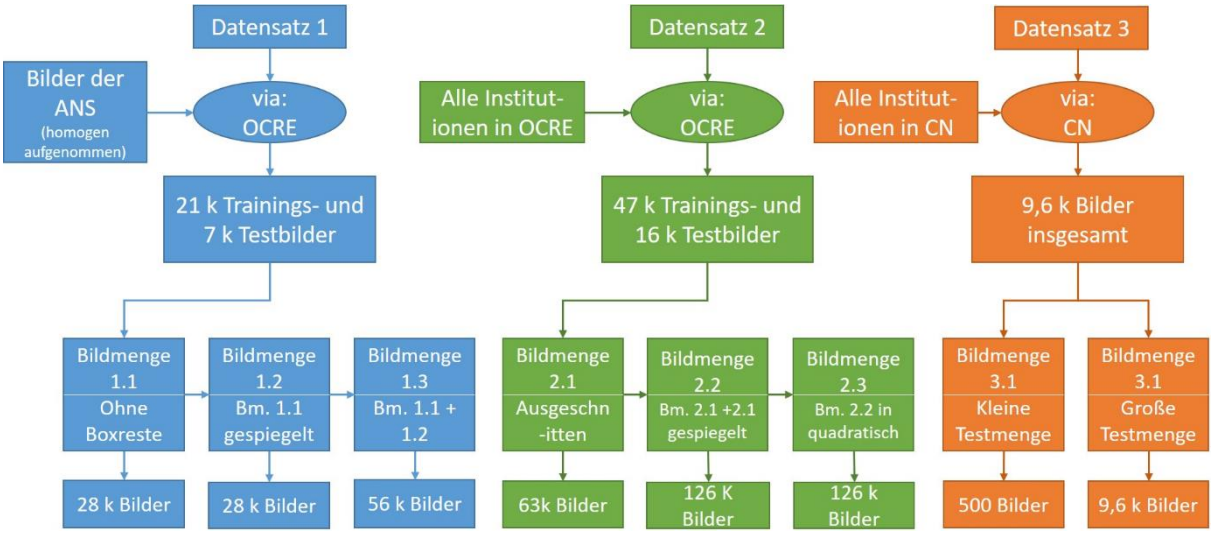

Abb. 13: Diagramm der verschiedenen hier vorgestellten Datenmengen

a) Unterschiede zu den Vorgängerarbeiten

An dieser Stelle soll zuerst auf einen wichtigen Unterschied zu der Masterarbeit von A. Loyal von 2018 hingewiesen werden, den Ausschluss dreier Klassen aus den ursprünglichen 69[50].

Die in dieser Arbeit verwendeten Namen von Mitgliedern des römischen Kaiserhauses stehen für die 69 verschiedenen Klassen, die entweder eine Kaiserin oder einen Kaiser repräsentieren. Für diese Arbeit wurden statt der von A. Loyal vorgegebenen 69 Klassen nur die Bilder von 66 Klassen verwendet, da die numerischen Codes „58396" und „60208" nicht für Kaiserinnen oder Kaiser stehen. Sie verweisen auf *Unified Ressource Identifier (URI)* des British Museum, die aber entgegen der Forderung nach Stabilität etablierter URIs zu diesem Zeitpunkt nicht mehr aufrufbar sind[51]. Der Code „58396" steht hier für das römische Konzept des (kaiserlichen bzw. göttlichen) Genius und hinter „60208" verbirgt sich die Göttin „Roma". Ebenso aussortiert wurde die Klasse „Constantinopolis", da diese die (göttliche) Personifikation der Stadt Konstantinopel darstellt. Während die „Roma" Münzen tatsächlich ein Porträt der Göttin Roma tragen, finden sich bei den im Ordner „58369" (Genius) gespeicherten Bildern verschiedene Kaiserporträts und keine Darstellung eines „Genius" (siehe Abb. 14).

Roma Der Fehler mit den Münzen der Göttin „Roma" (60208) lässt sich so erklären, dass der zu den Münzen gehörige *Ressource Descriptiuon Framework (RDF)* Dump der OCRE Datenbank aufgrund seiner Struktur nicht zwischen Gottheiten und Kaisern unterscheidet: Beide Entitäten sind durch das Prädikat „*nmo:hasPortrait*" an die Münze angebunden[52]. Dass die Entität „Roma" zu einer Gottheit gehört, sollte durch ihre weiteren Eigenschaften (unter der British Museum URI) klar werden[53]. Da aber die von A. Loyal benutzte Abfrage lediglich nach Vorderseiten mit dem *„nmo:hasPortrait"* Prädikat sucht, wurden auf diese Weise die „Roma" Münzen irrtümlich gefunden[54]. Da sich in dem „60208" Ordner somit nur Münzen mit der Darstellung der Göttin „Roma" befinden, wurden diese aussortiert.

---

[50] Loyal 2018.

[51] Genius, <http://collection.britishmuseum.org/id/person-institution/58396> (nicht aufrufbar, 28.07.2021); Roma, <http://collection.britishmuseum.org/id/person-institution/60208> (nicht aufrufbar, 28.07.2021); Eine Zeitlang waren die URIs des British Museum noch verfügbar, wenn man eine ResearchSpace Adresse vorangestellt hat, bspw.: <https://public.researchspace.org/resource/?uri= http://collection.britishmuseum.org/id/person-institution/58396>; Leider funktioniert dies zum Zeitpunkt des Schreibens dieser Arbeit ebenfalls nicht mehr. Bei Gruber 2018 wird der Thesaurus des British Museums sogar als ein Beispiel für Linked Open Data genannt: Gruber 2018, 19.

[52] Nomisma. „nmo:hasPortrrait" , <http://nomisma.org/ontology#hasPortrait> (28.07.2021).

[53] Der Entität unter der URI „http://collection.britishmuseum.org/id/person-institution/60208" können keine weiteren Eigenschaften zugeordnet werden, da diese Adresse zu diesem Zeitpunkt weiterhin nicht erreichbar ist. Da die Abfrage von A. Loyal diese Eigenschaften jedoch nicht berücksichtigt, ist dieser Umstand für die hier angesprochene Problematik nicht weiter relevant. Roma. <http://collection.britishmuseum.org/id/person-institution/60208> (nicht aufrufbar, 28.07.2021).

[54] Ein Beispiel für einen Typ mit „Roma" auf der Vorderseite und dessen zugewiesene Münze ist: American Numismatic Society. RIC II Trajan 769, <http://numismatics.org/ocre/id/ric.2.tr.769> (28.07.2021); Der von A.

Genius Das Problem mit den Genius-Münzen (58369) ist ähnlich gelagert: Hier wird die Personifikation „Genius“ ebenfalls mit *„nmo:hasPortrait“* an die Vorderseite der Typen und den zugeordneten Münzen verbunden. An dieser Stelle lässt sich ein direkter Bezug zur Bachelor Arbeit des Autors ziehen: Dort wurde mittels der Anwendung eines auf CN Daten trainierten NLP Models auf einen OCRE Datenbank Dump entdeckt, dass Münzen der Typusklasse *RIC VI Treveri* sehr oft einen falsch zugewiesenen Vorderseiten-Porträteintrag besitzen (NER-3 Fehlerklasse)[55]. Dies trifft auch auf Münzen zu, die auf der Vorderseite ein Kaiserporträt und auf der Rückseite die Darstellung eines „Genius“ tragen. Hier treffen nun die oben bereits erwähnte Problematik mit der *„nmo:hasPortrait“* Abfrage auf einen falsch zugewiesenen Porträteintrag auf der Vorderseite des Typs. Daher befinden sich in dem „58396“ Ordner 187 Münzen mit den Porträts verschiedener Kaiser, die jedoch nur in langwieriger Handarbeit der korrekten Klasse zugewiesen werden können (siehe Abb. 14)[56]. Durch das Vorhandensein mehrerer Kaiserklassen können auch die schlechten Werte der Genius Klasse beim F1-Score (0,16) in der Auswertung von A. Loyal erklären, wie die Autorin bereits selbst festgestellt hat[57]. Damit hat die „Genius“ Klasse aber auch das Potential die Erkennung anderer Klassen zu verschlechtern und wird mit ihren zugeordneten Münzen ebenso entfernt.

Abb. 14: Fälschlich der Klasse „58369“ (Genius) zugewiesene Münze des Typs RIC VI Treveri 139b mit dem Porträt des Kaisers Maximian.

Constantinopolis Die letzte weggefallene Klasse „Constantinopolis“ stellt einen leicht anderen Fall dar. Auch hier ist eine Personifikation (im weiteren Sinne eine Gottheit) auf der Vorderseite der Münzen abgebildet. Die Münzen wurden wiederum aufgrund der *„nmo:hasPortrait“* Problematik der von A. Loyal verwendeten Abfrage heruntergeladen.

Loyal verwendete Dump der AMS enthält nur die Münzen aus dieser Sammlung, daher befand sich lediglich der Silber Denar der AMS in diesem Ordner, nicht jedoch der Denar in Berlin; American Numismatic Society. Silver Denarius of Trajan, Rome, AD 98 - AD 117 1994.78.1, <http://numismatics.org/collection/1994.78.1> (28.07.21).

[55] Gampe 2019, 25 f.

[56] Es gibt 142 Bilder im „train“ und 45 Bilder im „test“ Ordner; Das von A. Loyal benutzte Programm hat diese Münzen ohne Typangabe in dem Ordner gespeichert. Als Beispiel für einen Typ mit falscher Zuweisung und einem Vertreter aus der AMS Sammlung in dem „58369“ Ordner ist: American Numismatic Society. RIC VI Treveri 139b, <http://numismatics.org/ocre/id/ric.6.tri.139b> (28.07.2021).

[57] Loyal 2018, 52 f. 74 Tab. 6.1; Es kann in der Tabelle abgelesen werden wie sämtliche Werte der „58396“ Klasse deutlich gegenüber den anderen Klassen abfallen.

Jedoch besitzt die Entität „Constantinopolis“ keine URI des British Museum (und auch keine andere) und könnte somit auch nicht als Personifikation/Gottheit mit einer SPARQL Abfrage identifiziert werden. Dies kann nur mit dem entsprechenden Wissen geleistet werden. Daher wird diese Klasse ebenfalls aussortiert[58].

b) Datensatz-1

Ein Teil in dieser Arbeit verwendeten Bilder der OCRE Datenbank wurden bereits von A. Loyal in ihrer Masterarbeit extrahiert. Dazu wurde eine vorher von M. Jostock erstellte Java Applikation verwendet, die mittels einer SPARQL Abfrage die benötigten Bilder von der American Numismatic Society (ANS) Webseite herunterlädt[59]. Die ANS als alleinige Quelle wurde gewählt, um eine möglichst hohe Homogenität der Bilder sicherzustellen. So sollten alle Bilder möglichst denselben Hintergrund haben[60]. Jedoch kann durch das in Abschn. 4.1 beschriebene Verfahren zur Trennung des Porträtbereichs von den anderen Bestandteilen der Münze sichergestellt werden, dass nur noch ein minimaler Teil des Hintergrundbereiches in dem ausgeschnittenen Bild vorhanden ist. Ein Bias ist somit unwahrscheinlich[61]. Außerdem wurden lediglich die Vorderseitenbilder der abgefragten Münzen verwendet, da diese im Allgemeinen das Porträt der Kaiserin oder des Kaisers tragen (siehe Abschn. 2.1). Anschließend wurden die Bilder mittels Anpassung der SPARQL Abfrage zur Auslesung des Porträtfeldes („*portraitOBV“*) in die folgende Ordnerstruktur ablegt[62]:

/CoinPortraitData/train/Kaisername und *CoinPortraitData/test/Kaisername*

Nach dem Herunterladen der Bilder hat A. Loyal eine Trainings-, Test, und Validierungsmenge mittels einer weiteren Java Applikation erstellt. Dabei verwendete sie eine Randomisierungsfunktion um einen Bias durch die Auswahl von Hand auszuschließen. Die Menge der Bilder beträgt ca. 23.000 in der Trainingsmenge und ca. 7000 jeweils in Test- und Validierungsmenge. Diese Vorauswahl sollte sicherstellen, dass die für ihre Experimente verwendeten Bildermengen immer dieselben sind und somit eine bessere Vergleichbarkeit der Modelle gewährleistet wird[63]. In dieser Arbeit werden in Datensatz 1 lediglich Trainings- und Testmenge der von A. Loyal erstellten Datenmengen verwendet, da im nachfolgend

[58] Ein Beispiel für eine solche Münze ist folgendem Typus zugewiesen: OCRE. RIC VII Lugdunum 246 <http://numismatics.org/ocre/id/ric.7.lug.246> (29.07.2021); American Numismatic Society. Bronze AE3 of Constantine I, Lugdunum, AD 330 - AD 331 1925.999.30, <http://numismatics.org/collection/1925.999.30> (29.07.2021).
[59] Jostock 2016.
[60] Loyal 2018, 38.
[61] Das Format unterscheidet sich bei diesen Bildern zwar in Höhe und Breite, die aber durch das Porträt selbst festgelegt werden. Somit ist ein durch äußere Umstände begründeter Bias ebenso unwahrscheinlich.
[62] Loyal 2018, 63–65; Die verwendete SPARQL Abfrage: Loyal 2018, 64 Abb. 5.2; vgl. Gampe 2019, 13.
[63] Loyal 2018, 64 f.; Die Experimente von A. Loyal beinhaltet u.a. die Nutzung verschiedener CNN Netzarchitekturen: Loyal 2018, 54–57.

beschriebenen Datensatz 2 ebenfalls keine Validierungsmenge erstellt wurde, um die Bildermenge von Klassen, die sowieso schon wenig Bilder beinhalten, nicht noch weiter zu verkleinern[64]. Die Bilder wurden für die hier beschriebenen Experimente mit der Porträtbereichserkennung bearbeitet und in einem neuen Ordner *„output“* und *„output_test“* und nach Kaisernamen geordneten Unterordnern neu gespeichert. Jeder Ordner enthält das vollständige Münzbild mit der darübergelegten Bounding Box der Porträtbereichserkennung. Weiterhin gibt es drei Unterorder (*„cropped1“* bis *„cropped3“*), die die ausgeschnittenen Porträts in verschiedenen Ausführungen beinhalten. Die Menge in Ordner „*cropped1“* enthält die ausgeschnittenen Bilder mit den Resten der oberen und linken Rahmenlinie der Bounding Box (Abb. 15 Links). Dies auf ein Problem beim Ausschneiden des Porträtbereichs zurückzuführen und deswegen wurden diese Bilder hier nicht verwendet. Bildmenge 1.1 in Ordner *„cropped2“* enthält die vollständig ausgeschnittenen Porträts ohne die Reste der Box, die als Grundlage der Experimente 1 und 2 dienen (siehe Abschn. 5.1, 5.2 und Abb. 15 Mitte)[65].

Abb. 15: Links: Ausgeschnittener Porträtbereich des Typs RIC I (second edition) Augustus 116 mit Überresten der Bounding Box links und oben; Mitte: Dieselbe Münze nach Entfernen der Überreste; Rechts: Gespiegelte Version derselben Münze. Zur besseren Sichtbarkeit der Boxproblematik wurden die Münzbilder mit einem schwarzen Rahmen umgeben.

Für diese Experimente wurde schließlich noch eine gespiegelte Version der ausgeschnittenen Porträts von *„cropped2“* mit der *NumPy* Funktion *„np.fliplr()“* angelegt und in *„cropped3“* als Bildmenge 1.2 gespeichert (siehe Abb. 15 Rechts)[66]. Getestet wurde in diesen Experimenten außerdem noch eine Kombination von Bildmenge 1.1 und 1.2 als Bildmenge 1.3., um die Datenbasis des Trainings künstlich zu vergrößern.

[64] Da die Datensatz 2 Menge ebenfalls sämtliche Bilder der ANS beinhaltet, werden die Bilder der Validierungsmenge also trotzdem genutzt.
[65] Die Überreste der Bounding Box wurden mit der *NumPy* Funktion *„np.delete()“* durch das Entfernen der Randpixel in der ersten Spalte und der ersten Reihe des Bildes beseitigt. NumPy. numpy.delete, <https://numpy.org/doc/stable/reference/generated/numpy.delete.html> (29.07.2021).
[66] NumPy. numpy.fliplr, <https://numpy.org/doc/stable/reference/generated/numpy.fliplr.html> (29.07.2021).

c) Datensatz-2

Der Datensatz-2 wurde vom Autor im Rahmen seiner Bachelorarbeit zur Auswertung des von A. Loyal trainierten VGG16 Models angelegt[67]. Die zugehörigen Bilder wurden ebenfalls mit der Java Applikation von M. Jostock aus dem Dump der OCRE Datenbank heruntergeladen. Hierbei wurden mittels Modifikation des Codes alle verfügbaren Bildquellen herangezogen. Hier kann ein Bias durch unterschiedliche Hintergründe und Bildformate durch den Einsatz der Porträtbereichserkennung als unwahrscheinlich eingestuft werden. Von den ca. 100.000 Bildern in der OCRE Datenbank wurden auf diese Weise ca. 64.000 Vorderseitenbilder mit Porträts der 66 Kaiserinnen und Kaiser (Klassen) heruntergeladen werden. Diese wurden ebenfalls in den zugehörigen Namensordnern abgelegt. Nach dem Ausschneiden des Porträtbereiches wurden aus diesem Datensatz für die verschiedenen in dieser Arbeit beschriebenen Experimente insgesamt drei Unterdatensätze gemacht. Bildmenge 2.1 (Ordner „*ds01"*) enthält die nicht modifizierten 64.000 ausgeschnittenen Porträtbereiche der Münzen. In Bildmenge 2.2 (Ordner „*ds02"*) wurden die nicht-modifizierten Bilder und ihre neu erstellten, gespiegelten Pendants zu einer ca. 128.000 Bilder großen Menge vereinigt. Die Bilder im letzten Bildmenge (Bildmenge 2.3, Ordner „*ds03*") wurden dagegen noch einmal einer größeren Modifikation unterzogen. Mittels einer Funktion zur Umwandlung des hochrechteckigen Porträtformats in ein quadratisches Bild mit schwarzen Seitenrändern wurden die 128.000 Bilder aus Bildmenge 2.2 bearbeitet und abgespeichert (siehe Abb. 11 Rechts)[68]. Da alle drei Bildmengen unterschiedlich sind, mussten sie jeweils in Trainings- und Testmengen neu aufgeteilt werden. Dazu wurde die Funktion *„train_test_split()"* mit einer 0,75 zu 0,25 Aufteilung aus der *scikitk-learn* Bibliothek genutzt[69]. Die neu erstellten Trainings- und Testmengen wurden jeweils in der folgenden Ordnerstruktur abgelegt:

- */CoinPortraitData_all/ds0x/train/Kaisername*
- */CoinPortraitData_all/ds0x/test/Kaisername*

d) Datensatz-3

Die letzte hier verwendete Bildmenge besteht aus Münzbildern mit Kaiserporträts der CN Datenbank. Diese Bilder dienen in erster Linie dazu die mit den OCRE Daten trainierten VGG16 Modelle weiter zu evaluieren und eine mögliche Übertragbarkeit auf andere Datensätze

[67] Gampe 2019, 13 f. 19.
[68] Quelle der verwendeten Funktion: Resizing Images for Convolutional Neural Networks, <https://gist.github.com/CMCDragonkai/cf25ad969e3a7fc1e48d60ecf20a2253> (30.07.2021).
[69] scikit-learn. sklearn.model_selection.train_test_split, <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html> (29.07.2021).

zu überprüfen. Für das Herunterladen der Bilder wurde ein eigenes Jupyter Notebook angelegt, welches mit Hilfe der *„pandas“* Bibliothek auf einen Dump der CN *MySQL* Datenbank zugreift. Mittels einer Abfrage wurden sämtliche Bilder mit einem Porträt der 66 verschiedenen Kaiserinnen und Kaiser heruntergeladen (siehe Abschn. 9.1 Listing 1)[70]. Die in der CN Datenbank verwendeten Bilder kommen von mehreren Institutionen. Die erste Quelle ist der Server des CN Projektes selber. Dieser stellt auch als einziger Abgussfotos von originalen Münzen bereit[71]. Daneben werden noch die Server des Münzkabinettes der staatlichen Museen zu Berlin (IKMK) und der Bibliothèque nationale de France (BNF) abgefragt[72]. Aufgrund dieser unterschiedlichen Quellen ist eine Vorverarbeitung der Abfragedaten in dem Notebook vor dem eigentlichen Downloadprozess nötig[73]. Der Datensatz 3 wurde ebenfalls in zwei verschiedene Bildmengen eingeteilt: Bildmenge 3.1 enthält etwas mehr als 500 Bilder und wurde im Rahmen der Bachelor Arbeit des Autors erstellt. Mit dieser Bildmenge konnte dort das Problem der Einbeziehung der Münzlegende bei der Klassifikation des Porträts erstmals gezeigt werden[74]. Abgelegt wurde diese Bildmenge nach der Bearbeitung mit der Porträtbereichserkennung unter der folgenden Ordnerstruktur:

/CNTTrainedClasses/output/Kaisername.

Da die Bildmenge 3.1 lediglich zehn Kaiserklassen umfasst, wurde im Rahmen der vorliegenden Arbeit weitere Bilder aus der CN Datenbank geladen. Bildmenge 3.2 beinhaltet 9562 Bilder aus 30 der 66 trainierten Kaiserklassen. Da die Fundmenge der CN Münzen nur ein begrenztes Gebiet des römischen Reiches abbilden (siehe Abschn. 2.2), können nicht alle 66 Kaiserklassen mit Bildern repräsentiert werden. Die CN Bilder bestehen jedoch nicht nur aus Fotos der Originalmünzen, sondern es werden auch Bilder von Gipsabgüssen originaler Münzen verwendet. Daher wurden diese Bildmenge während des Herunterladens in die Ordner *„originals“* und *„plastercasts“* getrennt. Diese beiden Bildmengen werden in Experiment 3 genutzt, um festzustellen, ob es Unterschiede in der die Erkennung von Originalen und Abgüssen gibt (siehe Abschn. 5.3.b). Weiterhin wird dort ebenfalls untersucht, ob die Erkennung von abfotografierten und gescannten Abgüssen Unterschiede aufweist. Die ganze Bildmenge 3.2 wurde nach der Freistellung des Porträtbereiches wie folgt gespeichert:

[70] pandas – Python Data Analysis Library, <https://pandas.pydata.org> (30.07.2021).
[71] Viele der Münzen in der CNT Datenbank werden nur durch solche Abgüsse repräsentiert, da kein Originalfoto veröffentlicht werden kann. Dies kann beispielsweise der Fall sein, wenn die Münze nach dem Abguss an einen privaten Sammler verkauft wurde.
[72] Münzkabinett – Staatliche Museen zu Berlin, <https://ikmk.smb.museum/> (01.08.2021); Bibliothèque nationale de France, <https://www.bnf.fr> (01.08.2021).
[73] Dazu gehört u.a. die Erstellung der vollständigen URLs, die Erstellung eines Datasets mit Originalmünzen und Abgüssen und die Umwandlung aus dem TIFF Format in das JPEG Format.
[74] Gampe 2019, 31 f.

- */CNTTrainedClasses/all_coins_test/output/originals/Kaisername*
- */CNTTrainedClasses/all_coins_test/output/plastercasts/Kaisername*

Alle im Rahmen dieser Arbeit neu erstellten Münzbilder wurden im *JPEG* Format gespeichert, das kompatibel zu dem verwendeten *VGG16* Netz ist. Die ebenfalls kompatiblen *PNG* Bilder wurden beim Test der vollständigen Münzbilder beibehalten, jedoch in Datensatz 1 bis 3 im Zuge des Herunterladens und neu Abspeicherns in das *JPEG* Format umgewandelt. In Datensatz 3 stehen viele Bilder (vor allem die Abgussbilder) im *TIFF* Format zum Download bereit. Da dieses Format aber nicht für die Eingabe in das CNN kompatibel ist, wurden diese Bilder auch in das *JPEG* Format konvertiert.

4.3 Arbeitsumgebung

Das Training und die Evaluation der Modelle, die in der vorliegenden Arbeit besprochen werden, wurde auf dem folgenden Rechner ausgeführt[75]:

- *CPU: AMD Ryzen 5 1600 6-Kern Prozessor*
- *RAM: 16 GB*
- *GPU: NVIDIA Geforce 1080 GTX (8 GB RAM)*

Der Vorteil dieses Rechners beim Trainieren von ML Modellen ist seine leistungsstarke Grafikkarte, die es ermöglicht einen vollständigen Trainingsdurchlauf in wenigen Stunden abzuschließen. Dies liegt vor allem an der hervorragenden Parallelisierbarkeit (concurrency) von CNN Berechnungen, die auf den vielen parallelen Rechenwerken einer Grafikkarte gleichzeitig ausgeführt werden können. Ein Training auf der CPU würde dagegen mehrere Tage in Anspruch nehmen[76]. Um das zum Training und Ausführung der Netze genutzte *TensorFlow Keras* (Version 2.2.4-tf) mit GPU-Beschleunigung auf dem *Ubuntu* 18.04.4 LTS Betriebssystem ausführen zu können, war es nötig vorher noch zwei andere Softwarepakete zu installieren[77]: Das erste Paket ist die *NVIDIA cuDNN Deep Neural Network* Bibliothek in der Version 7.6.0, die grundlegende Implementationen für Deep Learning Berechnungen mitbringt[78]. Auf dieser Bibliothek aufbauend wurde dann noch die *NVIDIA CUDA API*

[75] Vgl. Loyal 2018, 65 f.
[76] Weiterführend zu diesem Thema: towards data science. What is a GPU and do you need one in Deep Learning?, <https://towardsdatascience.com/what-is-a-gpu-and-do-you-need-one-in-deep-learning-718b9597aa0d> (01.08.2021).
[77] Da Keras ein Teil der TensorFlow Bibliothek ist, war es nötig auch TensorFlow in der Version 1.14.0 auf dem Rechner zu installieren. TensorFlow, <https://www.tensorflow.org> (01.08.2021); Keras. The Python deep learning API, <https://keras.io> (01.08.2021); Ubuntu, <https://ubuntu.com> (01.08.2021).
[78] NVIDIA cuDNN, <https://developer.nvidia.com/cudnn> (01.08.2021).

(Version 10.1.243) installiert. Diese ist dafür zuständig, die erforderlichen Berechnungen auf der GPU des Rechners auszuführen und diese durch die hohe Nebenläufigkeit der einzelnen Rechenschritte möglichst stark zu beschleunigen[79]. Nachdem alle drei grundlegenden Software-Bestandteile eingerichtet waren und ihre Funktion sichergestellt wurde, konnte die Arbeit an dem VGG16 Modell beginnen[80]. Dazu wurde die *VGG16* Netzwerkarchitektur aus *Keras* mit den auf dem *ImageNet* Datensatz vortrainierten Gewichten geladen und mittels den verschiedenen Trainingsmengen weitertrainiert (*Transfer Learning*) (siehe Abschn. 4.1)[81]. Das Training auf der größten Bildmenge (2.3) dauerte mit dem beschriebenen Setup bei einer Länge von sechs Epochen ca. vier Stunden.

4.4 VGG16 Architektur und Hyperparameter

In diesem Abschnitt soll die verwendete Deep Learning Netzarchitektur VGG16 und die zugehörigen Hyperparameter erläutert werden.

a) VGG16 Architektur

Die in dieser Arbeit verwendete VGG16 Netzarchitektur wurde in der Masterarbeit von A. Loyal gegen die AlexNet Architektur zur Erkennung von Kaiserporträts auf Münzen der OCRE Datenbank getestet. Dabei konnte sich VGG16 mit deutlichem Abstand bei sämtlichen ausgewerteten Metriken durchsetzen[82]. Daher wurde diese Architektur auch als Grundlage zur Lösung des festgestellten Problems der Einbeziehung der Münzlegende in die Identifizierung des kaiserlichen Porträts verwendet.

Die VGG16 Architektur wurde von der *Visual Geometry Group (VGG)* der Universität Oxford entwickelt[83]. Das Training der zugehörigen Netzwerkgewichte wurde auf dem *ImageNet* Bilddatensatz, der ca. 15 Mio. annotierte Bilder aus ca. 22.000 Kategorien enthält, ausgeführt[84]. Abb. 16 zeigt den Aufbau der Netzarchitektur[85]. Es gibt dreizehn Convolution und fünf Max Pooling Schichten. Auf zwei oder drei Convolution- folgt jeweils eine Max Pooling Schicht.

[79] CUDA Zone, <https://developer.nvidia.com/cuda-zone> (01.08.2021).
[80] Neben den drei oben erwähnten Bestandteilen ist ein zur installierten GPU passender Treiber nötig. Der gesamte Installationsvorgang (mit andern Versionen der Software) mit den benötigten Kommandos kann hier nachgesehen werden: TensorFlow. GPU support, <https://www.tensorflow.org/install/gpu> (01.08.2021).
[81] ImageNet, <https://image-net.org> (01.08.2021).
[82] Außerdem ergab das Experiment, dass die besten Werte bei einer unbegrenzten Anzahl von Bildern pro Klasse erreicht werden: Loyal 2018, 45 f. mit Tab. 4.5.
[83] Simonyan – Zisserman 2014.
[84] ImageNet, <https://image-net.org> (01.08.2021).
[85] Für einen detaillierten Aufbau des Netzwerks siehe: Loyal 2018, 87 Abb. A.2.

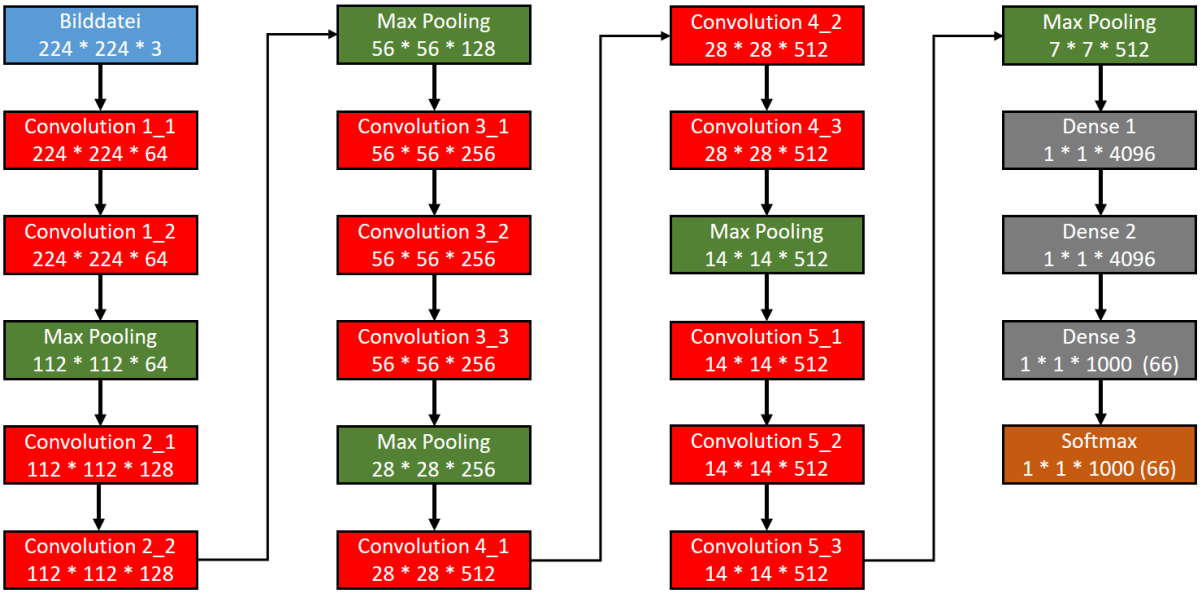

*Abb. 16: Aufbau des Standard-VGG16 Netzes; Die Zahlen geben die Größe der Ausgabe an: Höhe * Breite * Anzahl Kanäle . Der einzige Unterschied zu dem anderen hier verwendeten Aufbau ist in Schicht „Dense 3“ zu sehen: 1*1*66 Kanäle statt 1*1*1000. Die Softmax Aktivierungsfunktion von „Dense 3“ ist hier als eigene Schicht aufgeführt.*

Auf das Convolution/Max Segment werden dann drei vollständig verbundene Schichten und eine Softmax Schicht, die den Abschluss des Netzes bildet und die Klassen mit ihren Wahrscheinlichkeiten ausgibt, angehängt. Der Zahl „16“ im Namen der Netzarchitektur bezieht sich auf die Anzahl der mit Gewichten versehenen Schichten. Dies sind die dreizehn Convolution und die drei vollständig verbundenen Schichten. Jede Convolution Schicht besitzt einen 3*3 Pixel großen Filter-Kernel mit Schrittweite 1 und einem Padding von 1. Die Anzahl der Kanäle (Filter) steigt mit der Tiefe der Convolution Schicht an[86]. Als Aktivierungsfunktion kommt hier die *Rectified Linear Unit* (ReLU) zum Einsatz, die bei negativen Werten eine null ausgibt und ansonsten den Wert selber. Für das Max Pooling wird jeweils ein 2*2 Fenster mit Schrittweite 2 verwendet. Die ersten zwei der vollständig verbundenen Schichten besitzen eine Größe von 4096 Kanälen und verwenden ebenfalls eine ReLU Aktivierungsfunktion. Dagegen besteht die dritte vollständig verbundene Schicht aus 66 Kanälen analog zu den 66 trainierten Klassen. Dort sitzt auch die Softmax Aktivierungsfunktion zur Ausgabe der Wahrscheinlichkeiten. Es können aber bis zu 1000 Klassen mit dieser Architektur trainiert werden. Ob die originale Netzstruktur mit einer 1000 Klassen Softmax Ausgabe ebenfalls benutzt werden kann, wird in Experiment 1 (siehe Abschn. 5.1) geprüft. Das Netzwerk erwartet

[86] Lediglich Convlution Schicht 5 verwendet genauso viele Filter wie Schicht 4.

als Eingabe 224 * 224 Pixel große RGB-Bilddateien[87]. Der Convolution/Max Pooling Abschnitt des Netzwerkes mit den bereits vortrainierten Gewichten kann aus der Keras API geladen werden und wird somit als ein Layer des neuen Netzes behandelt (siehe Abb. 17). Wichtig ist es den VGG16 Layer mit dem Attribut „*layer.trainable=True*“ in die Lage zu versetzen seine Gewichte während des Trainings anzupassen, da sonst nur die Gewichte der vollständig verbundenen Schichten trainiert werden[88]. Diese vollständig verbundenen Schichten wurden für weitere Experimente neu erstellt, da sie lediglich die hier trainierten 66 Klassen ausgeben sollen (siehe Abb. 27 Endstruktur 1)[89]. Anschließend wird eine „Flatten“ Schicht als Verbindungsebene zu den vollständig verbundenen Schichten („dense layer“) dazwischengeschaltet. Diese Schicht dient dazu die zweidimensionale Ausgabe der letzten Max Pooling Schicht, um eine Dimension zu reduzieren, damit sie als Eingabevektor der ersten „dense“ Schicht dienen kann[90]. Darauffolgend werden zwei vollständig verbundene Schichten mit jeweils 4096 Kanälen angefügt. Den Abschluss des Netzes bildet eine vollständig verbundene Schicht mit einer „Softmax“ Funktion zur Ausgabe der Wahrscheinlichkeiten der 66 Klassen. Nach der Festlegung der Netzstruktur wird das Netz kompiliert (siehe Abb. 17).

In Experiment 2 (siehe Abschn. 5.2 mit Abb. 27 Endstruktur 2) wurde eine andere Endstruktur des Netzes getestet: Um die Convolution Schichten mit einer einzigen, vollständig verbundenen Schicht zu verknüpfen, wird eine Global Average Pooling Schicht dazwischengeschaltet, die ebenfalls die zweidimensionalen Ausgaben der „VGG16“ Schicht um eine Dimension für die „dense“ Schicht reduziert. Eine Softmax Funktion dient wiederrum als Abschluss. Vor dem Training muss die gesamte Struktur wiederrum kompiliert werden (siehe Abb. 18)[91].

---

[87] Die Bilder müssen bei abweichender Größe skaliert werden. Neurohive. VGG16 – Convolutional Network for Classification and Detection, <https://neurohive.io/en/popular-networks/vgg16> (1.8.2021); Loyal 2018, 41–44.

[88] Keras. Transfer learning & fine tuning, <https://keras.io/guides/transfer_learning> (01.08.2021). Das Setzten des Parameters auf “False” bewirkt, dass kein ausreichender Lernfortschritt erzielt werden kann, da die Gewichte „eingefroren“ werden. Die Convolutional Schichten sind somit weiterhin auf die 1000 Klassen des ImageNet Datensatzes ausgerichtet. Ein mit eingefrorenen Convolution/Max Pooling Layern trainiertes Netz bleibt bei unter 10% Test- Accuracy und einem Loss Wert von über 4 stehen. Damit erweist sich dieses Modell als untauglich zur Erkennung der Kaiserporträts.

[89] In Experiment 1 wurde probiert, ob das vollständige Netz mit der 1000 Klassen Ausgabe aus Keras geladen werden kann, siehe Abschn. 5.1. In den weiteren Experimenten wird dann die Struktur mit dem neuerstellten Endsegment benutzt: siehe Abschn. 5.2 und 5.3.

[90] Diese „Flatten“ Schicht kommt in der originalen Netzbeschreibung von VGG16 nicht vor, sie wird aber auch von der VGG16 Implementierung der Endsegmente in Keras verwendet, siehe: GitHub. Keras-applications/vgg16.py at master, <https://github.com/keras-team/keras-applications/blob/master/keras_applications/vgg16.py#L177> (01.08.2021).

[91] Die Implementation in Jupyter Notebook wurde nach der folgenden Quelle durchgeführt: Knowledge Transfer. How to use VGG model in TensorFlow Keras, <https://androidkt.com/how-to-use-vgg-model-in-tensorflow-keras> (01.08.2021); Im Gegensatz zu der dort beschriebenen Vorgehensweise den Parameter *„VGG16_MODEL.trainable“* auf *„False“* zu setzen, wurde er in der hier angewandten Umsetzung auf *„True“* gesetzt, da sonst so gut wie kein Lernerfolg während des Trainings erzielt werden kann. Die Convolutional/ Max Pooling Schichten müssen während des Trainings aktualisiert werden können, damit das Model die Erkennung der Kaiserporträts erlernen kann. Siehe auch Fußnote 87.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 4096)              102764544
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 66)                270402
=================================================================
Total params: 134,530,946
Trainable params: 134,530,946
Non-trainable params: 0
```

Abb. 17: Ausgabe des „model.summary()“ Befehls nach Kompilierung des VGG16 Modells mit Endstruktur 1[92].

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 66)                33858
=================================================================
Total params: 14,748,546
Trainable params: 14,748,546
Non-trainable params: 0
```

Abb. 18: Ausgabe des „model.summary()“ Befehls nach Kompilierung des VGG16 Modells mit Endstruktur 2.

b) Hyperparameter

Die wichtigen Hyperparameter zum Trainieren des VGG16 Netzes sollen hier vorgestellt werden[93]. Diese bestehen im Einzelnen aus:

- Obergrenze der Bilder pro Klasse
- Epochenanzahl,
- Batchgröße
- Größe des *Shuffle Buffers*
- Kostenfunktion
- Lernverfahren und Lernrate

Obergrenze der Bilder pro Klasse: Neben den klassischen ML Hyperparametern wurde die Obergrenze der Bilder pro Klasse von A. Loyal in einem Experiment getestet. Die zentrale Frage dabei war es, ob eine mögliche Überanpassung bei einer unbegrenzten Anzahl der Bilder pro Klasse auftreten kann. Diese Frage rührt aus den großen Unterschieden in der Anzahl der klassenspezifischen Bilder. Beispielsweise ist bei den Trainingsbildern der Bildmenge 1.1 die am stärksten repräsentierte Klasse „Konstantin I.“ mit 1835 Bildern. Dagegen besitzt „Phillipus

[92] TensorFlow Core v.2.6.0, tf.keras.Model, <https://www.tensorflow.org/api_docs/python/tf/keras/Model> (01.08.2021).
[93] Einige der benutzten Werte wurden aus der Masterarbeit von A. Loyal übernommen. Loyal 2018

II.“ als „kleinste“ Klasse lediglich 56 Münzfotos[94]. Um die Modelle auf eine mögliche Überanpassung aufgrund dieses Ungleichgewichts zu prüfen, hat A. Loyal für mehrere Begrenzungsstufen (max. 200, max. 300, max. 1000 Bilder und keine Begrenzung) Confusion Matrizen erstellt, die eine Überanpassung durch hohe Werte in einer der Klassen zugeordneten Spalte anzeigen können. Sie konnte mithilfe dieser Matrizen zeigen, dass es sowohl mit als auch ohne Begrenzung zu keiner Überanpassung kommt. Das Modell ohne Begrenzung zeigt sogar bessere Werte in den Metriken und somit auch weniger Masse abseits der Diagonalen der Confusion Matrix[95]. Sie kommt schließlich zu dem Fazit, dass eine Begrenzung der Bildanzahl pro Klasse für die Porträterkennung nicht nötig ist. Daher wurde bei den Modellen in dieser Arbeit ebenfalls keine Begrenzung vorgenommen. Da in Datensatz 2 die Bildmenge insgesamt deutlich gesteigert wurde, könnte eine Begrenzung dazu führen, dass der Vorteil der größeren Zahl der Trainingsdaten teilweise wieder zunichtegemacht wird. Somit wird eine mögliche Überanpassung durch zu viele Bilder pro Klasse noch einmal überprüft (siehe Abschn. 5.2)[96].

Epochenanzahl: Der Hyperparameter der *Anzahl der Epochen* gibt an wie oft der gesamte Trainingsdatensatz durch das Netzwerk während des Trainingsvorgangs propagiert werden soll. Eine Epoche besteht beispielsweise aus dem Durchlauf von allen ca. 21.000 Bildern der Trainingsmenge von Bildmenge 1.1. Dieser Parameter muss so gewählt werden, dass eine Über bzw. Unteranpassung (*Over- bzw. Underfitting*) verhindert wird (siehe Abschn. 3.1). A. Loyal hat dazu in ihrer Masterarbeit eine Epochenanzahl von fünfzehn während der Experimente zur Ermittlung der besten Optimierungsmethode gewählt. Letztendlich hat sie sich für das Training der finalen Modelle für eine Epochenanzahl von zehn entschieden, da man an der *„Accuracy“* und der *„Loss“* Metrik sehen konnte, dass ein über Epoche zehn hinausgehendes Training keine weitere Verbesserung bringt[97]. Daher wird diese Anzahl für die Trainingsvorgänge in dieser

[94] Somit überschreiten auch alle Klassen die von A. Loyal gewählte Untergrenze von 20 Bildern pro Klasse. Loyal 2018, 34 f.

[95] Loyal 2018, 49 f. mit Abb. 4.7, A.4 f. Eine leichte Überanpassung kann lediglich bei den Klassen „Valentinian“ und „Valens“ festgestellt werden. Dies kann aber dadurch begründet werden, dass sie Brüder sind und daher eine sehr ähnliche Ikonographie der Porträts aufweisen. Dies ist auch bei der Verwechslung von „Valentinian I.“ – „Gratian“ und „Gratian“ – „Valentinian II.“, die jeweils in einem Vater-Sohn Verhältnis stehen zu beobachten. Insgesamt lässt sich beispielsweise auf der Confusion Matrix des Modells mit 1000 Bilder Begrenzung eine Tendenz zur Verwechslung der Kaiser der 2. Hälfte des 4. Jhs. n. Chr. und der ersten Hälfte des 5. Jhs. n. Chr. beobachten. Dies kann nicht mit der Menge der Bilder begründet werden, da diese bei keiner der hier zu diesem Zeitraum gehörigen Klassen die Marke von 200 übersteigt. Insgesamt betrachtet kann man sagen, dass die zu dieser Zeit auf den Münzen vorherrschende Kaiserikonographie der Spätantike (beispielsweise die betont großen Augen) mit ihrer grundsätzlichen Ähnlichkeit der einzelnen Kaiser den Modellen Probleme bereitet. Vgl. Loyal 2018, 51. Zum Kaiserporträt der Spätantike siehe: Peschlow 1983.

[96] Loyal 2018, 52; A. Loyal empfiehlt bei einer Ausweitung der Trainingsmenge die Begrenzung der Bilder pro Klasse zu übernehmen.

[97] Lediglich die Optimierungsmethode „RMSProp“ zeigt nach Epoche zehn noch einen flachen Anstieg der *„Accuracy“*. Da diese Methode aber selbst nach fünfzehn Epochen nur Werte um die 52 % erreicht, spielt sie bei der Auswahl der besten Optimierungsmethode keine Rolle. Loyal 2018, 59 Abb. 4.12.

Arbeit übernommen[98]. Das Problem der Unter. bzw. Überanpassung wird mit der Epochenzahl in den Experimenten dieser Arbeit überprüft. Hat man den Modellen genügend Trainingsdurchläufe zukommen lassen, um den generellen Trend der Daten abzudecken ohne dass sich ein Modell zu sehr den Bildern der Trainingsmenge anpasst. Die Vermeidung von Unteranpassung kann an hohen Werten der *„Accuracy"* abgelesen werden. Das Problem der Überanpassung tritt dagegen nicht auf, wenn ein niedrige Werte bei der Differenz zwischen Trainings- und Testwerten der *„Loss"* Metrik gegeben sind und es nicht zu einem Absinken der Test-*„Accuracy"* während des Trainings kommt (siehe Abschn. 5.2)[99].

Batchgröße: Die *Batchgröße* gibt an wie viele Bilder dem Modell auf einmal durch das Netzwerk propagiert werden. Der Vorteil einer kleineren Batchgröße ist das schnellere Lernvorgang des Modells: Nach jedem Propagieren eines Batches (ein Trainingsschritt) werden die Gewichte des Netzes angepasst (s.u. *Lernverfahren*). Dies bedeutet, dass bei einer kleineren Batchgröße (sog. Minibatch) wesentlich mehr Updateschritte ausgeführt werden als bei einer Größe die in etwa der Trainingsmenge entspricht. Bei einer zu großen Bildermenge pro Batch werden also möglicherweise zu wenige Lernschritte durchgeführt, um ein gutes Ergebnis zu erreichen. Zu kleine Mengen können die Anpassung der Gewichte aber negativ beeinflussen, da die enthaltenen Informationen möglicherweise nicht ausreichend für den Lernvorgang sind. Das bedeutet jedoch auch, dass kleinere Batchgrößen die zum vollständigen Durchlauf des Trainings benötigte Zeit durch häufigere Lernschritte erhöhen. Daneben ist die Batchgröße außerdem durch die Speichermenge der verwendeten Grafikkarte begrenzt. A. Loyal hat in ihrer Masterarbeit eine Batchgröße von sechzehn Bildern benutzt[100]. Bei einer Trainingsmenge von ca. 21.000 Bilder (Datensatz 1) entspricht dies ca. 1300 Iterationen (und somit Lernschritten) pro Epoche. Diese Größe wurde für das Trainieren der Modelle in dieser Arbeit übernommen.

Mit der Speichergröße der verwendeten Grafikkarte hängt auch der Parameter *„Shufflebuffer"*, der von der TensorFlow API eingesetzt wird, zusammen. Der *Shufflebuffer* bezeichnet hier die Menge des vor dem Ausführen einer Epoche gleichzeitig in den Speicher der Grafikkarte geladenen Bilder. Aus diesen Bildern wird der zum Training verwendete Tensor-Datensatz (jeweils für Trainings- und Testmenge) erstellt[101]. Dieser Menge werden während des Trainingsvorgangs die Bilder für die einzelnen Batches per Zufall entnommen. Die Zufälligkeit der Bilder in den Batches ist für ein nicht von einem Bias beeinflusstes Training extrem wichtig,

[98] Loyal 2018, 44, 58–61.
[99] Der Unterschied zwischen den „Loss" Werten für Training- und Testdatensätze beträgt um die 0,3. Es gibt lediglich einen Ausreißer: Bei Datensatz 2 Bildmenge 1 betrug die Differenz zwischen beiden Werten etwas mehr als 0,5. Siehe Abb. 34.
[100] Loyal 2018, 44 f.; Goodfellow u. a. 2016, 274–279.
[101] TensorFlow. tf.data.Dataset <https://www.tensorflow.org/api_docs/python/tf/data/Dataset> (02.08.2021).

da sonst Bilder derselben Klasse mehrere Batches füllen und somit auch mehrere Lernschritte hintereinander beeinflussen würden[102]. Sobald ein Bild aus dem Datensatz entnommen wurde, wird ein Bild aus der Restmenge hinzugefügt sofern der Gesamtdatensatz größer als der *Shufflebuffer* ist. Da der Speicher der hier verwendeten Grafikkarte (GTX 1080 mit 8 GB RAM) nicht ausreicht, um sämtliche Bilder der einzelnen Datensätze aufzunehmen, wurde die Größe auf 12.000 Bilder festgelegt. Ein größerer Buffer erhöht das Risiko, dass Probleme wie ein voller Grafikspeicher auftreten und somit den Trainingsvorgang unterbrechen könnten. Um sicherzugehen, dass auch die Bilder in dem kleineren Buffer ausreichend durchmischt sind, wird vor dem Erstellen des TensorFlow Datensatzes die Liste der Dateipfade aller verwendeten Bilder mit der *„shuffle()"* des Python *„random"* Moduls noch einmal randomisiert[103].

Kostenfunktion: Als *Kostenfunktion* (*Loss*), die die Distanz zwischen den vorhergesagten und den tatsächlichen Klassen misst und somit ein Maß für die Güte des trainierten Modells ist, wird hier die „*Sparse Categorial Crossentropy"* (SCC) verwendet. Der Term „Categorial" bezieht sich hier auf die Fähigkeit dieser Methode die Güte von Modellen mit mehreren Klassen zu messen[104]. Da ML Modelle nicht mit den Klassenlabeln direkt arbeiten können, ist es notwendig diesen eine Zahl zuzuweisen. Die spezielle Methode des „*Sparse Categorial Crossentropy*" unterscheidet sich von der „*Categorial Crossentropy"* dadurch, dass die Klassen hier als Integer-Zahlen kodiert werden können, während das andere Verfahren eine „*One-Hot-Kodierung*" benutzt[105]. Beispielsweise wird die Klasse „Augustus" durch die Zahl „2" repräsentiert. Insgesamt gesehen misst die SCC die Differenz zwischen zwei Wahrscheinlichkeitsverteilungen, was hier der Verteilung der Klassen in der Vorhersage des Modells und der Verteilung der tatsächlichen Klassen entspricht[106]. Das Ziel der Kostenfunktion in Zusammenarbeit mit dem Lernverfahren ist es, eine möglichst hohe Wahrscheinlichkeit bei der tatsächlichen Klasse in der vorhergesagten Wahrscheinlichkeitsverteilung zu erreichen[107].

---

[102] Goodfellow u. a. 2016, 277.
[103] Python 3.9.6 documentation. random — Generate pseudo-random numbers<https://docs.python.org/3/library/random.html> (02.08.2021).
[104] Als Gegenstück für binäre Klassifikatoren wird die „*Binary Crossentropy*" verwendet.
[105] Die „One-Hot-Kodierung" stellt die Klassen als Reihe von binären Zahlen dar. Jeder Stelle in dem Kode ist eine Klasse zugeordnet, die zugewiesen Klasse wird durch eine „1" an der entsprechenden Stelle markiert.
[106] Die tatsächliche Verteilung einer Klasse weißt der Zielklasse die Wahrscheinlichkeit „1" den restlichen Klassen die Wahrscheinlichkeit „0" zu.
[107] Goodfellow u. a. 2016, 72 f.; Peltarion Platform. Categorial crossentropy loss function, <https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy> (02.08.2021). Das von A. Loyal eingesetzte NVIDIA Digits hat in der Weboberfläche zur Erstellung einer Architektur keine Möglichkeit zur Auswahl einer Kostenfunktion. Daher wird wahrscheinlich die Standardeinstellung eines „Softmax with Loss Layer" von Caffe eingesetzt worden sein. Diese spezielle Schicht nutzt die Softmax Schicht des VGG16 Netzwerkes, um eine Multinomiale logistische

Lernverfahren und Lernrate: Das *Lernverfahren* (auch *Optimierungsmethode* genannt) ist dafür zuständig die Gewichte der einzelnen Neuronen eines Netzwerkes nach jedem Trainingsschritt (s.o.: *Batch*) anzupassen. Erst diese Methode ermöglicht es einem Modell aus den vorhandenen Trainingsdaten Schlüsse zu ziehen und somit hinzuzulernen. Das Ziel ist es hierbei, dass ein genereller Trend aus den Daten gelernt wird. In ihrer Masterarbeit hat A. Loyal verschiedene von NVIDIA Digits zur Verfügung gestellte Lernverfahren in einem Experiment getestet. Sie kam zu dem Ergebnis, dass die „*Stochastic Gradient Descent*" (SGD) Methode die besten Ergebnisse bei der Porträterkennung liefert[108]. Dieses Verfahren basiert auf der Gradientenabstiegsmethode (oder auch Fehlerrückführung, siehe auch Abschn. 3.2) und nutzt den durchschnittlichen Gradient der Minibatches um den vollständigen Gradienten zu schätzen. Der wichtigste Parameter dieser Methode ist die sog. Lernrate. Sie bestimmt wie stark sich die Gewichte des Netzes in einem Lernschritt in Richtung des negativen Gradienten der Kostenfunktion ändern. Somit ergibt sich als Formel für die Iteration $\tau + 1$ mit dem SGD Verfahren:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n$$

Formel 5: SGD Verfahren[109]

Die Gewichte ($w^{(\tau + 1)}$) der neuen Iteration $\tau + 1$ werden aus den alten Gewichten ($w^{(\tau)}$) der Iteration τ minus der Lernrate (η) mal dem Gradienten der Kostenfunktion (∇E_n) gebildet. Dieses Verfahren bietet außerdem die Möglichkeit eine fixierte Lernrate einzusetzen oder diese schrittweise abzusenken. Bei einer zu kleinen fixierten Lernrate kann es passieren, dass man mit dem Gradientenabstieg in einem lokalen Minimum steckenbleibt. Auch wenn dies nicht eintritt, werden doch sehr viele Lernschritte nötig, um ein gutes Ergebnis zu erzielen. Wenn man jedoch eine zu große fixierte Rate einsetzt, ist es möglich, dass man zu große Sprünge innerhalb des Verlaufs der Kostenfunktion macht und das Verfahren somit niemals konvergiert[110]. Daher empfiehlt es sich mit einer Basis-Lernrate das Training zu beginnen und diese dann schrittweise abzusenken[111]. Das Festlegen der Lernrate wurde von A. Loyal wiederrum in einem Experiment getestet. Sie testete eine feste, eine mit stufenweiser

Regression zu berechnen. Loyal 2018, 69 Abb. 5.6; Caffe. Softmax with Loss Layer, <http://caffe.berkeleyvision.org/tutorial/layers/softmaxwithloss.html> (02.08.2021).

[108] Das „*Nesterov's Accelerated Gradient*" (*NAG*) Verfahren lieferte in dem Experiment ähnlich gute Ergebnisse. Jedoch entschied sich A. Loyal dann dafür SGD als einzige Methode weiterzuverwenden. Das finale Modell „VGG16_layers_Ziel1b_min100_max1000_final_SGD_LR0.001", welches in der Porträterkennung die besten Ergebnisse lieferte das in der Bachelor Arbeit des Autors ausgewertet wurde, basiert auf dem SGD Lernverfahren. Loyal 2018, 58–61; Gampe 2019, 12 f. 19 f. 27–37.

[109] Formel nach: Bishop 2006, 240 Formel 5.41.

[110] Für Beispiele zu dieser Problematik siehe: Loyal 2018, 47 Abb. 4.6.

[111] Loyal 2018 46; Goodfellow u. a. 2016, 290 f.

Verringerung und eine sigmoid verringerte Lernrate. Dabei konnte das VGG16 Modell mit einem Startwert von „0,001“ und stufenweiser Verringerung die besten Ergebnisse erbringen. Daher wurde für das Training der Modelle hier der Ansatz von SGD mit einem Startwert von „0,001“ und stufenweiser Verringerung um eine Zehnerpotenz übernommen (siehe Abb. 28). In Experiment 2 (Siehe Abschn. 5.2) werden verschiedene stufenweise Anpassungen der Lernraten darauf überprüft, ob sie zu einer möglichen Überanpassung führen können. Dabei werden jeweils zwei Abstufungen gemacht: Ab einer bestimmten Epoche greift die erste Verringerung auf „0,0001“ und die letzten Epochen werden mit dem Wert „0,00001“ trainiert. A. Loyal benutzte während ihrer Tests eine Epochenanzahl von fünfzehn. Da sie aber den Lernratenverlauf ihres finalen Trainings mit lediglich zehn Epochen nicht angegeben hat, mussten hier mehrere Verringerungsraten getestet werden. Das Ergebnis dieses Experiment war, dass eine Lernrate von wenigstens sechs Epochen mit zwei Abstufungen völlig ausreichend ist, um die besten Ergebnisse mit minimaler Überanpassung bei der „*Loss*“ Metrik zu erreichen (siehe Abb. 21, Abb. 28 Links und Rechts).

4.5 Implementierung der Auswertungsfunktionen

Dieser Abschnitt beschäftigt sich mit dem Ausführen der in dieser Arbeit trainierten Modelle und der Implementierung der Methoden, mit welchen diese Modelle im nächsten Kapitel untersucht werden sollen (Siehe Kap. 5).

a) Modellausführung

Da die TensorFlow Keras API vollständig in Python ausführbar ist, kann das Trainieren und die Ausführung eines Modelles in einem Jupyter Notebook erfolgen. Ein bereits trainiertes Modell wird zusammen mit seinem Aufbau und den Gewichten mit der „*models.save_model()*“ Methode in einer Datei abgespeichert und kann anschließend mit „*models.load_model()*“ geladen werden[112]. Weitere Dateien werden nicht gebraucht, da sämtliche Klassen in numerischer Form, der Netzaufbau und die Gewichte enthalten sind. Zur Vorhersage von Klassen auf neuen Bildern wird lediglich das Verzeichnis mit den Bildern und ein Python *Dictionary* zur Übersetzung der numerischen Repräsentation der Klassen in die Namen der Kaiser benötigt. Dieses *Dictionary* kann aus der Angabe eines Ordners mit den nach den Kaisern benannten Unterordnern erstellt werden (z.B. aus dem Ordner der Trainingsmenge). Um ein Bild mittels der „*model.predict()*“ Methode (auf dem geladenen model-Objekt) durch

[112] TensorFlow Core. Save and load Keras models, <https://www.tensorflow.org/guide/keras/save_and_serialize> (02.08.2021).

das Netz zu schicken, muss das Porträt vorher noch auf die benötigte 224 * 224 Pixel Größe gebracht werden. Dazu wird das Bild mit „*keras.preprocessing.image.load_img()*“ geladen und auf die benötige Größe skaliert, anschließend wird es mit „*keras.preprocessing.image.img_to_array()*“ in ein entsprechend großes Array geladen[113]. Mit der Numpy Methode „*expand_dims()*“ können noch Batches von Bildern zur Propagation durch das Netz erstellt werden[114]. Für die Auswertung wurde eine Batchgröße von eins gewählt. Im letzten Schritt wird das Array noch mit der „*keras.applications.xception.preprocess_input()*“ Funktion bearbeitet, welche die Kompatibilität des Arrays mit der benötigten Eingabe des Netzes sicherstellt (siehe auch Abschn. 5.1)[115]. Die Vorhersage des Netzes zu dem propagierten Bild wird als Array mit den Wahrscheinlichkeiten der einzelnen Klassen ausgegeben. Mit der NumPy „*argmax()*“ Funktion kann jetzt der Indizes der wahrscheinlichsten Klasse bestimmt und anschließend mit dem erstellten *Dictionary* in den zugehörigen Kaisernamen umgewandelt werden[116]. Um die Top-5 der wahrscheinlichsten Klassen zu finden, wird die Funktion „*numpy.argsort()*“ angewandt, die die Indizes der n wahrscheinlichsten Klassen nach ihrer Wahrscheinlichkeit sortiert zurückgibt[117]. Diese werden ebenfalls mit dem *Dictionary* in die Klarnamen der Kaiser umgewandelt.

b) Auswertungsmethoden

Metriken: Zur Evaluation der neu trainierten Modelle der verschiedenen Experimente (siehe Kap. 5) wurden mehrere Metriken und Tools verwendet. Als erstes werden die in Abschn. 3.3 aufgeführten Metriken von ML Modellen implementiert. Zur Ausführung werden sämtliche Bilder der verschiedenen Testmengen erneut durch das fertige Netz propagiert und die *Groundtruth* der Bilder mit den Top-1 und Top-5 Wahrscheinlichkeiten der Vorhersage verglichen. Die Ergebnisse werden dann zur Berechnung der Metriken mithilfe der Funktion „*sklearn.metrics.classification_report()*“ verwendet[118].

---

[113] TensorFlow Core v.2.6.0. tf.keras.utils.load_img, <https://www.tensorflow.org/api_docs/python/tf/keras/utils/load_img> (02.08.2021); TensorFlow Core v.2.6.0. tf.keras.utils.img_to_array, <https://www.tensorflow.org/api_docs/python/tf/keras/utils/img_to_array> (02.08.2021).

[114] NumPy v.1.21 Manual. numpy.expand_dims, <https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html> (02.08.2021).

[115] Falls sich das Bildformat und das vom Netz benötige Eingabeformat nicht unterscheiden führt diese Methode keine Konvertierung aus. TensorFlow. tf.keras.applications.xception.preprocess_input, <https://www.tensorflow.org/api_docs/python/tf/keras/applications/xception/preprocess_input> (02.08.2021).

[116] NumPy v.1.21 Manual. numpy.argmax, <https://numpy.org/doc/stable/reference/generated/numpy.argmax.html> (02.08.2021).

[117] NumPy v.1.21 Manual. numpy.argsort, <https://numpy.org/doc/stable/reference/generated/numpy.argsort.html> (02.08.2021).

[118] Loyal 2018, 72; scikit-learn 0.24.2 documentation. sklearn.metrics.classification_report, <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html> (03.08.2021).

Grad-CAM: Um das Verhalten des trainierten Modelles, das ohne ein erläuterndes Verfahren eine „Black Box" darstellt, besser erklären zu können, wurde die *Grad-CAM* Methode in dieser Arbeit verwendet (siehe Abschn. 3.5). Dazu wurde eine bereits vorhandene Implementierung von der *Keras* Webseite verwendet und für das hier benutzte VGG16 Modell angepasst[119]. Dazu mussten die Namen der einzelnen Schichten des Netzes im Code angepasst werden, da diese Implementation für ein anderes Netz geschrieben wurde. Insbesondere werden die Convolution Schichten der von Keras geladenen VGG16 Implementation als eine eigene Schicht gesehen (sie Abschn. 4.4.a mit Abb. 17). Da die Grad-CAM Methode Zugriff auf die Schicht 5_3 als letzte Convolution Schicht im VGG16 Netz braucht (siehe Abb. 16), war es nötig den Zugriff auf diese mit der „*get_layer('vgg16')*" Funktion der „Model" Klasse zu ermöglichen[120]. Jetzt kann eine *Heatmap* mit den stärksten Aktivierungsbereichen des Bildes erstellt werden. Diese Heatmap, die das Format 14 * 14 Pixel (entsprechend der Ausgabe der Convolution 5_3 Schicht) hat, wird im letzten Schritt im Format angepasst und über das zugrundeliegende Porträt oder die Münze gelegt. Danach wird das neue mit der Heatmap versehene Bild zur weiteren Auswertung abgespeichert (siehe Abb. 8).

Confusion Matrix: Um sich einen Überblick über die verschiedenen Klassen und eine mögliche Überanpassung (*Overfitting*) auf einzelne Klassen zu verschaffen werden Confusion Matrizen für jedes Modell auf den jeweiligen Testmengen erstellt. Wie auch bei der Arbeit von A. Loyal wird dazu eine farbige Darstellung der einzelnen Punkte gewählt. Die Skala der Farben rangiert dabei von Dunkelblau für sehr niedrige Werte bis tiefrot für sehr hohe Werte (siehe Abb. 49). Implementiert wurde dieses Verfahren mit zwei Funktionen. Die „*sklearn.metrics.confusion.matrix()*" erstellt mittels der Vorhersagen und der Groundtruth der einzelnen Bilder eine zahlenbasierte Matrix[121]. Diese wird dann mittels der Funktion „*heatmap()*" des „*seaborn*" Moduls in eine farbenbasierte Confusion Matrix umgewandelt[122]. Die erstellte Matrix wird zur Auswertung der einzelnen Modelle ebenfalls abgespeichert.

[119] Keras. Grad-CAM class activation visualization, <https://keras.io/examples/vision/grad_cam/> (03.08.2021); Mittlerweile ist diese Implementation aktualisiert worden und entspricht nicht mehr der hier verwendeten Version. Diese ältere Version kann hier gefunden werden: GitHub. Keras.io / examples / vision / grad-cam.py, <https://github.com/keras-team/keras-io/commit/8320a6c0a63cd6debe960689c182cee491193fa9#diff5ffd0b9f9b5d9d4dacd6723b201147c03d071d76755b1fa33e87ba6044b8221d> (03.08.2021).
[120] Keras. The Model class, <https://keras.io/api/models/model> (03.08.2021).
[121] scikit-learn 0.24.2 documentation. sklearn.metrics.confusion_matrix, <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html> (03.08.2021).
[122] seaborn 0.11.2 documentation. seaborn.heatmap, <https://seaborn.pydata.org/generated/seaborn.heatmap.html> (03.08.2021).

5. Experimente und Evaluation der Ergebnisse

Das nachfolgende Kapitel legt die Ergebnisse der Experimente mit dem VGG16 CNN Modell zur Erkennung von römischen Kaisern auf antiken Münzen der OCRE und CN Datenbank vor. In jedem Experimentabschnitt wird jeweils die Trainings- und Testmenge vorgestellt und auf die Durchführung eingegangen. Die anschließende Auswertung der einzelnen Experimente beinhaltet einen Blick auf die Metriken und eine genauere Prüfung der Ergebnisse der einzelnen Testmengen. Abschließend sollen die Möglichkeiten der Verwendung und die Probleme der Modelle diskutiert werden. In Experiment 1 wird die originale VGG16 Endstruktur mit der 1000 Klassen Ausgabe auf seine Tauglichkeit für die Porträterkennung getestet. Im zweiten Experiment soll die beste Kombination aus zwei unterschiedlichen Endstrukturen, verschiedener Trainings- und Testmengen, unterschiedlicher Epochenanzahlen und Lernraten gefunden werden. Das letzte Experiment beschäftigt sich dann der Auswertung des Vorhersageverhaltens des besten in den vorherigen Experimenten gefundenen Modells.

5.1 Experiment 1: Test der originalen VGG16 Struktur

Das erste ausgeführte Experiment widmet sich der Endstruktur der verwendeten VGG16 Architektur (siehe Abschn. 4.4.a). Genauer soll hier geprüft werden, ob sich das originale vortrainierte VGG16 Netz mit seiner Ausgabe von 1000 Klassen auf die 66 Kaiserinnen und Kaiser anwenden lässt, oder ob es nötig ist das Endsegment des Netzes bestehend aus den vollständig verbundenen Schichten und der Softmax Ausgabe ohne vortrainierte Gewichte neu zu erstellen und zu trainieren (siehe Abb. 19).

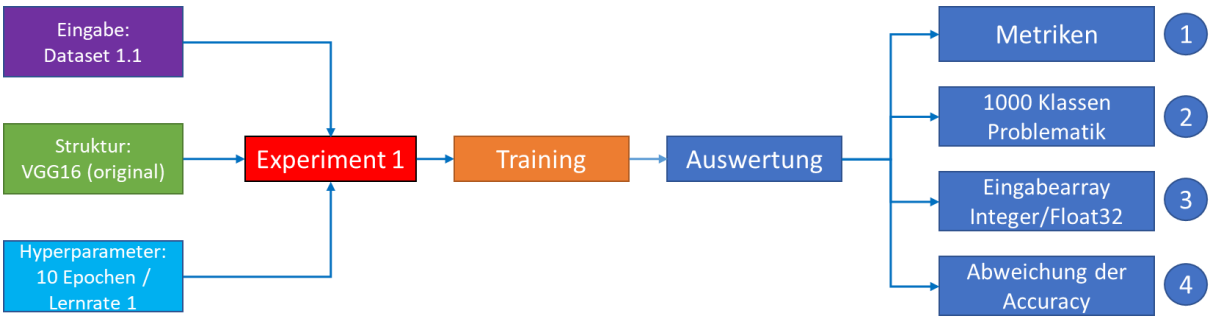

Abb. 19: Ablaufdiagramm des ersten Experiments. Links befinden sich die variablen Eingabeparameter der Experimente und rechts werden die wichtigsten Punkte der Auswertung angezeigt.

Die originale VGG16 Netzarchitektur lässt sich in Keras sehr leicht mit dem Befehl in Listing 1 laden. Das Argument „*include_top=True*“ sorgt dafür, dass die vollständig verbundenen Schichten mit den vortrainierten Gewichten ebenfalls geladen werden.

```
VGG16_MODEL = tf.keras.applications.VGG16(input_shape=IMG_SHAPE, include_top=True, weights='imagenet')
```

Listing 1: Der Befehl zum Laden des vollständigen vortrainierten VGG16 Netzes (Python Code).

Der Befehl „*model.summary()*“ zeigt den kompilierten Aufbau des Netzes (siehe Abb. 20)[123].

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 1000)              138357544
=================================================================
Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0
```

Abb. 20: Ausgabe des model.summary() Befehls bei der originalen VGG16 Netzstruktur.

Das Modell 1 wurde mit zehn Epochen und der stufenweise verringerten Lernrate 1 (siehe Abb. 21 und Abschn. 5.2) trainiert[124]. Die Trainings- und Testmenge besteht aus den ausgeschnittenen Porträts des Bildmenge 1.1, was genau den verwendeten Münzfotos von A. Loyal entspricht, die jedoch zuerst mit der Porträtbereichserkennung ausgeschnitten wurden.

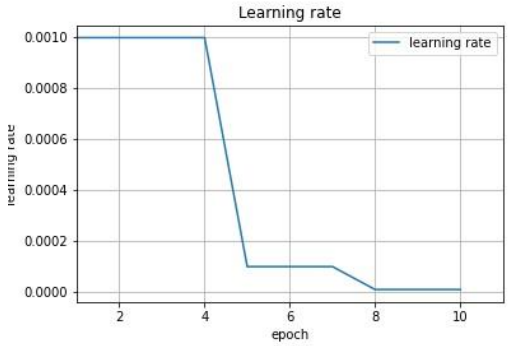

Abb. 21: Die getestete Lernrate 1 mit vier großen, drei mittleren und drei kleinen Lernschritten.

(1) Die Metrik *Accuracy* auf der Testmenge zeigt während der zehn Epochen des Trainings einen durchgehenden Anstieg von 46% in Epoche eins bis 81% in Epoche sechs. Danach steigt das Wert noch sehr langsam bis er in Epoche zehn 81,6% erreicht. Die *Accuracy* der Trainingsmenge steigt bis zur sechsten Epoche sehr stark und danach nur noch relativ langsam an, bis ein Wert von 94,5% erreicht wird (siehe Abb. 22)[125]. Bei der *Loss* Metrik zeigt sich ein ähnliches Bild. Die Testmenge zeigt ein stärkeres Absinken des Loss Wertes bis Epoche sechs. Danach sinkt der Wert langsamer bis zum Endergebnis von 0,58. Der Wert der Trainingsmenge

[123] TensorFlow Core v.2.6.0, tf.keras.Model, <https://www.tensorflow.org/api_docs/python/tf/keras/Model> (03.08.2021). Man kann erkennen, dass Keras das geladene VGG16 Netz hier als einen Layer behandelt. Wenn man Zugriff auf die verschiedenen Schichten innerhalb der VGG16 Schicht nehmen will, muss man dem entsprechenden Befehl mit dem Ausdruck „.get_layer(„VGG16“)“ versehen. Siehe dazu auch Abschn. 4.5.b.
[124] Diese Epochenanzahl wurde von A. Loyals Masterarbeit übernommen. Da sie die finalen Verringerungsstufen der Lernrate nicht angegeben hatte, wurde in Experiment 2 (siehe Abschn. 5.2) noch einmal überprüft, welche Lernrate das beste Ergebnis ermöglicht. Loyal 2018, 51.
[125] Zu beachten ist, dass die Epochenzahl in den Diagrammen von *Accuracy* und *Loss* immer bei null beginnt. Somit ist die nullte Epoche im Diagramm die erste Epoche im Text.

verhält sich mit einem noch stärkeren Absinken analog. In Epoche zehn erreicht er den Wert 0,17. Beide Metriken zeigen keine signifikante Tendenz zur Überanpassung an die Trainingsdaten (*Overfitting*)[126].

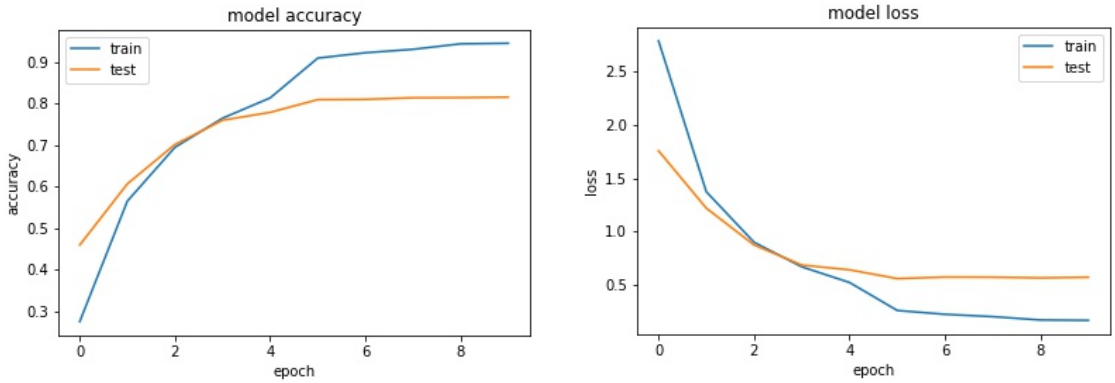

Abb. 22: Links: Die Accuracy von Modell 1 während des Trainings. Rechts: Der Loss desselben Modells während des Trainings.

Laut den untersuchten Metriken ist das Modell 1 bei der Erkennung etwas schlechter als das VGG16 Netz, welche von A. Loyal trainiert worden ist. Ihr bestes Ergebnis lag bei ca. 92% auf der Testmenge, während dieses Modell ca. 82% erreicht[127]. Der Unterschied kann hier möglicherweise durch das Fehlen der Daten der Münzlegende erklärt werden. Wie in der Bachelorarbeit des Autors festgestellt wurde, verlässt sich das von A. Loyal trainierte VGG16 Modell bei seiner Erkennung sehr stark auf die Münzlegende (siehe Abschn. 1.2 und 4.1)[128]. Es ist daher mit einiger Zuverlässigkeit anzunehmen, dass das Fehlen dieser zusätzlichen Identifizierung-Features sich bei der Erkennung negativ auswirken kann.

(2) Das hier untersuchte Modell 1 hat jedoch ein mögliches Problem. Durch die originale VGG16 Struktur bleibt die Ausgabe von 1000 Klassen erhalten (siehe Abb. 24). Lediglich die ersten 66 Klassen wurden auf die hier zu erkennenden 66 Porträtklassen gemappt. Dies kann problematisch werden, da alle Klassen im VGG16 Netz numerisch repräsentiert werden (siehe Abschn. 4.5.a). Das zur Rückübersetzung in die Klarnamen der Kaiser nötige Dictionary hat jedoch nur 66 Einträge. Dies bedeutet, wenn ein Index größer als 65 als Schlüssel zur Suche in diesem Dictionary verwendet wird, kann ein sogenannter „*Key Error*“ ausgelöst werden. Dies könnte dadurch gelöst werden, dass das Dictionary Einträge mit den Schlüsseln 66 bis 999 „*unbekannte Klasse*“ als zugeordneten Werte erhält. Da hier aber eine Erkennung von kaiserlichen Porträts ohne solche „*unbekannte Klassen*“ angestrebt wird, kann dies keine praktikable Lösung sein. Daher bleibt es noch zu testen, ob bei der Ausgabe von Top-1 und

[126] Lediglich der *Loss* der Testmenge steigt von Epochen neun zu zehn von 0,5686 auf 0,5745 minimal an. Eine Überprüfung einer möglichen Überanpassung scheint hier jedoch nicht notwendig zu sein.
[127] Loyal 2018, 76.
[128] Gampe 2019, 31–33.

Top-5 Klassen ein Fall auftritt, der solch einen Key Error auslöst. Um dies zu testen wurde die eine Top-1 und eine Top-5 Ausgabe mit den Klarnamen der Kaiser aus dem Dictionary implementiert und sämtliche Testbilder der Bildmenge 1.1 als Eingabe verwendet. Dieser Testlauf erbrachte mehrere Key Errors als Ergebnis (siehe dazu auch (3)) und zeigte somit, dass dieses Problem durch die oben beschriebene Lösung abgefangen werden muss, die allerdings der Erkennung der Kaiser durch die Verwendung unbekannter Klassen zuwiderläuft.

(3) Ob die 1000 Klassen Ausgabe ein Problem darstellt, hing ebenfalls von der Umwandlung des getesteten Bildes in ein für das CNN lesbares Array ab. Wenn man das Bild mit der NumPy Funktion „*.asarray()*“ in ein mit Integern gefülltes Array umformte, gab das Modell als Ergebnis ein NumPy Array mit „float32“ Einträgen aus, das an den meisten Stellen mit dem Wert „0.0“ gefüllt war (siehe Abb. 23)[129]. Lediglich die wahrscheinlichste Klasse erhielt eine 1.0 als Eintrag[130]. Dieses Verhalten wurde an mehreren Beispielbildern getestet und dabei zeigte es sich, dass das Netzwerk mit Integer-Arrays grundsätzlich Probleme hat. Es kann ebenso vorkommen, dass Klassen, die nicht zu den 66 trainierten Kaiserinnen und Kaisern gehören als Top-1 Ergebnis vorkommen können. Die Top-5 Ausgabe funktioniert aufgrund der vielen 0.0 Werte ebenfalls nicht, da fast immer mindestens ein Ergebnis außerhalb der ersten 66 Indizes liegt. Das ändert sich jedoch grundlegend, wenn die Funktion „*img_to_array()*“ aus dem Keras „*Preprocessing.Image*“ Modul zur Umwandlung des Bildes in ein NumPy Array genutzt wird[131]. Der Unterschied liegt hier in der Repräsentation der Pixelwerte als Float32 Variablen in dem benutzten Array. Das Ergebnisarray enthält jetzt wiederrum Float32 Einträge, die aber für jede Klasse (sehr kleine) Wahrscheinlichkeiten repräsentieren (siehe Abb. 24). Als Funktionstest wurden sämtliche Münzbildern der hier verwendeten Testmenge noch einmal durch das Modell geschickt und es wurde die Top-5 der wahrscheinlichsten Ergebnisse mit Klarnamen erstellt. Während bei einem Integer-Array sofort Key-Errors aufgrund der vielen Null-Werte entstehen (s.o), liegen die Ergebnisse der Float32-Umwandlung weitestgehend innerhalb der 66 trainierten Klassen und es entstehen somit nur vereinzelte Key Errors[132]. Damit wird jedoch trotzdem klar, dass das aus dem Münzbild erzeugte Eingabearray für sämtliche Modelle immer mit Float32 Einträgen gefüllt sein muss, da mit einem Integer Array die oben beschriebenen Probleme weit häufiger auftauchen. Ein Float32 Array gibt dagegen auch

[129] NumPy v.1.21 Manual. numpy.asarray,
<https://numpy.org/doc/stable/reference/generated/numpy.asarray.html> (04.08.2021).
[130] Nicht jedes Porträtbild erhielt eine 100% Wahrscheinlichkeit der Top-1 Klasse als Ergebnis.
[131] Die Funktion ist in der neuesten Version in dem Utils Modul von TensorFlow Keras zu finden: TensorFlow Core v.2.6.0. tf.keras.utils.img_to_array,
<https://www.tensorflow.org/api_docs/python/tf/keras/utils/img_to_array> (04.08.2021).
[132] Insgesamt lagen 34 der ca. 34.340 Top-5 Ergebnisse der 6868 Bilder außerhalb der 66 Klassen. Dies entspricht einem Anteil von ca. 0,1% aller Ergebnisse.

kleinere Wahrscheinlichkeiten für jede Klasse wesentlich genauer an und ermöglicht somit eine Top-5 Ausgabe.

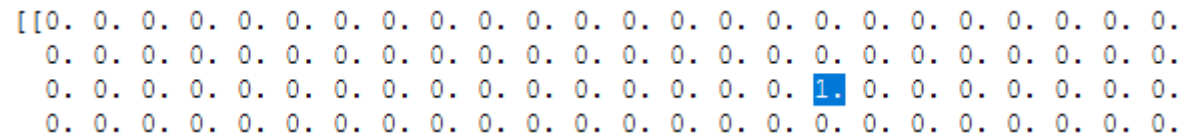

Abb. 23: Ausgabe der ersten 96 Ergebnisse des trainierten Original VGG16 Modells in Jupyter Notebook. Der markierte Eintrag zeigt das wahrscheinlichste Ergebnis mit Indexwert 64 (Vespasian). In diesem Fall wurde ein NumPy ndarray mit Integer Einträgen in das Modell geladen.

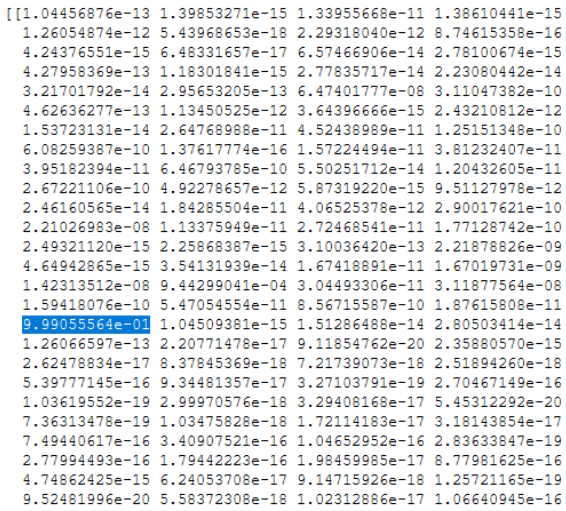

Abb. 24: Ausgabe der ersten 100 Ergebnisse des trainierten Original VGG16 Modells in Jupyter Notebook. Der markierte Eintrag zeigt das wahrscheinlichste Ergebnis mit Indexwert 64 (Vespasian). Hier diente eine PIL Image Instanz mit Float32 Einträgen als Eingabe für das Modell.

(4) Neben dem Problem der Bildumwandlung konnte bei den zugehörigen Testläufen ein weiteres unerwartetes Verhalten beim Training des Modells festgestellt werden. Dies betrifft den Accuracy Wert (siehe Abschn. 3.3), der am Ende des Trainingsvorgangs ausgegeben wird. Bei dem oben beschriebenen erneut durchgeführten Nachtest mit sämtlichen Testbildern wurde ebenfalls der Accuracy Wert berechnet. Am Ende der zehnten Epoche des Trainings betrug der von Keras ausgegebene Wert ca. 81,6% (siehe Abb. 25), während der selbst berechnete Wert in dem Nachtest ca. lediglich 76,7% erreicht (siehe Abb. 26). Aufgrund der Tatsache, dass kein Einblick in den letzten Durchlauf der Testbilder während des Trainings genommen werden kann, also die Anzahl der korrekt erkannten Porträts nicht ausgelesen werden kann und dem Fakt, dass der Accuracy Wert in Keras genauso wie bei dem Nachtest errechnet wird (Anzahl der korrekt erkannten Bilder / Anzahl aller Bilder), kann der bestehende Unterschied von ca. 5% nicht erklärt werden[133]. Daher wird jetzt bei jedem der hier folgenden Experimente der

[133] Keras. Accuracy metrics, <https://keras.io/api/metrics/accuracy_metrics> (5.08.2021). So ein ähnliches Verhalten wurde bereits in der Bachelorarbeit des Autors festgestellt. Bei der Auswertung des von A. Loyal trainierten VGG 16 Modells wurde eine Accuracy von ca. 91% errechnet. Bei der Ausweitung der Testmenge auf alle 64.000 Bilder der OCRE Datenbank mit den Porträts der trainierten Kaiserinnen und Kaiser sank die

Accuracy Wert eines manuellen Nachtests mitangegeben und somit untersucht, ob sich dieses Verhalten fortsetzt. Wenn man die problematischen Indizes (Key-Error) bei der Erstellung eines manuellen Top-5 Accuracy Wertes rausfiltert, erhält man einen Wert von ca. 97,5% (siehe Abb. 26), der sogar das erreichte Ergebnis von A. Loyal bei demselben Wert (92%) übertrifft[134].

```
Epoch 10/10
1285/1285 [==============================>.] - ETA: 0s - loss: 0.1714 - acc: 0.9453lr: 1e-05
1286/1285 [==============================] - 284s 221ms/step - loss: 0.1714 - acc: 0.9453 - val_loss: 0.5746 - val_acc: 0.8
157
```

Abb. 25: Der während dem Trainingsvorgang von Keras errechnete Accuracy Wert auf der Testmenge von Datensatz 1 Bildmenge 2.

```
All files: 6868,Top-1 correct: 5267, Top-5 correct:6696
Top-1 Accuracy: 0.7668899242865463
Top-5 Accuracy: 0.9749563191613279
```

Abb. 26: Die beim erneuten Durchlauf aller Bilder der Testmenge von Datensatz 1 Bildmenge 2 errechneten Accuracy Werte.

Fazit Insgesamt lässt sich das Experiment 1 folgendermaßen zusammenfassen: Es ist möglich, dass vollständige vortrainierte VGG16 Netz aus Keras zu laden und neu auf die 66 in dieser Arbeit verwendeten Klassen zu trainieren. Die Originalstruktur neigt bei zehn Trainingsepochen und einer stufenweise absinkender Lernrate nicht zu einer Überanpassung und eine Top-1 Test-Accuracy von 81,6% konnte erreicht werden (76,7% bei dem Nachtest). Die Top-5 Accuracy lag bei einem sehr guten Wert von ca. 97,5%. Es muss bei einem Einsatz allerdings auf die korrekte Umwandlung der Eingabebilder in ein Array mit Float32 Einträgen geachtet werden. Da letztendlich aber immer das Risiko eines Key Errors durch Indizes außerhalb der 66 Klassen in den Top-5 Ergebnissen besteht und die Erstellung eines eigenen VGG16 Endsegments mit einer Softmax-Schicht mit lediglich 66 Klassen möglicherweise bessere Ergebnisse bringen könnte, wird von der Verwendung des vollständig aus Keras geladenen Netzes abgesehen.

5.2 Experiment 2: Test zweier VGG16 Endstrukturen und der zugehörigen Hyperparameter

Dieses Experiment dient der Erprobung zweier unterschiedlicher VGG16 Endstrukturen und der zugehörigen Hyperparameter Lernrate und Epochenanzahl auf unterschiedlichen Bildermengen. Da das erste Experiment das Ergebnis erbracht hat, dass das Laden des

Accuracy auf einen Wert von 81%, obwohl diese Testmenge auch die Bilder der von A. Loyal benutzen Trainingsmenge enthielt. Gampe 2019, 19 f. Loyal 2018, 73–76 Tab. 6.1.

[134] Loyal 2018, 67.

vollständigen VGG16 Aufbaus mit den vortrainierten Gewichten und der 1000 Klassen Softmax Schicht Probleme verursachen kann, werden hier zwei Möglichkeiten des Aufbaus der vollständig verbundenen Schichten am Ende des Netzes getestet. Endstruktur 1 entspricht dabei dem Endsegment des originalen VGG16 ohne die vortrainierten Gewichte. Der Aufbau der Endstruktur 2 besitzt dagegen lediglich eine voll verbundene Schicht. Hier soll auch ein Augenmerk darauf gerichtet werden, wie sich die Hyperparameter Lernrate und Epochenanzahl auf das Verhalten während des Trainings auswirken. Da A. Loyal keine Lernrate für ihr finales Modell angegeben hatte, mussten verschiedene Lernraten und Epochenanzahlen hier erprobt werden[135]. Es werden außerdem die unterschiedlichen Bildermengen aus den zur Verfügung stehenden Datensätzen für Training und Tests genutzt (siehe Tab. 1 und Tab. 2). Das primäre Ziel dieses Experiment ist es, ein auf OCRE Daten trainiertes Modell zu finden, das auch auf den CN Porträts möglichst gute Ergebnisse erreicht.

a) Die Endstrukturen

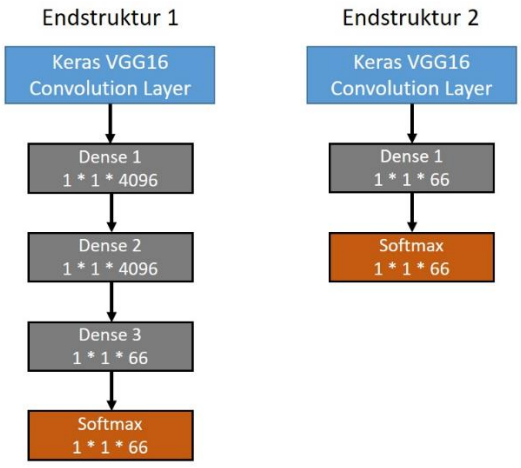

Abb. 27: Diagramm der beiden Endstrukturen. Die Softmax Aktivierungsfunktion wurde hier als eigene Schicht dargestellt.

Die beiden untersuchten Endstrukturen wurden bereits in Abschn. 4.4.a kurz vorgestellt. Endstruktur 1 ahmt den Aufbau des originalen VGG16 mit zwei vollständig verbundenen Schichten mit jeweils 4096 Kanälen, einer vollständig verbundenen Schicht mit 66 Kanälen und einer Softmax Ausgabe der 66 Klassen nach (anstatt der originalen 1000 Klassen, siehe Abb. 16 Abb. 17 und Abb. 27). Ein weiterer Unterschied zu der VGG16 Struktur von Experiment 1 ist, dass keine vortrainierten Gewichte in diesem Endsegment verwendet werden können. Somit müssen diese Gewichte mit den verschiedenen Bildermengen und Hyperparametern neu trainiert werden.

[135] Loyal 2018, 61. 73–76.

Die vom originalen Aufbau des VGG16 Modells abweichende Endstruktur 2 besteht dagegen aus einer Global Average Pooling Schicht, die auf eine vollständig verbundene Schicht mit 66 Kanälen überleitet, welche auch eine Softmax Aktivierung einsetzt (siehe Abb. 27). Auch diese Struktur wurde mit den verschiedenen Bildmengen trainiert und die verwendeten Hyperparameter überprüft.

Der Hintergrund der Tests mit der Endstruktur 2 ist der folgende: Zu Anfang der Implementierung des VGG16 Netzes mit Keras und Jupyter Notebook wurde diese Endstruktur nach einem Beispiel umgesetzt[136]. Dabei stellte sich die wesentlich einfachere und schlankere Endarchitektur als sehr leistungsfähig heraus. Gleichzeitig bietet sie mit ihrem geringeren Speicherplatzbedarf (ca. 58 MB zu 526 MB bei Endstruktur 1) und der dadurch verringerten Ladezeit (ca. 10 Sek. zu ca. 14 Sek.) auch Vorteile bei einem Einsatz als Webapplikation oder auf mobilen Geräten[137]. Dieser Größenunterschied kann mit der weit größeren Anzahl an trainierbaren Parametern der Endstruktur 1 begründet werden, da sie etwa neunmal mehr davon aufweist als Endstruktur 2 (siehe Abb. 17 und Abb. 18). Dieser Vorteil und die guten Werte bei den Metriken veranlassten den Autor die speicherplatzeffizientere Endstruktur 2 mit in das Experiment aufzunehmen.

b) Die Hyperparameter

In diesem Experiment wurde mit drei verschiedenen Lernraten und zwei Epochenanzahlen gearbeitet. Es sind folgende Lernraten zum Einsatz gekommen:

- **Lernrate 1**: Vier große, drei mittlere und drei kleine Lernschritte in **zehn Epochen** (siehe Abb. 21)
- **Lernrate 2**: Drei große. Drei mittlere und vier kleine Lernschritte in **zehn Epochen** (siehe Abb. 28 Links).

[136] Die Implementation in Jupyter Notebook wurde nach der folgenden Quelle durchgeführt: knowledge Transfer. How to use VGG model in TensorFlow Keras, <https://androidkt.com/how-to-use-vgg-model-in-tensorflow-keras> (01.08.2021). Siehe auch Fußnote 91.

[137] Die Zeitangaben beziehen sich auf das Laden des Modells mit der jeweiligen Endstruktur nach einem Neustart des hier verwendeten Rechners (siehe Abschn. 4.3 Arbeitsumgebung). In Kiourt – Evangelidis 2021 wurde einem MobileNetV2 Modell für die angestrebte Webapplikation zur Klassifikation von thrakischen Münzen den Vorzug gegeben, da es mit ca. 25 MB wesentlich kleiner war als das bei den *Top-1 Accuracy* Werten überlegene VGG16 Netz (ca. 120 MB). Hierbei spielen auch eher geringe Größenunterschiede eine Rolle, um Forschungsergebnisse in Form der Webapplikation einer interessierten Öffentlichkeit zugänglich zu machen. Kiourt – Evangelidis 2021, 59 f. Leider scheinen die Autoren das *Corpus Nummorum* Webportal zu dem Zeitpunkt ihrer Tests nicht gekannt zu haben, da sie sonst den mit 565 Münzen in zwölf unterschiedlichen Klassen doch eher beschränkten Datensatz erheblich hätten erweitern können. Mit der größeren Zahl an Bildern hätten sie das für ihren Ansatz effizienteste MobileNetV2 Modell sicherlich noch besser trainieren können. Beispielsweise ist die Klasse „Maroneia Horse“ mit 50 Exemplaren in dem Datensatz vertreten. Eine Suche auf dem CN Webportal mit dem Stichwort „Horse“ und der Münzstätte Maroneia ergibt über ca. 1040 Treffer mit Abgüssen und Originalen. Hier hätte lediglich noch die zur Klasse gehörenden Münzen mit der Darstellung einer Pferdeprotome von den Exemplaren mit dem Bildnis eines vollständigen Pferdes getrennt werden müssen.

- **Lernrate 3**: Drei große, zwei mittlere und ein kleiner Lernschritt in **sechs Epochen** (siehe Abb. 28 Rechts)

Erläuterung:

Großer Lernschritt: 0,001, mittlerer Lernschritt: 0,0001, kleiner Lernschritt: 0,00001.

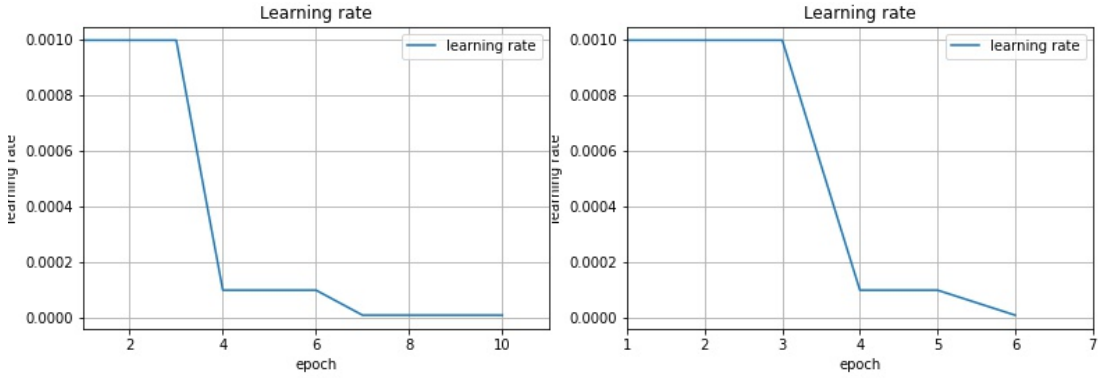

Abb. 28: Links: Verlauf der Lernrate 2; Rechts: Verlauf der Lernrate 3.

c) Die Datensets

Das Experiment umfasste Trainingsdurchläufe mit den verschiedenen Bildmengen von Datensatz 1 und 2. Weitere Tests wurden auf den Untermengen von Datensatz 3 gemacht (siehe auch Tab. 3). Sämtliche zu einem Modell gehörende Metriken (bis auf die von Datensatz 3) wurden auf den jeweiligen Testbildern der Bildmenge erstellt, mit welcher das Modell zuvor trainiert wurde. Die Metriken auf dem Datensatz 3 bilden dagegen jeweils die Ergebnisse auf allen Bildern der zwei zugehörigen Bildmengen ab.

d) Durchführung

Die Durchführung des Experiments besteht im ersten Schritt aus dem Training der Modelle mit beiden Endsegmenten auf den verschiedenen Bildermengen. Dabei kommen die vorgestellten Lernraten zum Einsatz. Insgesamt wurden hier achtzehn verschiedene Modelle trainiert (siehe Tab. 1 und Tab. 2). Aus den während und nach dem Training ermittelten Metriken wird ein erstes Fazit zu der Eignung der Modelle für die angestrebte kaiserliche Porträterkennung gezogen. Am Ende wird ein Gesamtresümee zu allen Modellen erstellt und das am besten abschneidende Modell wird anschließend in Experiment 3 einer genaueren Auswertung unterzogen (siehe Abschn. 5.3).

e) Modelle mit Endsegment 1

Der Ablauf des Trainings der Modelle mit Endsegment 1 und die Begründung für die erneuten Trainingsdurchläufe können in Abb. 29 eingesehen werden.

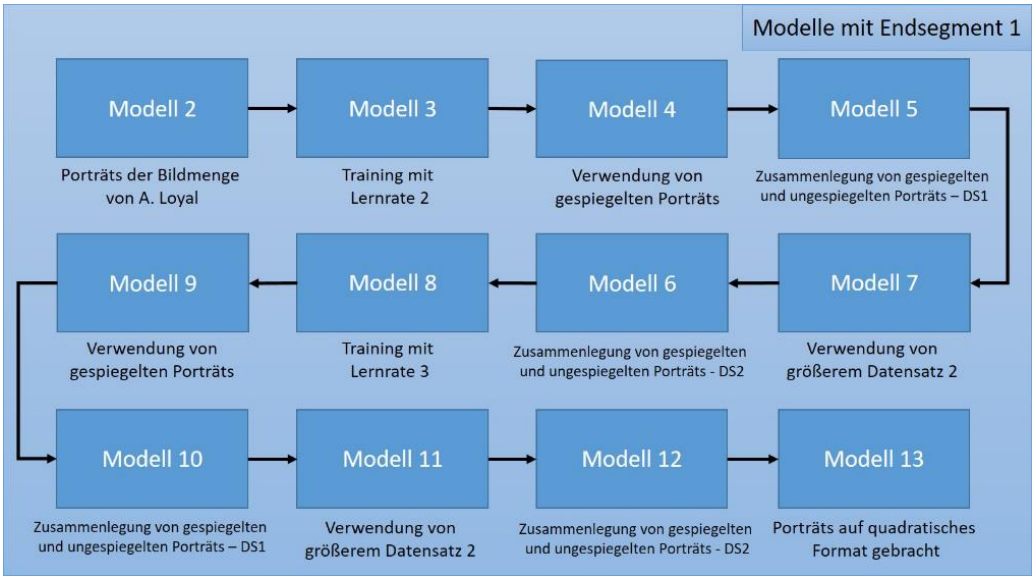

Abb. 29: Ablaufdiagram des Trainings der Modelle und Begründung dafür.

Die Ergebnisse der Metriken der hier trainierten Modelle sind in Tabelle 1 zu finden.

Tab. 1: Ergebnisse der Metriken der Modelle mit Endsegment 1.

Trainingsmenge / Lernrate	Lernrate 1	Lernrate 2	Lernrate 3
Bildmenge 1.1	Modell 2 Accuracy (Train): 82%, Loss(Train): 0,77, Top-1 Accuracy: 80%, Top-5 Accuracy: 98%, Precision: 81%, Recall: 80%, F1-Score: 80%, Overfitting: niedrig	Modell 3 Accuracy (Train): 83%, Loss(Train): 0,77, Top-1 Accuracy: 80%, Top-5 Accuracy: 98%, Precision: 80%, Recall: 80%, F1-Score: 80%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 2%, Top-5 Acc.: 10% CN (3.2): Top-1 Acc.: 8%, Top-5 Acc.: 21%	Modell 8 Accuracy (Train): 83%, Loss(Train): 0,71, Top-1 Accuracy: 80%, Top-5 Accuracy: 98%, Precision: 81%, Recall: 80%, F1-Score: 80%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 3%, Top-5 Acc.: 10% CN (3.2): Top-1 Acc.: 8%, Top-5 Acc.: 20%
Bildmenge 1.2	Nicht trainiert	Model 4 Accuracy (Train): 82%, Loss(Train): 0,78, Top-1 Accuracy: 79%, Top-5 Accuracy: 97%, Precision: 80%, Recall: 79%, F1-Score: 78%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 1%, Top-5 Acc.: 4% CN (3.2): Top-1 Acc.: 1%, Top-5 Acc.: 5%	Modell 9 Accuracy (Train): 82%, Loss(Train): 0,69, Top-1 Accuracy: 80%, Top-5 Accuracy: 98%, Precision: 81%, Recall: 80%, F1-Score: 80%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 1%, Top-5 Acc.: 4% CN (3.2): Top-1 Acc.: 1%, Top-5 Acc.: 5%
Bildmenge 1.3	Nicht trainiert	Model 5 Accuracy (Train): 82%, Loss(Train): 0,84, Top-1 Accuracy: 79%, Top-5 Accuracy: 97%, Precision: 80%, Recall: 79%, F1-Score: 79%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 2%, Top-5 Acc.: 7% CN (3.2): Top-1 Acc.: 7%, Top-5 Acc.: 19%	Modell 10 Accuracy (Train): 82%, Loss(Train): 0,76, Top-1 Accuracy: 78%, Top-5 Accuracy: 98%, Precision: 80%, Recall: 78%, F1-Score: 78%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 4%, Top-5 Acc.: 10% CN (3.2): Top-1 Acc.: 7%, Top-5 Acc.: 20%
Bildmenge 2.1	Nicht trainiert	Model 6 Accuracy (Train): 86%, Loss(Train): 0,70, Top-1 Accuracy: 85%, Top-5 Accuracy: 98%, Precision: 85%, Recall: 85%, F1-Score: 85%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 26%, Top-5 Acc.: 53% CN (3.2): Top-1 Acc.: 28%, Top-5 Acc.: 53%	Modell 11 Accuracy (Train): 86%, Loss(Train): 0,66, Top-1 Accuracy: 85%, Top-5 Accuracy: 98%, Precision: 85%, Recall: 85%, F1-Score: 85%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 23%, Top-5 Acc.: 49% CN (3.2): Top-1 Acc.: 27%, Top-5 Acc.: 54%
Bildmenge 2.2	Nicht trainiert	Model 7 Accuracy (Train): 86%, Loss(Train): 0,71, Top-1 Accuracy: 86%, Top-5 Accuracy: 98%, Precision: 86%, Recall: 86%, F1-Score: 86%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 23%, Top-5 Acc.: 54% CN (3.2): Top-1 Acc.: 27%, Top-5 Acc.: 53%	Modell 12 Accuracy (Train): 86%, Loss(Train): 0,62, Top-1 Accuracy: 86%, Top-5 Accuracy: 98%, Precision: 86%, Recall: 86%, F1-Score: 86%, Overfitting: niedrig CN (3.1): Top-1 Acc.: 25%, Top-5 Acc.: 51% CN (3.2): Top-1 Acc.: 25%, Top-5 Acc.: 50%
Bildmenge 2.3	Nicht trainiert	Nicht trainiert	Modell 13 Accuracy (Train): 86%, Loss(Train): 0,60, Top-1 Accuracy: 86%, Top-5 Accuracy: 99%, Precision: 85%, Recall: 86%, F1-Score: 85%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 19%, Top-5 Acc.: 50% CN (3.2): Top-1 Acc.: 26%, Top-5 Acc.: 54%

Modell 2 Das erste Modell (2) dieses Experiments wurde auf Bildmenge 1.1 mit Lernrate 1 trainiert. Die Bildmenge entspricht somit der auf die Porträts reduzierten Trainings- und Testmenge von A. Loyal. Die Top-1 Accuracy auf der Trainingsmenge steigt von ca. 50% in Epoche eins auf ca. 99,9% in Epoche zehn, was dafürspricht, dass kein weiterer Erkenntnisgewinn aus den Trainingsdaten zu erwarten ist. Vielmehr besteht bei einer solchen Anpassung an die Trainingsdaten die Gefahr des Overfittings. Die während des Trainings ausgegebene Top-1 Accuracy der Testmenge steigt in Zeitraum der zehn Epochen von ca. 69% auf ca. 82%. Es kann nach Epoche sechs kein nennenswerter Anstieg der Accuracy festgestellt werden (siehe Abb. 30 Links). Der Wert der Top-1 Accuracy im Nachtest betrug ca. 80% und der Top-5 Wert konnte ca. 98% erreichen. Die Loss Metrik auf der Testmenge zeigt ihren Tiefpunkt nach Epoche vier mit einem Wert von 0,63. Bis Epoche zehn ist jedoch wieder ein Anstieg auf 0,76 zu messen, was wiederrum ein Anzeichen für Overfitting sein könnte (siehe Abb. 30 Rechts). Die Recall, Precison und F1-Score Metriken erreichten jeweils um die 80% (siehe Tab. 1).

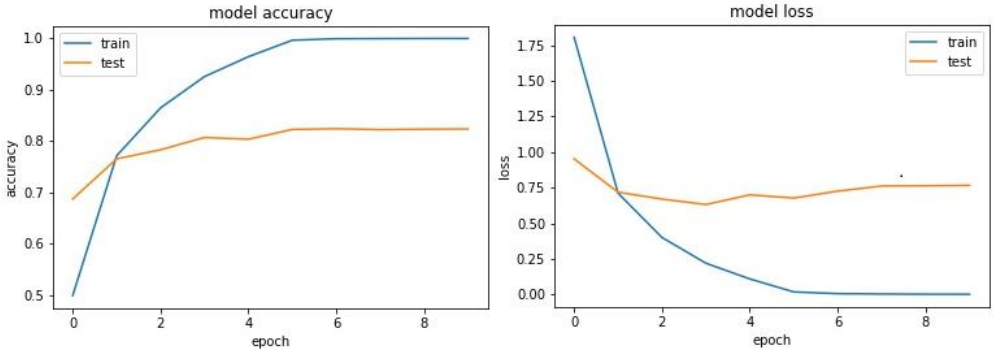

Abb. 30: Links: Verlauf der Accuracy von Modell 2 während des Trainings; Rechts: Verlauf des Loss von Modell 2 während des Trainings.

Das Modell 2 zeigt allgemein eine gute Leistung anhand der ermittelten Metriken, welche die Ergebnisse von Modell 1 sogar noch übersteigen. Jedoch werden wiederrum die Top 1 Werte, die A. Loyal mit ihrem finalen Modell erzielte (92%), nicht erreicht. Da Modell 2 mögliche Anzeichen der Überanpassung an die Trainingsdaten zeigte, wurde hier eine Confusion Matrix erstellt, um dies zu überprüfen (siehe Abb. 49). Dort können auch nur wenig Anhaltspunkte für ein größeres Problem mit der Überanpassung gefunden werden. Auffällige Punkte abseits der Diagonalen der Confusion Matrix finden sich vor allem bei den Familiendynastien, deren Porträtdarstellungen sich besonders stark ähneln. Zu nennen sind hier besonders die konstantinische Dynastie (siehe Abb. 31 Links) und die valentianische Dynastie (siehe Abb. 31 Rechts). In der Dynastie des Konstantins des Großen werden besonders seine drei Söhne Constans, Constantius II. und Konstantin II., die das römische Reich zeitweise gemeinsam

beherrschten, miteinander verwechselt. Die in diesen Fällen gegebene sehr niedrige Interklassenvarianz scheint das Modell vor Probleme zu stellen. Auch das gleichartige Problem bei der valentianischen Dynastie wird auf die sehr ähnliche Ikonographie von Valentinian I. und seinem Bruder Valens zurückgehen (vgl. Abb. 32 Links und Mitte). Auch A. Loyal stellte in Ihrer Masterarbeit bereits fest, dass die beiden Brüder des Öfteren miteinander verwechselt werden. Ebenso werden Porträts des Gratian, dem Sohn Valentinians I., des Öfteren als sein Onkel Valens erkannt (vgl. Abb. 32 Mitte und Rechts)[138]. Diese Probleme werden daher weniger auf eine Überanpassung zurückzuführen sein.

Obwohl das Modell 2 trotz der Auffälligkeiten in den Metriken eine eher geringe Tendenz zum Overfitting zeigt, wird im nächsten Schritt mit einer leicht angepassten Lernrate probiert die Tendenz zum Overfitting in den Metriken und die Werte abseits der Diagonalen in der Confusion Matrix zu senken.

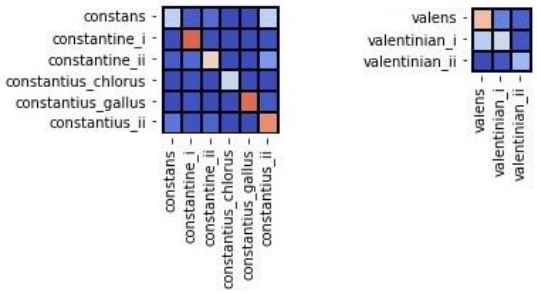

Abb. 31: Links: Die Konstantinische Dynastie in der Confusion Matrix von Modell 2 (siehe Abb. 49); Rechts: Die Valentianische Dynastie in derselben Confusion Matrix. Beide Abildungen zeigen vermehrt Verwechslungen der Klassen untereinander durch Werte abseits der Diagonalen.

Abb. 32: Vergleich von den ikonographisch sehr ähnlichen Porträts des Valentinian I. (links), Valens (Mitte) und Gratian (rechts). Links: Münze des Typus RIC IX Antioch 2B; Mitte: Münze des Typus RIC IX Antioch 2C; Rechts: Münze des Typus RIC IX Treveri 39C.

Modell 3 Dieses Modell wurde jetzt mit Lernrate 2 auf der Bildmenge 1.1 mit zehn Epochen trainiert. Trotz der etwas geringeren Lernrate mit einem großen Schritt weniger und einem kleinen Schritt mehr sehen die Metriken sehr ähnlich zu Modell 2 aus (siehe Tab. 1). Die von Keras ausgegebene Testmengen Accuracy steigt minimal auf 83%, wobei dieser Wert in Epoche sechs erreicht wird und danach auf diesem Niveau verharrt (siehe Abb. 33 Links). Auch bei der Loss Metrik wird der Tiefpunkt in Epoche fünf erreicht (0,65) und es gibt einen leichten

[138] Loyal 2018, 51 f.

Anstieg danach auf 0,77 (siehe Abb. 33 Rechts). Ein erster Test des Modells auf den ausgeschnittenen Porträts der CN Bildmengen 3.1 zeigt sehr schlechte Werte. Auch die Ergebnisse auf der größeren Bildmenge 3.2 können bei der Top-1 und Top-5 Accuracy nicht überzeugen (siehe Tab. 1).

Die Metriken bestätigen dem Modell 3 eine ähnlich gute Leistung wie dem Vorgängermodell auf den OCRE Daten. Auch die Confusion Matrix des dritten Modells zeigt trotz der leicht angepassten Lernrate keine große Veränderung. Es fällt jedoch auf, dass auch bei allen Mitgliedern der valentianischen Dynastie und der theodosianischen Dynastie untereinander ein größeres Verwechslungspotential bei diesem Modell besteht (siehe Abb. 50)[139]. Der Grund wird ebenfalls in den sehr ähnlichen kaiserlichen Porträtdarstellungen des 4. und 5. Jhs. n. Chr. liegen, da dieses eher abstrakt wirkt und weniger Wert auf individuelle Merkmale legt[140].

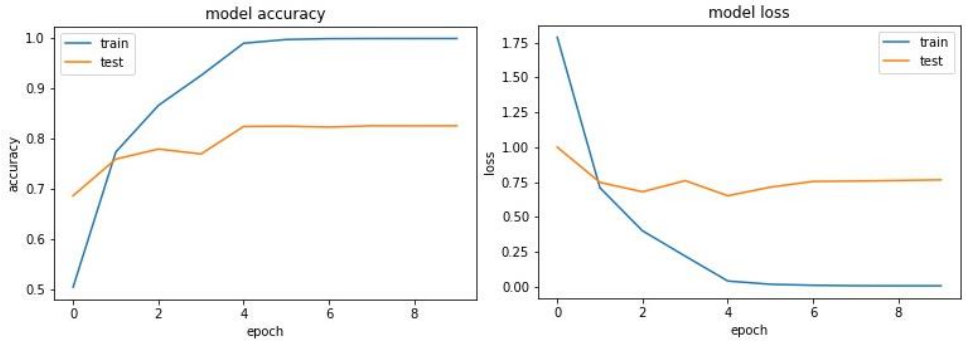

Abb. 33: Links: Verlauf der Accuracy von Modell 3 während des Trainings; Rechts: Verlauf des Loss von Modell 3 während des Trainings.

Das Modell 3 mit der Lernrate 2 erreicht in etwa dieselben Werte in den Metriken wie Modell 2. Jedoch sind auch die Anzeichen des Overfittings immer noch vorhanden. Die zugehörige Confusion Matrix kann aber keine Belege für ein größeres Problem damit zeigen. Lediglich die Familienähnlichkeit der Dynastien führt weiterhin zu Verwechslungen. Auf den CN Daten liefert das Modell eine schlechte Performance (siehe Tab. 1). Das Modell erreichte zwar bessere Werte auf diesen Bildern als das von A. Loyal trainierte Netz, welches in der Bachelor Arbeit des Autors auf diesen Daten getestet wurde (damals erreichte Top-1 Accuracy auf Bildmenge 3.1: 2% und Top-5 Accuracy: 5%), kann Numismatikern in diesem Arbeitsbereich aber noch keine Hilfe sein[141]. Im nächsten Schritt wird jetzt geprüft, wie sich eine Umkehrung der Porträtausrichtungen in den Trainings- und Testdaten auf das Verhalten eines Modells auswirkt.

[139] Zu diesen Dynastien gehören und sind in den 66 Klassen berücksichtigt: *Valentianische Dynastie*: Valentinian I., Valens (Bruder des Valentinian I.), Gratian (Sohn des Valentinian I.), Valentinian II. (Sohn des Valentinian I.). *Theodosianische Dynastie*: Theodosius I. der Große (Mitkaiser der Söhne von Valens, jedoch nicht mit ihnen verwandt), Arcadius (Sohn des Theodosius I.), Honorius (Sohn des Theodosius I.) und Theodosius II. (Sohn des Arcadius).
[140] Siehe zum Kaiserbild des 4. und 5. Jhs. n. Chr.: Peschlow 1983.
[141] Gampe 2019, 31 f.

Modell 4 Das Modell 4 wurde mit der Bildmenge 1.2 und Lernrate 2 einem Training mit zehn Epochen unterzogen. Dieses Modell erreicht bei den erhobenen Metriken eine sehr ähnliche Performance wie das dritte Modell. Zwar sind sämtliche Werte minimal schlechter geworden, ändert das aber nichts an der guten Leistung auf den Testdaten. Dies verwundert auch nicht, da Trainings und Testdaten bis auf die Spiegelung der Porträts dieselben sind und das Netz die Klassen mit diesen veränderten Daten genauso gut lernen kann. Somit ist es für dieses Modell interessanter die nicht-gespiegelten Testdaten aus der Bildmenge 1.1 als Eingabe zu benutzen. Hierbei zeigt sich ein gänzlich anderes Bild. Die Top-1 Accurcay sinkt auf 17% und die Top-5 Accuracy auf 19% (siehe Abschn. 9.3 Auflistung 1). Dies weist auf ein starkes Ungleichgewicht der Verteilung der nach links oder rechts im Profil gerichteten Porträts hin. Und tatsächlich ergibt ein Blick in die Ordner der Klassen mit den meisten korrekten Vorhersagen aus Bildmenge 1.1, dass beispielsweise von den 168 zugeordneten Bildern der Klasse „Augustus“ in der Testmenge lediglich 40 eindeutig nach links ausgerichtet sind. Die Menge der nach rechts gerichteten Porträts ist somit viel größer. Da das Modell 4 durch die Spiegelung auf größtenteils nach links gerichteten Augustusporträts trainiert wurde, kann es die vielen nach rechts gerichteten Bilder dieser Testmenge wohl schlechter erkennen. Nach einem Blick auf die Gesamtdaten kann gesagt werden, dass bei den Porträts der hier behandelten Quellen OCRE und CN eindeutig eine Bevorzugung der Ausrichtung nach rechts ausgemacht werden kann. Es scheint, dass die Ausrichtung nach links noch am ehesten bei den Kaiserdarstellungen auf Münzen des 1. Jhs. n. Chr. beliebter gewesen ist.

Die drei Klassen mit den größten Prozentsätzen an korrekten Top- 1 Vorhersagen von Modell 4 auf Bildmenge 1.1 sind: „Vespasian“ (79%), „Probus“ (72%) und „Augustus“ (71%) (siehe Abschn. 9.3 Auflistung 1). Die kleinste Anzahl von nach rechts gespiegelten Porträts dieser drei Klassen besaß „Vespasian“ mit 21 Bildern (von 291 insgesamt)[142]. Diese doch eher kleine Anzahl an in Trainingsmenge 1.2 war wohl zusammen mit den 270 jetzt nach links gespiegelten Porträts ausreichend, um die 93 nach rechts gerichteten (ungespiegelten) Vespasiansporträts in Testmenge 1.1 zu einem Großteil zu erkennen. Eine schlechtere Top-1 Erkennungsrate von ca. 23% konnte bei der Klasse „Licinius“ festgestellt werden, obwohl diese 24 nach rechts gespiegelte Porträts in der Trainingsmenge von Datensatz 1.2 aufweist. Lediglich die Top-5 Erkennungsrate von ca. 79% ist hier zufriedenstellend. Auch die 50% Top-1 Erkennungsrate der Klasse „Nero“ kann angesichts der 34 (von 134) nach rechts gespiegelten Porträts nur ansatzweise überzeugen. Die Top-5 Accuracy fällt mit 85% dagegen wieder gut aus.

[142] Die Klasse „Probus“ besitzt 76 und die Klasse „Augustus“ 109 nach links ausgerichtete Porträts in dieser Trainingsmenge.

Somit kann geschlossen werden, dass eine überzeugende Top-5 Erkennung von nach links gerichteten Kaiserporträts ca. zwanzig derartig gerichteten Darstellungen in der Trainingsmenge nötig sind. Diese Anzahl könnte sich, wenn keine ausreichende Anzahl in der vorhandenen Trainingsmenge gegeben ist, notfalls mit einer Spiegelung nach rechts ausgerichteter Porträts künstlich herstellen lassen, wobei eine größere Anzahl die Wahrscheinlichkeit von besseren Ergebnissen erhöhen dürfte. Bei Klassen, die über keinerlei nach links gerichtete Porträts verfügen, da dies unter dem jeweiligen Kaiser wohl nicht in Mode war, bringt diese Vorgehensweise wahrscheinlich keine besseren Ergebnisse. Ob die Zusammensetzung einer Trainingsmenge aus den normalen Bildern und ihren Spiegelungen einen Vorteil bei der Erkennungsrate bringt, wird jetzt anschließend geprüft.

Modell 5 Ein zehn Epochen langes Training mit Lernrate 2 auf der Bildmenge 1.3 führte zu Modell 5. Die Bildmenge 1.3 fasst die Mengen 1.1 und die gespiegelten Porträts von 1.2 zusammen, so dass hier eine verdoppelte Trainings- und Testmenge vorliegt. Bei den Metriken kann das Modell sich allerdings nicht von seinen Vorgängern abheben. Die Top-1 und Top-5 Accuracy bleiben sowohl bei den am Ende des Trainings und im Nachtest erhobenen Werten bis auf minimale Unterschiede gleich. Der Loss Wert steigt sogar noch leicht an (siehe Tab. 1). Dies stellt jedoch keine Überraschung dar, da jedes individuelle Münzporträt in der Testmenge noch einmal mit seiner Spiegelung vertreten ist. Jedoch kann man daran sehen, dass dieses Modell in der Lage ist, während des Trainings noch nicht gesehenen Porträts unabhängig von ihrer Ausrichtung größtenteils korrekt zu bestimmen[143]. Während Modell 4 noch auf den Testbildern von Bildmenge 1.1 große Probleme hatte, kann dieses Modell sowohl mit Porträtsammlungen die entweder größtenteils nach links oder nach rechts gerichtet sind gleich gut umgehen, da jedes Testbild in beiden Richtungen vorliegt. Somit kann man ein Modell mit der relativ einfachen Operation der Spiegelung besser auf die möglichen Ausrichtungen eines antiken Münzporträts vorbereiten. Das Problem des Overfittings kann auch bei diesem Modell nur in geringer Ausprägung festgestellt werden. Die Confusion Matrix zeigt, dass die Probleme mit den Familiendynastien immer noch bestehen (siehe Abb. 51). Der Test des Modells auf Bildmengen 3.1 und 3.2 weist aber ebenfalls keine Steigerung der Werte im Vergleich zu Modell 3 auf (siehe Tab. 1). Somit kann ein Problem mit den CN Daten aufgrund einer großen Zahl nach links gerichteter Porträts ausgeschlossen werden. Um jetzt noch bessere Werte in der Kaisererkennung erreichen zu können ist es also notwendig die Anzahl der individuellen Trainingsmünzen ohne eine Operation wie die Spiegelung zu steigern.

[143] Auch eine Nachtest mit den individuellen Trainingsmengen von 1.1 und 1.2 brachte auf beiden Datensätzen eine Top-1 Accuracy von ca. 79%-

Modell 6 Das hier vorgestellte Modell 6 wurde mit Lernrate 2 in zehn Epochen auf den wesentlich zahlreicheren Trainingsdaten von Bildmenge 2.1 (ca. 47.000 Münzen) trainiert. Neben der gesteigerten Bilderanzahl musste hier auch erstmals eine neue Aufteilung in Trainings- und Testmenge durchgeführt werden (siehe Abschn. 4.2). Die Metriken des Modells (siehe Tab. 1) können vor allem bei der Top-1 Accuracy von 86% am Ende des Trainings (siehe Abb. 34 Links) und 85% im Nachtest noch einmal zulegen. Die Top-5 Accuracy bleibt mit ihrem schon sehr guten Wert von 98% auf dem Niveau der Vorgängermodelle. Der Loss Wert unterbietet des bisherigen besten Wert von Modell 3 (0,77) mit 0,7 noch einmal. Auch hier kann man sehen, dass der Loss Wert aber in Epoche 3 seinen Tiefpunkt erreicht hatte und danach wieder einen leichten Zuwachs zeigt (siehe Abb. 34 Rechts).

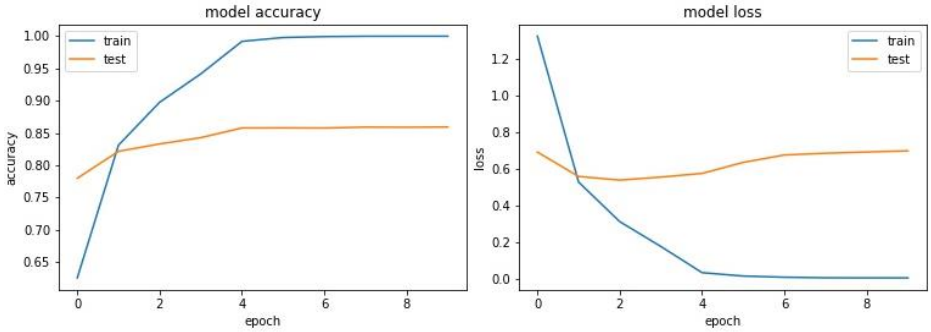

Abb. 34: Links: Verlauf der Accuracy von Modell 6 während des Trainings; Rechts: Verlauf des Loss von Modell 6 während des Trainings.

Auch die Precision und Recall Werte (und somit auch der F1 Score) konnten jeweils um knapp 5% gesteigert werden. Insgesamt erreicht das Modell somit die bisher besten Werte. Bei der Überanpassung an die Trainingsdaten zeigt die Confusion Matrix eine rückläufige Tendenz an Werten abseits der Diagonalen (siehe Abb. 52). Bei den Familiendynastien gibt es bei der Konstantinischen Dynastie kaum Änderungen zu vermerken. Lediglich bei Kaiser Constantius I. (Klasse „Constantius_Chlorus“) kann eine Zunahme der korrekten Erkennung festgestellt werden. Die Problematik der geringen Inter-Klassen-Varianz von Constans, Constantius II. und Konstantin II. kann nicht durch die neuen Daten behoben werden[144]. Bei der Valentianischen Dynastie ist eine Verbesserung zu erkennen. Der Kaiser Valentinian I. wird jetzt etwas weniger häufig mit seinem Bruder Valens verwechselt (siehe Abb. 52 mit der roten Markierung). Zum ersten Mal kann jetzt auch eine deutliche Steigerung der Metriken auf den Münzen der CN Bildmengen 3.1 und 3.2 erkannt werden (siehe Tab. 1). Die Top-1 Accuracy erreicht auf dem größeren Datensatz 3.2 einen Wert von 28% und die Top-5 Accuracy 53%. Damit zeigt die

[144] Obwohl die Anzahl der Trainingsbilder für alle drei Klassen gegenüber der Bildmenge 1.1 noch einmal angestiegen ist: Constans von 417 auf 724 Bilder; Constantius II. von 1563 auf 2399 und Konstantin II. von 601 auf 1066.

Top-5 Vorhersage das erste Mal das Potential den Numismatikern eine Hilfe zu sein. Anschließend wurde jetzt noch einmal der Frage nachgegangen, ob die Verdoppelung der Porträts durch die Spiegelung noch einen Einfluss auf die Erkennungsraten nehmen kann.

Modell 7 Das siebte Modell wurde mit zehn Epochen und Lernrate 2 auf der Bildmenge 2.2 trainiert. Auch hier liegt wie bei Modell 5 eine Verdoppelung der Trainings und Testmenge vor, jedoch mit dem Unterschied, dass hier nicht jeweils beide Mengen einzeln gespiegelt wurden, sondern dass die Gesamtmenge der Bilder zuerst gespiegelt wurde und anschließend eine Aufteilung in Trainings- und Testmenge erfolgte. Somit besteht beispielsweise die Testmenge nicht aus dem individuellen Münzporträt und seiner Spiegelung, sondern sie enthält möglicherweise nur das normale Porträt oder nur seine Spiegelung oder eben beides. In den Metriken ist zu sehen, dass eine sehr kleine Steigerung der Werte Accuracy, Precision, Recall und des F1-Scores um ca. 1% erfolgt ist. Bei den CN Daten bleibt das Modell auf dem Niveau von Modell 6 (siehe Tab. 1). Die Overfitting Werte in der Confusion Matrix erreichen ebenfalls die guten Werten des Vorgängermodells und bei den Familiendynastien gibt es keine Änderungen (siehe Abb. 51).

Da das Modell durch die gespiegelten Porträts somit besser auf dem Umgang mit links wie auch rechts ausgerichteten Porträts geschult ist, kann dieses Modell dem Modell 6 auch aufgrund seiner leicht besseren Werte bei den Metriken vorgezogen werden. Im nächsten Schritt wurde nun versucht das Problem des Overfittings noch einmal mit einer Verkürzung der Trainingsepochenanzahl auf sechs und einer angepassten Lernrate zu begegnen. Dieser Entscheidung wurde wegen der Beobachtung gefällt, dass in allen Fällen die Test Accuracy und Loss während des Trainings vor oder mit dem Abschluss von Epoche sechs ihren Hoch- bzw. Tiefpunkt erreichen (siehe dazu Abb. 30, Abb. 33 und Abb. 34). Die letzten vier Epochen bestehen im Falle der Accuracy nur noch aus Stagnation oder im Falle des Loss aus einem leichten Anstieg der Werte. Daher wurde eine kürzere Trainingsdauer untersucht.

Modell 8 – 11 Das Training der hier untersuchten Modelle mit Lernrate 3 (siehe Abb. 28 Rechts) dauerte sechs Epochen auf den verschiedenen Bildmengen. Grundsätzlich wurden noch einmal die verschiedenen Trainingsmengen und ihre Auswirkung auf die Metriken eines Modells untersucht. Da keine größeren Änderungen zu den Modellen 3 – 7 gefunden wurden, werden die ersten vier Modelle dieser Reihe hier summarisch beschrieben. Das erste Modell (8) weist fast genau dieselben erreichten Ergebnisse bei den Metriken auf wie das Modell 3, welches mit zehn Epochen trainiert wurde (siehe Tab. 1 und Abb. 35 Links). Jedoch konnte der Loss von 0,77 auf 0,71 gesenkt werden (siehe Abb. 35 Rechts). Ein Anstieg des Wertes ist jetzt nur noch in der sechsten und letzten Trainingsepoche zu sehen. Damit wurde die Gefahr eines

Overfittings etwas gemildert. Die Confusion Matrix zeigt aber auch in etwa dasselbe Bild wie bei Modell 3 (vgl. Abb. 50 und Abb. 54). Die Werte auf den CN Daten bleiben auch auf dem gleichen schlechten Niveau. Auch Modell 9 ist bei den Metriken zu Modell 4 vergleichbar. Jedoch konnte der Loss bei diesem Modell auf 0,69 gesenkt werden. Dies gilt auch für Modell 10 und sein Pendant Modell 5. Hier sank der Loss von 0,84 (5) auf 0,76 (10). Bei den Modellen, die auf dem größeren Datensatz 2 trainiert wurden, ist das Bild sehr ähnlich. Modell 11 und Modell 12 können ihre Losswerte gegenüber ihren Vergleichsmodellen 6 und 7 beide noch einmal leicht senken auf 0,66 (11) und 0,62 (12). Damit kann Modell 12 den bisher geringsten Losswert für sich verbuchen, auch wenn dieser während des Trainings bereits einmal bei 0,52 lag und danach wieder leicht angestiegen ist.

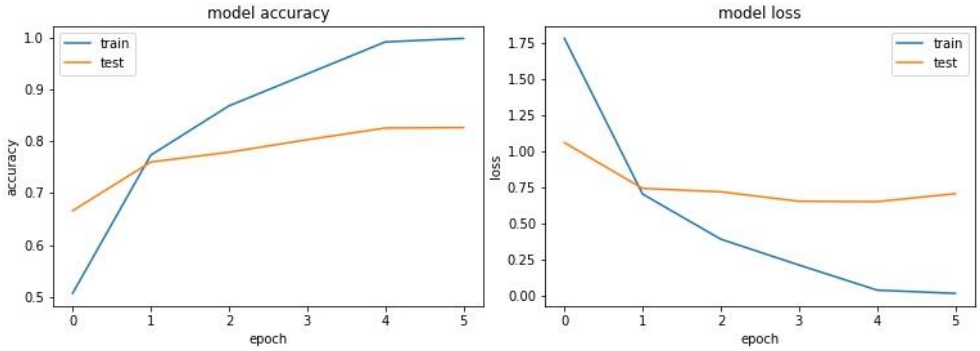

Abb. 35: Links: Verlauf der Accuracy von Modell 8 während des Trainings; Rechts: Verlauf des Loss von Modell 8 während des Trainings.

Insgesamt kann gefolgert werden, dass die Reduzierung der Trainingsdauer auf sechs Epochen und die daran angepasste Lernrate 3 für die Modelle und ihre erreichten Ergebnisse bei den Metriken keine Nachteile gebracht hat. Sämtliche Werte erreichen das Niveau der Vergleichsmodelle mit zehn Epochen Trainingsdauer (siehe Tab. 1). Der Loss Wert konnte jedoch durchgehend für alle Modelle gesenkt werden und somit die Gefahr einer möglichen Überanpassung noch einmal etwas gesenkt werden. Jetzt blieb noch zu überprüfen wie sich das bislang von den Metriken her gesehene beste Modell 12 auf den Porträts von vollständigen Münze schlug. Dazu wurde die von A. Loyal erstellte Testmenge mit vollständigen Münzen dem Modell 12 als Eingabe vorgelegt. Im Zuge dessen konnte das Modell eine Top-1 Accuracy von 11% und eine Top-5 Accuracy von 31% erreichen. Eine Verschlechterung der Werte war hier aufgrund der für das Netzwerk neuen Elemente wie der Legende und den anderen Bestandteilen einer Münze zu erwarten, aber diese niedrigen Werte waren doch überraschend. Eine mögliche Erklärung dafür hängt mit dem Format der Bilder nach dem Ausschneiden durch die Porträtbereichserkennung zusammen. Sämtliche Porträts haben danach ein hochrechteckiges Format (siehe Abb. 10 und Abb. 15). Bei der Skalierung auf die quadratische Größe von 224 * 224 Pixel, dem Eingabeformat des VGG16 Netzes, werden diese in der Breite

verzerrt. Somit benutzt das Netz während des Trainings diese verzerrten Porträts, um einen Kaiser zu erkennen. Solange man dem trainierten Netzwerk danach Bilder eingibt, die auch von der Porträtbereichserkennung bearbeitet wurden, stellt dies kein Problem dar, weil diese Porträts ja ebenfalls hochrechteckig sind und somit verzerrt werden. Wenn man aber ein bereits quadratisches Bild einer vollständigen Münze als Eingabe nutzt, auf welchem der Kaiser nicht verzerrt dargestellt ist, kann dies wohl zum Problem werden. Zur Überprüfung dieser Annahme wurde noch ein Modell mit einer angepassten Bildmenge trainiert.

Modell 13 Das Modell 13 wurde mit sechs Epochen und Lernrate 3 auf der noch nicht zum Einsatz gekommenen Bildmenge 2.3 trainiert. Diese zeichnet sich dadurch aus, dass sämtlichen Porträts seitlich schwarze Balken hinzugefügt wurden, um das bisher hochrechteckige Bild wieder in ein quadratisches Format zu bringen (siehe Abb. 11 Rechts)[145]. Ein mögliches Problem dabei könnte sein, dass nun ein recht großer Teil des auf das 224 * 224 Pixel verkleinerte Bild von diesen Balken eingenommen wird und somit auch weniger Pixel mit relevanten Informationen zur Klasse dem Netz zur Verfügung stehen. Dies kann jedoch durch die nach dem Training zur Verfügung stehenden Metriken widerlegt werden (siehe Tab. 1). Die Werte des bisher besten Modells 12 können sogar übertroffen werden. Die Top-5 Accuracy steigt auf ca. 99% und der Loss erreicht einen neuen Tiefpunkt mit dem Wert 0,6. Zwar kann der beste von A. Loyal erreichte Wert von 92% bei der Top-1 Accuracy nicht erreicht werden, dafür übersteigt der Top-5 Wert ihren besten Wert von 92%[146]. Auch bei CN Bildern gehört das Modell mit einer Top-5 Accuracy von 54% zu den bisher besten Modellen auf diesem Datensatz. Die Performance auf den vollständigen Münzen der Testmenge von A. Loyal wurde ebenfalls auf eine Top-1 Accuracy von 34% und eine Top-5 Accuracy von 58% gesteigert. Mit diesen im Vergleich zu Modell 12 deutlich besseren Werten kann gezeigt werden, dass das vermutete Problem mit der Verzerrung eine Rolle gespielt hat. Außerdem können die Bedenken zerstreut werden, dass die durch die Balken verkleinerte Bilder sich negativ auf die Ergebnisse der Metriken auswirken könnten. Insgesamt gesehen ist somit Modell 13 das bisher in vielen Bereichen der hier untersuchten Münzbestände variabel einsetzbarste Modell.

Fazit Der erste Teil des Experiments hat als Ergebnis erbracht, dass eine Trainingsdauer mit sechs Epochen völlig ausreichend bemessen ist, da hier die gleichen guten Werte bei den Metriken wie bei einem längeren Training erreicht werden. Ein größeres Problem mit einer Überanpassung an die Trainingsdaten konnte bei allen Modellen nicht festgestellt

[145] Eine sehr ähnliche Technik benutzte A. Loyal in ihrer Masterarbeit um die kombinierten Vorder- und Rückseitenbilder für ihre Typenerkennung in ein quadratisches Format zu bringen. Loyal 2018, 33.
[146] Loyal 2018, 74–76 mit Tab. 6.1.

werden. Weiterhin sind die auf gespiegelten und ungespiegelten Bildern trainierten Modelle insgesamt besser auf Datensätze mit einem größeren Anteil von nach links oder rechts ausgerichteten Porträts vorbereitet. Außerdem sollte auch die Verzerrung des hochrechteckigen Porträtformats durch die Hinzufügung von schwarzen Balken zur Herstellung eines quadratischen Bildformats korrigiert werden, da dies keine schlechteren Ergebnisse bringt. Schließlich führt eine größere Anzahl von Trainingsbildern zu deutlich verbesserten Ergebnissen (vor allem auf den CN Daten).

f) Modelle mit Endsegment 2

Der Ablauf des Trainings der Modelle mit Endsegment 2 und die Begründung für die erneuten Trainingsdurchläufe können in Abb. 36 eingesehen werden.

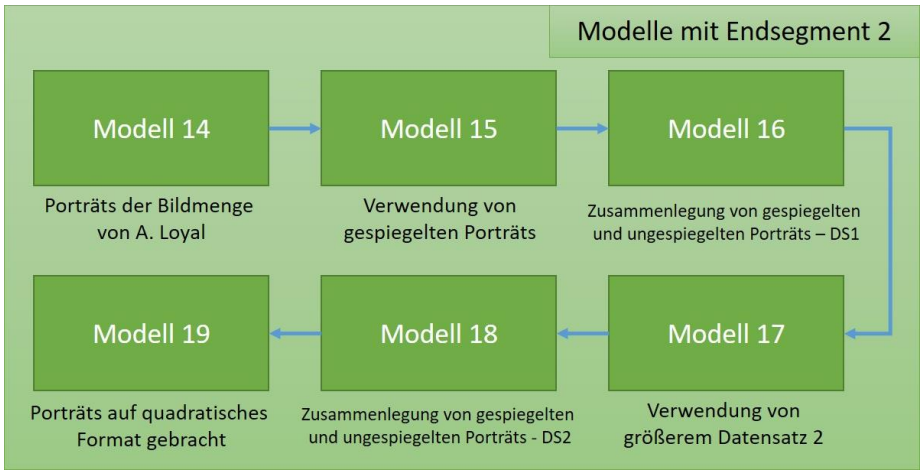

Abb. 36: Ablaufdiagram des Trainings der Modelle und Begründung dafür.

Die Ergebnisse der Metriken der Modelle dieses Teilexperiments sind in Tab. 2 zu finden.

Tab. 2: Ergebnisse der Metriken der Modelle mit Endsegment 2

Trainingsmenge / Lernrate	Lernrate 1
Bildmenge 1.1	Modell 14 Accuracy (Train): 78%, Loss(Train): 0,65, Top-1 Accuracy: 71%, Top-5 Accuracy: 96%, Precision: 75%, Recall: 71%, F1-Score: 70%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 2%, Top-5 Acc.: 11% CN (3.2): Top-1 Acc.: 5%, Top-5 Acc.: 15%
Bildmenge 1.2	Modell 15 Accuracy (Train): 79%, Loss(Train): 0,65, Top-1 Accuracy: 72%, Top-5 Accuracy: 96%, Precision: 76%, Recall: 72%, F1-Score: 72%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 1%, Top-5 Acc.: 9% CN (3.2): Top-1 Acc.: 2%, Top-5 Acc.: 6%
Bildmenge 1.3	Modell 16 Accuracy (Train): 79%, Loss(Train): 0,69, Top-1 Accuracy: 72%, Top-5 Accuracy: 96%, Precision: 75%, Recall: 72%, F1-Score: 72%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 1%, Top-5 Acc.: 4% CN (3.2): Top-1 Acc.: 4%, Top-5 Acc.: 13%

Trainingsmenge / Lernrate	Lernrate 1
Bildmenge 2.1	Modell 17 Accuracy (Train): 84%, Loss(Train): 0,53, Top-1 Accuracy: 84%, Top-5 Accuracy: 98%, Precision: 83%, Recall: 84%, F1-Score: 83%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 25%, Top-5 Acc.: 52% CN (3.2): Top-1 Acc.: 25%, Top-5 Acc.: 50%
Bildmenge 2.2	Modell 18 Accuracy (Train): 88%, Loss(Train): 0,41, Top-1 Accuracy: 87%, Top-5 Accuracy: 98%, Precision: 87%, Recall: 87%, F1-Score: 87%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 19%, Top-5 Acc.: 50% CN (3.2): Top-1 Acc.: 22%, Top-5 Acc.: 48%
Bildmenge 2.3	Modell 19 Accuracy (Train): 87%, Loss(Train): 0,42, Top-1 Accuracy: 86%, Top-5 Accuracy: 99%, Precision: 86%, Recall: 86%, F1-Score: 86%, Overfitting: sehr niedrig CN (3.1): Top-1 Acc.: 20%, Top-5 Acc.: 44% CN (3.2): Top-1 Acc.: 24%, Top-5 Acc.: 49%

Modell 14 – 19 Für das Endsegment 2 (siehe Abb. 27) wurden insgesamt sechs Modelle jeweils auf einer Bildmenge der Datensätze 1 und 2 trainiert. Das Training umfasste insgesamt 10 Epochen mit Lernrate 1 (siehe Abb. 21). Die Ergebnisse der einzelnen Modelle sollen hier ebenfalls summarisch zusammengefasst werden. Das Modell 14 erreicht mit einer Top-1 Accuracy von 71% im Nachtest eine schlechtere Leistung als beispielsweise Model 8, welches auf derselben Bildmenge trainiert wurde. Auffällig ist hier, dass die Accuracy auf dem Trainingsdatensatz am Ende von Epoche zehn den Wert von 89% nicht überschreitet (siehe Abb. 37 Links). Der Loss Wert sinkt während des gesamten Trainings kontinuierlich, was bei allen Modellen mit Endsegment 1 mit den drei angepassten Lernraten nicht erreicht werden konnte (siehe Abb. 37 Rechts). Die zugehörige Confusion Matrix zeigt abseits der Diagonalen entsprechende der schlechteren Werte der Metriken etwas mehr Masse. Ein generelles Überanpassungsproblem kann aber auch hier nicht ausgemacht werden. Die Probleme mit den Familiendynastien bleiben ebenso erhalten (siehe Abb. 56).

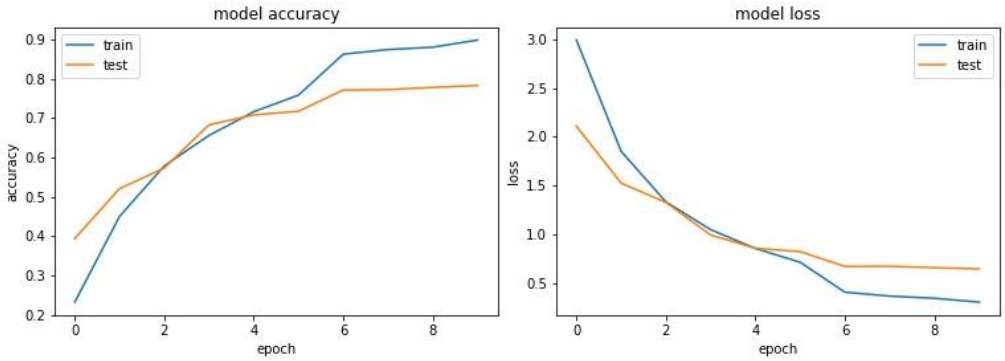

Abb. 37: Links: Verlauf der Accuracy von Modell 14 während des Trainings; Rechts: Verlauf des Loss von Modell 14 während des Trainings.

Die Modelle 15 und 16 ändern an dem bei Modell 14 beschriebenen Bild wenig, da die Werte der Metriken auf demselben Niveau verharren. Erst mit dem größeren Datensatz 2 schließen die Werte jetzt an die der Modelle mit Endsegment 1, welche auch auf Datensatz 2 trainiert wurden, an. Modell 17 erreicht die sehr gute Werte von Modell 11 und kann es bei den Loss Werten mit 0,53 sogar noch unterbieten. Auch das Modell 18 mit den normalen und den gespiegelten Porträts in der Trainingsmenge erreicht mit einem Loss Wert von 0,41 den niedrigsten Wert aller Modelle in diesem Experiment. Das letzte Modell (19), welches auf den Porträts mit der quadratischen Form trainiert wurde, kann sämtliche Werte von Modell 13 auf den Testbildern der Bildmenge 2.3 erreichen und den Loss Wert sogar deutlich unterbieten. Was jedoch bei den Modellen 17 – 19 auffällt ist, dass sie die Werte ihrer Pendants auf den Datensatz 3 mit den CN Bildern nicht ganz erreichen können. Nur ein Modell (17) kann auf die 50% Marke bei der Top-5 Accuracy vordringen. Fünf Modelle (6, 7, 11, 12 und 13) mit

Endsegment 1 konnten diese Hürde nehmen. Das Modell 19 schlägt jedoch in einem Bereich das Modell 13 deutlich. Auf den Bildern mit vollständigen Münzen erreicht es eine Top-1 Accuracy von 37% und eine Top-5 Accuracy von 67% (Modell 13: Top-1: 34% und Top-5: 58%). und zeigt somit einen etwas besseren Umgang mit diesen vollständigen Münzen.

Fazit Die Speicherplatz-effizienteren Modelle mit dem Endsegment 2 konnten in diesem Test mit ihren Pendants mit Endsegment 1, wenn der größere Datensatz 2 als Grundlage des Trainings herangezogen wurde, weitestgehend bei den Metriken mithalten und diese bei den Loss Werten sogar unterbieten. Jedoch zeigte es sich, dass die Modelle mit Endsegment 2 auf den CN Daten etwas unterhalb der Modelle mit Endsegment 1 rangieren. Daher können die Modelle mit Endsegment 1 als etwas geeigneter für einen Gebrauch auf anderen Datensätzen angesehen werden. Auf vollständigen Münzen liefere das Modell 19 die bisher beste Leistung. Insgesamt gesehen kann gesagt werden, dass die sehr speicherplatzintensive Endstruktur 1 nicht nötig ist, um mit den entsprechenden Daten ein sehr gutes Modell zu trainieren. Die wesentlich schlankere Endstruktur 2 zeigt dazu während des Trainings keine Zeichen von Überanpassung und es waren keinerlei Anpassungen der Lernrate nötig. Daher stellen diese Modelle gute Alternativen dar, wenn es auf die Größe und Ladezeit eines Modells ankommt.

g) Resümee des zweiten Experiments

Dieses Experiment sollte die Auswirkungen von zwei unterschiedlichen Endstrukturen des VGG16 Netzes, von unterschiedlichen Epochenanzahlen und Bildmengen auf die Werte der einzelnen Metriken untersuchen. Dabei wurden die folgenden Erkenntnisse gewonnen:

- Ein Training auf dem ausgeschnittenen Porträtbereich einer Münze kann sehr gute Ergebnisse bringen. Eine größere Trainingsmenge kann die Werte der Metriken noch verbessern.
- Mit einer Spiegelung von nach rechts gerichteten Porträts kann ein Modell die selteneren nach links ausgerichteten Porträts besser vorhersagen, wenn das vorhandene Trainingsmaterial dafür nicht ausreichend ist.
- Die Verzerrung bei der Umwandlung des langrechteckigen Porträtbereiches in ein quadratisches Format spielt eine Rolle bei der Anwendung auf Bilder vollständiger Münzen. Die Korrektur durch das Hinzufügen von schwarzen Balken zu einem quadratischen Format wirkt sich jedoch nicht auf die Metriken aus.
- Die Modelle mit beiden Endstrukturen haben sich als geeignet zur Identifizierung von Porträts erwiesen. Die Modelle mit Endstruktur 2 benötigen eine Trainingsdauer von zehn Epochen, während die Modelle mit Endstruktur 1 mit sechs Epochen und

angepasster Lernrate auskommen, um somit die Gefahr einer Überanpassung zu vermindern.

- Die Modelle 13 (Endstruktur 1) und 19 (Endstruktur 2) wurden anhand ihrer Metriken als jeweils beste Modelle bewertet. Den Top-1 Accuracy Wert des Modells von A. Loyal (91%) kann keines von beiden erreichen, aber ihren Top-5 Wert von 92% können beide mit ca. 99% überbieten.

Im nächsten Experiment wird das Vorhersageverhalten von Modell 13 genauer untersucht, da dieses Modell in Zusammenarbeit mit der Porträtbereichserkennung sich etwas besser auf andere Datensätze wie die CN Münzen generalisieren lässt. Jedoch ist Modell 19 eine fast gleichwertige Alternative, wenn es auf die Speicherplatzgröße und Ladezeit wie z.B. in Webanwendungen ankommt.

5.3 Experiment 3: Test und Auswertung des finalen Modells

Das letzte hier ausgeführte Experiment widmet sich der Auswertung des Modells 13, welches sich in Experiment 2 mittels der Metriken auf den verschiedenen Datensätzen als das zur Übertragung auf andere Datensätze etwas geeigneter von den zwei besten Modellen gezeigt hat. Zu Anfang soll ein direkter Vergleich der Metriken mit A. Loyals finalen Modell zur Kaisererkennung gezogen werden. Dabei soll besonders auf die Klassen eingegangen werden, deren Werte sich durch das Entfernen der Legende am stärksten verändert haben. Anschließend sollen die Ergebnisse des Modells auf den Porträts des Datensatzes 3 mit den CN Daten durch das Erstellen von Heatmaps weiter analysiert werden und mögliche Gründe für das schlechtere Abschneiden von Modell 13 bei diesen Bildern gefunden werden. Im Zuge dessen soll auch auf die Problematik des Umgangs des Modells mit Fotos von Münzabgüssen aus Gips eingegangen werden. Den letzten Teil des Experiments bildet eine Auswertung des Modellverhaltens auf den Bildern von vollständigen Münzen. Hier sollen besonders die Elemente der Münzen identifiziert werden, die das Modell möglicherweise von dem eigentlichen Porträt ablenken und somit falsche Vorhersagen provozieren.

a) Vergleich mit dem finalen Modell von A. Loyal

Das Kaisererkennungsmodel, das in der Masterarbeit von A. Loyal am besten abschnitt, konnte eine Top-1 Accuracy von 91% und eine Top-5 Accuracy von 92% auf den vollständigen Münzen ihrer Testmenge erreichen. Das hier am besten bewertete Modell 13 erreicht einen Top-1 Accuracy Wert von 86% und einen Top-5 Accuracy Wert von 99% auf der größeren Testmenge 2.3 mit den ausgeschnittenen Kaiserporträts. Die Werte von Precision, Recall und

F1-Score liegen jeweils um ca. 6% tiefer als die des Modells von A. Loyal[147]. Dieses Absinken kann besonders an neunzehn Klassen festgemacht werden, deren Werte sich gegenüber ihrer Auswertung verschlechtert haben. Zu diesen Klassen gehören insbesondere die Mitglieder der konstantinischen, der theodosianischen und der valentinianischen Dynastie[148]. Daneben sind auch die Mitglieder der sog. Tetrarchie für diese Verschlechterung verantwortlich[149]. Jedoch gibt es auch eine merkliche Verbesserung der Werte bei drei Mitgliedern des judisch-claudischen Kaiserhauses[150]. Die restlichen Klassen können ihre Ergebnisse mit kleineren Schwankungen nach unten oder oben halten (siehe Abschn. 9.3 Auflistung 2). Diese Dynastie-bezogenen schlechteren Ergebnisse von Modell 13 können mit der Ähnlichkeit der einzelnen Familienporträts untereinander und das abstraktere mit weniger individuellen Merkmalen versehen kaiserliche Porträt des 4. und 5. Jhs. n. Chr. erklärt werden[151]. Hierbei war die Legende der ursprünglichen kompletten Münzbilder ein wichtiges Identifizierungsmerkmal, welches auch von Numismatikern herangezogen wird[152]. Durch die Entfernung der Legende fiel dieses Merkmal weg und konnte auch nicht durch die Steigerung der Anzahl der Trainingsbilder der entsprechenden Klassen kompensiert werden. Somit wird ein Erreichen von besseren Ergebnissen beim Training der Modelle durch inhärente Eigenschaften des spätantiken Kaiserporträts verhindert.

b) Einsatz auf CN Münzen

Auf dem Datensatz 3 mit Münzporträts, die mittels des Corpus Nummorum Portals gefunden wurden, erlangt das hier besprochene Modell 13 auf der größeren Bildmenge 3.2 mit etwa 9600 Münzen eine Top-1 Accuracy von ca. 26 % und eine Top-5 Accuracy von ca. 54 %. Diese Werte weichen gegenüber den Testbildern der Menge 2.3 stark ab (siehe Tab. 1). Daher stellt sich die Frage, wie man diese Unterschiede erklären kann. Ein erster Vergleich der Bildmengen zeigt, dass in den CN Daten lediglich 30 der 66 trainierten Klassen mit Kaiserporträts vertreten sind[153]. Ein Blick auf die vorhandene Auswahl der Klassen lässt erkennen, dass vorwiegend

[147] Loyal 2018, 74–76 mit Tab. 6.1.
[148] Betroffene Mitglieder des konstantinischen Kaiserhauses: Constans**,** Konstantin II.und Constantius I. (Constantius Chlorus, auch Mitglied der Tetrarchie). Betroffene Mitglieder des theodosianischen Kaiserhauses: Arcadius, Honorius und Theodosius I.). Betroffene Mitglieder des vantianischen Kaiserhauses: Gratian, Valens, Valentinian I. und Valentianian II.
[149] Tetrarchen mit schlechteren Werten: Constantius I. (Constantius Chlorus, auch Mitglied der konstantinischen Dynastie), Diocletian, Galerius, Licinius II., Maximian und Maximinus Daia.
[150] Diese drei Mitglieder sind: Tiberius, Claudius und Nero.
[151] Siehe zum Kaiserbild des 4. und 5. Jhs. n. Chr.: Peschlow 1983.
[152] Auch bei A. Loyal konnte bei einer Heatmap ein Fokus ihres Modells auf die Legende einer Münze mit der Darstellung des Valentinian I. beobachtet werden. Loyal 2018, 52. Abb. 4.10.
[153] In den CN Daten vorhandene Kaiser: Antoninus Pius, Augustus, Caracalla, Claudius, Commodus, Domitian, Elagabal, Gallienus, Geta, Gordian III., Hadrian, Lucius Verus, Maximinus Thrax, Nero, Nerva, Philippus

Kaiser und Kaiserinnen des 1. und 2. Jhs. n. Chr. vertreten sind. Die häufig wechselnden Herrscher des 3. Jhs. n. Chr. sind nur mit zehn Kaisern vertreten. Der zeitlich gesehen letzte Kaiser in den Daten ist Gallienus, der bis 268 n. Chr. herrschte. Damit fallen die bereits geschilderten Probleme des Modells mit den Familiendynastien und den Tetrarchen als Grund für das schlechtere Abschneiden auf diesen Münzen weg, da die zugehörigen Kaiser alle ab dem Ende des 3. Jhs. n. Chr. bis in das 5. Jh. n. Chr. regierten.

Gipsabgüsse Eine mögliche Erklärung wäre die in der Bachelor Arbeit des Autors bereits geschilderte These, dass die in den CN Bildern vorhandenen Gipsabgüsse von Münzen für das schlechtere Abschneiden verantwortlich wären. Abgüsse sind Kopien einer originalen Münze und somit möglicherweise fehlerhaft. Vor allem könnte der Abrieb der Umrisse und Kanten des Porträts, welcher bei einem im Gegensatz zu Metall deutlich weicheren Material wie Gips stärkere Auswirkungen hat, das Modell vor größere Probleme stellen. Auch andere Fehler, die beim Herstellen des Abgusses oder durch unsachgemäße Handhabung bei der Aufbewahrung passiert sind, könnten einen Einfluss haben[154]. Und tatsächlich enthält die Bildmenge 3.2 zu ca. 48 % Gipsabgüsse von Münzen. Um diese These nun testen zu können, wurden die Porträts dieser Bildmenge in Abgüsse und Originale aufgeteilt und dann entsprechend dieser Trennung dem Modell nacheinander zugeführt. Das Ergebnis dieses Tests konnte die Annahme der Abgussprobeme jedoch nicht bestätigen. Auf den Originalmünzen erreichte das Modell eine Top-1 Accuracy von 29% und eine Top-5 Accuracy von 57%. Der Test auf den Abgüssen erbrachte dagegen einen Top-1 Wert von 23% und einen Top-5 Wert von 51% (siehe Abschn. 9.3 Auflistung 4). Somit ist dieser Unterschied in dem Material der fotografierten Porträts für die schlechteren Ergebnisse auf den CN Bildern nicht signifikant.

Die Fotos der Abgüsse sind noch einmal aufgeteilt in Bilder, die mit einer Fotostation (56 % der Daten) oder einem normalen Scanner (44% der Daten) gemacht wurden. Wenn man diese noch einmal separat auf Modell 13 testet, dann erhält man für die Scannerbilder folgende Werte[155]: Top-1 Accuracy: 16%, Top-5 Accuracy: 44%. Bei dem Einsatz auf Bilder der Fotostation konnte man dagegen diese Werte sehen: Top-1 Accuracy: 25%, Top-5 Accuracy: 53%. Somit reichen die Bilder von der Fotostation sogar sehr nahe an die Werte der Originalmünzen heran. Das schlechtere Abschneiden der Scannerbilder liegt wohl daran, dass sie verwaschener wirken als die Bilder der Fotostation (vgl. Abb. 38 Links und Rechts).

Arabs, Philippus II., Septimius Severus, Severus Alexander, Tiberius, Titus, Trajan, Trebonianus Gallus und Vespasian.

[154] Gampe 2019, 32.

[155] Dieser Test fand auf einer auf etwa 4100 Bilder (von ca. 4560 Bildern) reduzierten Menge an Abgussbildern statt, da bei sechs Klassen die Bilder nicht mehr nach ihrer Herstellungsart separiert werden konnten.

Abb. 38: Links: Aufnahme eines von Abrieb gezeichneten Abgussporträts des Domitian mit der Fotostation; Rechts: Scan eines abgeriebenen Porträts des Domitian. Trotz des sogar schlechteren Zustands des linken Porträts gelingt es der Fotostation besser die Kanten des Porträts hervorzuheben.

Die Fotostation kann die Kanten aufgrund der pointierteren Darstellung von Schattenverläufen deutlicher darstellen. Ohne die Bilder, die mit dem Scanner hergestellt wurden, würde man somit kaum einen Unterschied zwischen Bilder von originalen Münzporträts und den Abgüssen messen können. Der bestehende Unterschied von 6 % von Originalen zu Abgüssen auf beiden Werten rührt also hauptsächlich von den gescannten Abgussporträts her. Jedoch kann diese Beobachtung die schlechteren Ergebnisse auf den CN Daten nicht wirklich erklären.

Um andere mögliche Gründe für das schlechtere Abschneiden zu finden, wurden mehrere Serien von Heatmaps mit dem Grad-CAM Technik (siehe Abschn. 3.5) erstellt. Die zugrundeliegende Idee war es hier die Heatmaps von Klassen auszuwerten, deren Werte bei den Metriken auf den Testbildern von Bildmenge 2.3 und den Porträts von Bildmenge 3.2 eine besonders große Differenz zeigen. Dabei soll auch der Erhaltungszustand der Münzen betrachtet werden, da dieser sich ebenfalls auf die Werte einer Klasse auswirken kann, wie bereits in der Bachelorarbeit des Autors beobachtet wurde[156]. Nötig ist außerdem eine aussagekräftige Menge an Bildern. Beispielsweise ist die Klasse „Titus“ lediglich mit sechs Bildern in der Bildmenge 3.2 vertreten. Gleichzeitig weist sie einen F1-Score von null Prozent auf, während das Modell auf den Testbildern dieser Klasse aus Menge 2.3 einen F1-Score von 0,78 bei 309 Bildern erreicht. Doch ist die Anzahl an Bildern im CN Datensatz zu gering, um mehr als oberflächliche Aussagen treffen zu können. Daher wurde entschieden, dass vorwiegend Klassen mit einer größeren Anzahl an Bildern in beiden Vergleichsmengen zur Auswertung herangezogen werden.

Philippus II. Den auffälligsten Befund auf den CN Daten liefert *Philippus II.* In der Testmenge erreicht die Klasse mit ca. 71% Top-1 und 100% Top-5 Accuracy bei 84 Testbildern ein gutes Ergebnis. Jedoch erlangt das Modell 13 auf den CN Daten bei dieser Klasse eine Top-1 Accuracy von 0% und eine Top-5 Accuracy von ca. 8% bei 65 Bildern. Die falschen Top-1 Ergebnisse der OCRE Bilder zeigen, dass die Top-1 Ergebnisse recht nah an der wirklichen

[156] Gampe 2019, 29–31.

Klasse liegen, da hier bei 24 falschen Bildern sechsmal der Vater von Philippus II. Philippus Arabs vorkommt und insgesamt elfmal Gordian III., der Vorgänger seines Vaters im Amt des römischen Kaisers war. Weiterhin ist viermal Severus Alexander vertreten, der jedoch etwa 20 Jahre früher herrschte. Bei allen drei Klassen ist also eine zeitliche Nähe zu Philippus II. gegeben, jedoch unterscheidet sich ihre Porträtdarstellungen bei der Barttracht. Während Phillipus II. und Gordian III. auf den OCRE Münzen bartlos dargestellt werden, hat Philippus Arabs immer und Severus Alexander oft einen Kinnbart. Bei den 65 falsch vorhergesagten CN Bildern kommt der Vater Philippus II. und Severus Alexander lediglich zweimal und Gordian III. sechsmal als Top-1 Ergebnis vor. Die dominierende Klasse ist hier der zeitlich in keinem Zusammenhang stehende Augustus mit acht Nennungen. Nach der Erstellung von Heatmaps zu beiden Bilderserien fällt auf, dass Philippus II. auf den OCRE Daten vorwiegend an der kaiserlichen Strahlenkrone (sofern vorhanden) den Gesichtspartien und dem bartlosen Kinn erkannt wird (siehe Abb. 39 Links). Die CN Heatmaps zeigen dagegen, dass hier von dem Modell weit öfter auf andere Teile des Kopfes, beispielsweise die Regionen im Bereich des Ohres geachtet wird. Das Kinn spielt hier meist in Verbindung mit der Unterkieferpartie eine Rolle (siehe Abb. 39 Mitte). Sofern die Strahlenkrone vorkommt, wird sie auch in die Erkennung einbezogen, aber auch zusammen mit der Kinnpartie kann sie bei vier Fällen nicht zu dem richtigen Top-1 Ergebnis führen (siehe Abb. 39 Rechts). Lediglich in einem der vier Fälle ist Philippus II. wenigstens in den Top-5 Vorhersagen an vierter Stelle mit einer Wahrscheinlichkeit von ca. 3% vertreten. Der Münzzustand spielt ebenfalls eine wichtige Rolle: Zehn der 65 Bilder werden von sehr abgeriebenen Abgüssen eingenommen, bei welchen die Kanten mit dem Auge nur noch schwer zu erkennen sind. Auch die Originalmünzen sind in einem deutlich schlechteren Zustand als diejenigen aus der OCRE Datenbank, da sie in vielen Fällen von Korrosion oder starkem Abrieb betroffen sind (mindesten 34 der 52 Bilder von Originalmünzen, siehe Abb. 39).

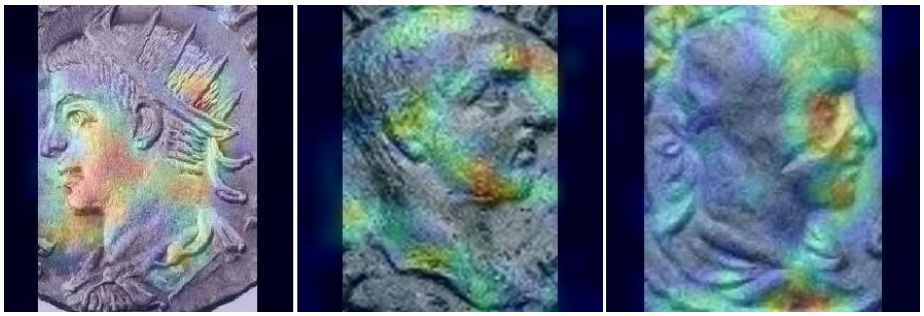

Abb. 39: Links: Heatmap des Typus RIC IV Philip I 240 (gespiegelt)[157]; Mitte: Heatmap des Typus CN Type 1622; Rechts: Heatmap des Typus CN Type 1272 (falsch klassifiziert).

[157] Dargestellt auf dieser Münze des Philippus Arabs ist sein Sohn Philippus II.

Diese Beschädigungen zusammen mit kleinen Abweichungen bei der Darstellung des Kaisers in dem regionalen Stil des CN Gebietes führen wohl dazu, dass oft andere Partien des Porträts von dem Modell zur Identifizierung des Kaisers herangezogen wurden und letztendlich damit zu den sehr schlechten Ergebnissen in den herangezogenen Metriken.

Severus Alexander Eine andere Klasse, die ebenfalls große Unterschiede in den Metriken von OCRE und CN Bildern zeigt, wird durch dem Kaiser *Severus Alexander* verkörpert. Auf den Testdaten der Bildmenge 3.2 werden von dem Modell 13 mit einer Top-1 Accuracy von 95% und eine Top-5 Accuracy von 99% sehr gute Werte bei 535 Bildern erreicht. Die Werte auf den CN Bildern von Bildmenge 3.2 fallen dagegen wieder stark ab: 13% Top-1 Accuracy und 33% Top-5 Accuracy bei 563 Bildern. Auch hier sind einige der falschen Top-1 Ergebnisse auf den OCRE Daten zeitlich wieder recht nah an der tatsächlichen Klasse. Es sind bei 25 falschen Ergebnissen fünfmal Elagabal, der Vorgänger und Vetter des Severus Alexander, viermal der spätere Kaiser Philippus II. und zweimal Philippus Arabs vertreten. Bei den CN Daten zeichnet sich ein anderes Bild. Bei den falschen Top-1 Ergebnissen ist mit 53 Zuweisungen einer der Vorgänger des Severus Alexander Kaiser Caracalla die dominante Klasse. Da sich Severus Alexander durch zu der Dynastie der Severer zugehörig fühlte, ist eine Ähnlichkeit seines Porträts mit dem von Caracalla nicht verwunderlich. Von beiden Kaisern existieren Porträts in unterschiedlichen Alter (jugendlich ohne Bart und erwachsen mit Kinnbart) auf Münzen, die auch in den Trainingsdaten von Bildmenge 2.3 vorhanden sind. Die Klasse mit den zweitmeisten Nennungen verwundert dagegen umso mehr. Es ist der Kaiser Augustus mit 50 Vorhersagen als wahrscheinlichstes Ergebnis. Es folgen Philippus II. mit 26, Elagabal mit 23 und Gordian III. mit zwölf Nennungen, die aber im Gegensatz zu Augustus in einem zeitlichen Zusammenhang mit Severus Alexander stehen. Die korrekt erkannten Porträts aus den OCRE Daten werden laut den Heatmaps vorwiegend am Haarbereich mit dem Lorbeerkranz und der Kinnzone mit dem kurzen Bart erkannt (siehe Abb. 40 Links). Bei den bartlosen Porträts im jugendlichen Alter findet die Identifizierung wiederrum hauptsächlich im Bereich der Haare mit dem Lorbeerkranz statt und wird in einigen Fällen mit Partien aus dem Gesicht ergänzt, wobei auch das bartlose Kinn sehr oft eine Rolle spielt. Dieselben Zonen können auch bei den korrekt vorhergesagten Porträts der CN Daten aus Bildmenge 3.2 ausgemacht werden (siehe Abb. 40 Mitte links), sofern das Porträt in einem guten Zustand ist. Bei stärker durch Abnutzung oder Korrosion beschädigten Münzen kann auch die zu einer Schleife zusammengebundene Schnüre des Lorbeerkranzes als ein Identifizierungsmerkmal dienen (siehe Abb. 40 Mitte rechts). Dies war bei den OCRE Daten, die meist deutlich besser erhalten waren, weitaus seltener zu beobachten. Bei den falsch klassifizierten Porträts der CN

Bilder zeigt sich bei den wichtigen Bereichen auf den Heatmaps ein deutlich diffuseres Bild als bei den OCRE Daten. Es werden weitaus mehr Bereiche des Porträts miteinbezogen und teilweise sind die Wärmezonen wesentlich großflächiger als bei den Pendants der OCRE Bilder. Dies fällt vor allem bei den Porträts auf, die irrtümlich als Augustus klassifiziert wurden. Dies ist aber ebenso bei anderen Klassen in geringerem Umfang möglich. Auch wird hier das Gewand oft miteinbezogen, was ebenfalls bei den OCRE Bildern viel seltener zu beobachten war (siehe Abb. 40 Rechts). Der Münzzustand wird wieder für die Ergebnisse eine tragende Rolle spielen, da dieser bei den CN Münzen durchgehend schlechter ist, als es bei den OCRE Münzen der Fall ist. Häufig sind Zonen wie Kinn und Haarbereich von starken Beschädigungen betroffen, so dass diese von dem Modell nicht zur Identifizierung verwendet werden können. Dies kann aufgrund der vorhandenen Daten auch auf die Metalle der Münzen zurückgeführt werden. Die Münzen mit dem Porträt des Severus Alexander aus der CN Datenbank bestehen alle aus Bronze, während die OCRE Stücke größtenteils aus Silber gefertigt sind. Im Vergleich zu Silber erleiden Bronzemünze durch Korrosion im Laufe der Zeit wesentlich stärkere Schäden (Abb. 40 Mitte rechts). Dadurch, dass Bronzegeld weit mehr als das höherwertige Silber für das Bezahlen alltäglicher Waren und Dienstleistungen verwendet wurde, ist eine Beschädigung durch Abrieb weitaus wahrscheinlicher (siehe Abb. 40 Rechts).

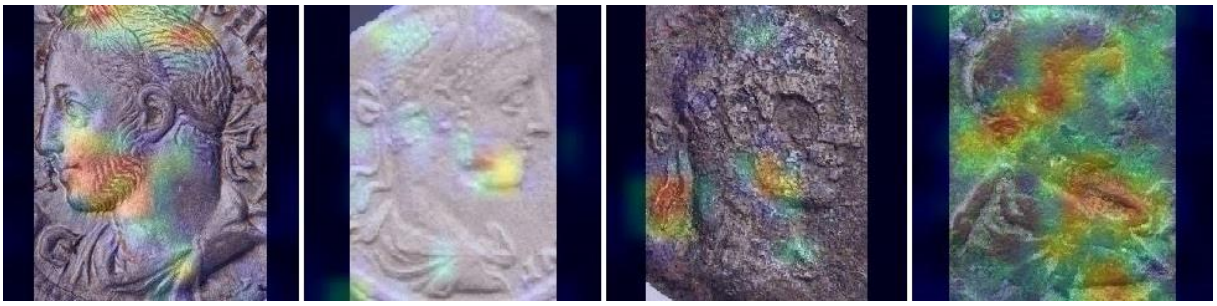

Abb. 40: Links: Heatmap des Typus RIC IV Severus Alexander 246c (gespiegelt); Mitte links: Heatmap des Typus CN Type 8507; Mitte rechts: Heatmap einer unveröffentlichten CN Münze; Rechts: Heatmap der Münze CN coin 7167 als Augustus klassifiziert.

Aber auch bei den Bronzemünzen der OCRE Datenbank ist der Erhaltungszustand weitaus besser einzuschätzen als der der CN Bronzemünzen. Somit wird auch bei dieser Klasse ein gewichtiger Anteil an den Ursachen der schlechten Ergebnisse des Modells 13 durch den metallbedingten Erhaltungszustand der Münzen und ihrer Porträts auszumachen sein. Daneben können kleinere Unterschiede bei dem Stil der Porträts, die durch das unterschiedliche Können der Stempelschneider, welche den Städten aus dem CN Gebiet zur Verfügung standen, mithineinwirken. Dies beides führt wohl dazu, dass sich das Modell laut den Heatmaps nicht auf die Identifizierungsmerkmale der OCRE Porträts konzentriert, sondern häufig andere Bereiche wählt, was zu den vielen falschen Klassifizierungen führt.

Hadrian *Hadrian* ist als Klasse ebenso von schlechten Werten des Modells 13 auf den CN Daten betroffen. Auf den OCRE Daten konnte das Modell noch mit einer Top-1 Accuracy von ca. 99% und einer Top-5 Accuracy von annähernd 100% hervorragend abschneiden. Dagegen zeigt die Performance auf den CN Daten mit einer Top-1 Accuracy von 30% und einer Top-5 Accuracy von 56% wieder große Probleme bei der Erkennung, obwohl die Klasse damit sogar leicht über den Werten für alle Klassen liegt (siehe Tab. 1). Die 31 falschen Top-1 Ergebnisse der OCRE Münzen sind hier teilweise wieder zeitlich nah an der gesuchten Klasse. Während die sechs Nennungen des Trajan, der als erster Kaiser seinen Nachfolger Hadrian adoptierte, und die sieben Identifizierungen als Antoninus Pius, dem adoptierten Nachfolger des Hadrian, zeitlich passend zu der richtigen Klasse sind, können die fünf Identifizierungen als Augustus keine Hilfe sein. Die 157 falsch klassifizierten Porträts der CN Daten liefern wieder ein schwerer durchschaubares Bild. Die beiden dominierenden Klassen der Top-1 Erkennung sind Antoninus Pius und Augustus mit jeweils 30 Nennungen. Somit stehen hier zwei Klassen im Fokus, die entweder zeitlich sehr nah (Antoninus Pius) oder über ein Jahrhundert entfernt (Augustus) von der gesuchten Klasse sind. Nach dem Sichten der Heatmaps der Testdaten von Bildmenge 2.3 können wieder einige Merkmale der Identifizierung der Klasse ausgemacht werden. Die dominierenden Elemente sind hier der Kinnbereich mit dem Vollbart und die Nasen- und Augenpartie (siehe Abb. 41 Links). Ebenfalls wichtig sind die Haare mit dem Lorbeerkranz als kaiserliche Insignie (sofern vorhanden) (siehe Abb. 41 Mitte links und Abb. 42 Mitte). Bei den richtig erkannten Porträts des CN Datensatzes bietet sich ein ähnliches Bild. Der Kaiser wird vorwiegend an Kinnbereich mit Bart und den Haaren erkannt. Die Nasen- und Augenpartie spielte eine geringere Rolle als auf den OCRE Bildern. Die falsch klassifizierten Porträts zeigen auf den Heatmaps in vielen Fällen ebenso das Kinn und die Haare als wichtigste Bereiche, die das Modell zur Identifizierung benutzt hat. Auffällig ist, dass wenn Kinn- und Haarpartie von starkem Abrieb betroffen sind, so dass vor allem der Bart kaum noch auszumachen ist, dann scheint eine Identifizierung als Augustus am wahrscheinlichsten zu sein, da dieser bartlos dargestellt wird (siehe Abb. 41 Mitte rechts). Dazu kommt, dass ebenso wie bei den Severus Alexander Porträts (s.o.) der Wärmebereich des Modells für eine Identifizierung als Augustus weitaus großflächiger ausfällt als bei anderen Klassen. Wenn beide Partien und der Augenbereich etwas besser erhalten sind, dann wird das Porträt eher als Antoninus Pius, dessen Porträt große Ähnlichkeiten bei Frisur und Bart zu

Hadrian aufweist, identifiziert (siehe Abb. 41 Rechts)[158]. CN Porträts guter Qualität, die sich eng an den dominierenden Darstellungstypus in der OCRE Datenbank anlehnen, können von dem Modell noch gut erkannt werden (siehe Abb. 42 Links, vlg. mit Abb. 41 Mitte links). Porträts in gutem Erhaltungszustand, die darstellungstechnisch zu OCRE Porträts nur noch relativ ähnlich sind und sich bei Details der Wiedergabe des Haares und des Bartes unterscheiden, werden bei den CN Daten, obwohl der Fokus auf Bart und Haaren liegt, oftmals nicht als Hadrian erkannt (vgl. Abb. 42 Mitte mit Abb. 42 Rechts).

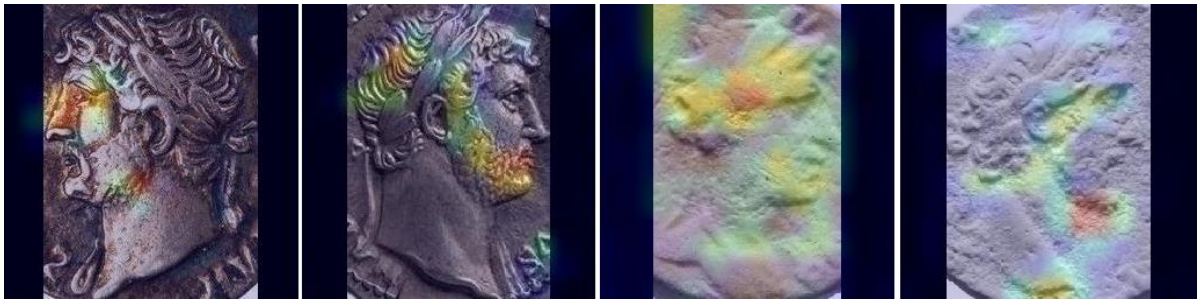

Abb. 41: Links: Heatmap des Typus RIC II, Part 3 (second edition) Hadrian 798; Mitte links: Heatmap des Typus RIC II, Part 3 (second edition) Hadrian 857; Mitte rechts: Heatmap des Typus CN Type 4726, als Augustus klassifiziert; Rechts: Heatmap eines unveröffentlichten CN Abgusses, als Antoninus Pius klassifiziert.

Abb. 42: Links: Heatmap einer unveröffentlichten CN Münze Mitte: Heatmap des Typus RIC II, Part 3 (second edition) Hadrian 141-143; Rechts: Heatmap einer unveröffentlichten CN Münze, als Caracalla klassifiziert.

Die Münzen des CN Datensatzes beinhalten wiederum hauptsächlich Abgüsse und Münzen aus Bronze und weisen durchweg stärkere Abnutzung und Beschädigung als ihre OCRE Pendants auf. Daher wird hier ebenfalls ein nicht geringer Anteil der falschen Klassifizierungen auf diese Beschädigungen zurückzuführen sein. Die Details (Features) der Hadrian Porträts in den CN Daten scheinen oft nicht deutlich genug erhalten zu sein. Diese Details werden jedoch vom Modell in den darstellungstechnisch sehr homogenen Trainingsdaten mit meist gut erhaltenen Münzen des OCRE Datensatzes gelernt und werden als Grundlage der Identifizierung des

[158]Übersicht zu den verschiedenen Münzporträts des Antoninus Pius in der OCRE Datenbank: Online Coins of the Roman Empire. Antoninus Pius, <http://numismatics.org/ocre/results?q=portrait_facet%3A%22Antoninus+Pius%22> (15.10.2021).

Kaisers herangezogen[159]. Die stilistischen Unterschiede in den verschiedenen Stadtprägungen des CN Gebietes im Vergleich zu den offiziellen Reichsprägungen der OCRE Daten erschweren es dem Modell wohl zusätzlich die richtigen Ergebnisse bei der Klassifikation zu finden.

Augustus Eine Klasse mit deutlich besseren Werten auf der CN Bildmenge 3.2 ist dem ersten Kaiser *Augustus* zugeordnet. Die Performance des Modells 13 auf den OCRE Testdaten ist mit einer Top-1 Accuracy von 93 % und einer Top-5 Accuracy von 99% sehr gut. Auf den CN Porträts kann das Modell hier immerhin mit einer Top-1 Accuracy von 64% und einer Top-5 Accuracy von 94% gute Werte erreichen und somit deutlich besser abschneiden als die Durchschnittswerte des Modells auf allen Klassen (siehe Tab. 1). Von den 61 falsch Klassifizierten Bildern der OCRE Testdaten wurden 14 als Tiberius, der Nachfolger des Augustus, identifiziert. Die anderen Falschklassifizierungen verteilen sich hauptsächlich auf Kaiser des 1. Jhs. n. Chr., so dass das Modell hier zeitlich recht nahe an der wirklichen Klasse liegt. Auch bei den 92 falsch vorhergesagten CN Porträts ist die Klasse Tiberius mit 15 Nennungen vor Nero mit 11 Identifizierungen, die beide Mitglied des von Augustus gegründeten julisch-claudischen Kaiserhauses sind, führend. Die anderen falschen Top-1 Klassen verteilen sich auf die Kaiser des ersten bis dritten Jahrhunderts. Was bei den Klassen Severus Alexander und Hadrian für die Identifizierung als August bereits beobachtet wurde (s.o.) ist der deutlich großflächigere über das Porträt verteilte Wärmebereich der Heatmaps, der auch an vielen Bildern der korrekt erkannten OCRE Testdaten ausgemacht werden kann. Dieser kann sich über alle Elemente des Porträts verteilen (siehe Abb. 43 Links). Ansonsten dienen die Gesichtspartie und die Haare oft als Identifizierungsmerkmale (siehe Abb. 43 Mitte links). Was beim Sichten der OCRE Daten auffiel ist, dass hier deutlich mehr abgegriffene und durch Korrosion beschädigte Münzen in der Bildmenge vorhanden sind als bei den oben besprochenen Klassen (wenigstens 50% der Trainings- und Testdaten des Augustus)[160]. Was wohl auch darauf zurückzuführen ist, dass hier ein recht ausgewogenes Verhältnis von Silber- zu Bronzemünzen herrscht. Jedoch sind auch nicht wenige Silbermünzen durch häufige Verwendung stärker abgegriffen (siehe Abb. 43 Mitte rechts). Weiterhin unterliegen auch die Porträts einer stilistisch stärker differenzierten Darstellung (vgl. Abb. 43 Links mit Abb. 43 Mitte links). Es ist außerdem zu beobachten, dass die großflächigen Wärmebereiche häufiger auf den stärker beschädigten Münzen zu finden sind (siehe Abb. 43 Links).

[159] Übersicht zu den verschiedenen Münzporträts des Hadrian in der OCRE Datenbank: Online Coins of the Roman Empire. Hadrian, <http://numismatics.org/ocre/results?q=portrait_facet%3A%22Hadrian%22> (16.10.2021).

[160] Übersicht zu den verschiedenen Münzporträts des Augustus in der OCRE Datenbank: Online Coins of the Roman Empire. Augustus, <http://numismatics.org/ocre/results?q=portrait_facet%3A%22Augustus%22> (16.10.2021).

Abb. 43: Links: Heatmap des Typus RIC I (second edition) Augustus 230; Mitte links: Heatmap des Typus RIC I (second edition) Augustus 1A; Mitte rechts: Münze des Typus RIC I (second edition) Augustus 1A; Rechts: Heatmap einer unveröffentlichten CN Münze.

Die wichtigen Bereiche auf den korrekt klassifizierten CN Heatmaps unterscheiden sich kaum von den der korrekten ORCE Testmünzen. Jedoch ist hier die Identifizierung durch großflächige Wärmebereiche dominant (etwas mehr als 40% der Heatmaps). Die nicht korrekt erkannten CN Porträts zeigen hierbei ein anderes Bild. Die großflächigen Wärmbereiche machen nur noch etwas mehr als 20% der Heatmaps aus. Auf den anderen Heatmaps verteilen sich die wichtigen Zonen auf Hals, Haar- und seltener den Gesichtsbereich (siehe Abb. 43 Rechts). Auch diese Porträts sind wiederrum von Beschädigungen durch häufige Nutzung und Korrosion betroffen. Die meisten Münzen dieser Klasse in den CN Daten wurden hier ebenfalls aus zu schlechterer Erhaltung neigender Bronze geschlagen. Diese hier und bei den anderen Klassen gemachte Beobachtungen lassen den folgenden Schluss zu: Durch die OCRE Trainingsdaten der Klasse Augustus mit ihrem großen Anteil an beschädigten Silber- und Bronzemünzen ist das Modell 13 für die ebenfalls von stärkeren Beschädigungen geprägten CN Porträts dieser Klasse besser in der Lage die korrekte Klasse vorherzusagen. Das Modell hat gelernt die Augustus Porträts an deutlich größeren Bereichen und nicht nur einigen Details zu erkennen. Weiterhin kann das Modell durch die deutlich vom Stil her disparateren OCRE Porträts ebenfalls besser mit den durch unterschiedliche Herstellungsorte bedingten lokaleren Stil der CN Porträts umgehen. Dies führt aber nun zu einer Art von „Bias“ bei beschädigten CN Münzen. Die oben festgestellten häufigen Fehlklassifizierungen als Augustus auf Porträts anderer Kaiser werden wohl damit zusammenhängen, dass das Modell 13 stärker beschädigte Münzen, an denen Details wie der Bart kaum noch zu erkennen sind, mit einer größeren Wahrscheinlichkeit durch großflächigen Wärmezonen als Augustus erkennt. Dies kann auf die fehlende Ausgewogenheit der Trainingsdaten der drei oben betrachteten Kaiser zurückgeführt werden. Diese besitzen deutlich mehr Porträts in gutem Zustand bei denen das Porträt vom Stil her relativ uniform ist, so dass das Modell deren gut erhaltene Details (Features) als Merkmale lernen kann und somit wohl bei den schlechter erhaltenen Münzen anderer Kaiser im CN Datensatz sehr viele Fehlklassifizierungen durchführt.

Faustina II. Um die bei der Klasse Augustus gemachten Beobachtungen noch einmal zu überprüfen, soll hier noch einmal in kürzerer Form auf die Klasse Faustina II. eingegangen werden. Deren auf den OCRE Daten hervorragenden Accuracy Werte (Top-1: 97% und Top-5: annähernd 100%) sind auf den CN Daten ebenfalls noch gut (Top-1: 63% und Top-5: 87%). Die OCRE Trainingsdaten haben wiederrum einen beachtlichen Anteil von Porträts inne, die durch Korrosion oder Abnutzung verschiedene Grade der Beschädigung aufweisen (besonders bei Bronzemünzen). Genauso wichtig ist die Tatsache, dass die Porträts in unterschiedlichen Weisen dargestellt sind. Dies fällt besonders bei den unterschiedlichen Frisuren der Kaiserin auf (vergl. Abb. 44 Links und Abb. 44 Mitte links)[161]. Die OCRE Heatmaps der korrekt erkannten Porträts zeigen, dass hier wie bei der Klasse Augustus vorwiegend bei schlechter erhaltenen Porträts großflächige Wärmebereiche zu finden sind (siehe Abb. 44 Links). Der Fokus liegt allgemein besonders auf der Haarpartie und wird durch Zonen wie den Hals- und Gesichtsbereich in einigen Fällen ergänzt (siehe Abb. 44 Mitte links). Dieses Bild findet sich auch bei den korrekt erkannten CN Porträts. Es gibt viele großflächige Wärmebereiche bei den Porträts mit Beschädigungen und die Haarpartie ist ansonsten wichtigstes Identifizierungsmerkmal (siehe Abb. 44 Mitte rechts). Die falsch erkannten CN Bilder wurden häufig anhand des Gesichtes, ergänzt durch die Haare, klassifiziert (siehe Abb. 44 Rechts). Wenn großflächige Wärmezonen auf den Falschklassifizierungen vorkommen, dann sind es meist stärker beschädigte Porträts und diese werden auch bei dieser Klasse öfter als Augustus angesprochen (15 von 115 Münzen). Somit wirkt der bei Augustus festgestellte „Bias“ auch bei einer weiblichen Klasse in geringerem Umfang nach. Letztendlich können die bereits oben bei Augustus festgestellten Beobachtungen zum besseren Abschneiden der Klasse auf CN Daten auch bei Faustina II. gemacht werden:

Abb. 44: Links: Heatmap des Typus RIC III Marcus Aurelius 1688; Mitte links: Heatmap des Typus RIC III Antoninus Pius 517C (denarius); Mitte rechts: Heatmap einer unveröffentlichten CN Münze; Rechts: Heatmap des Typus CN Type 2448.

[161] Übersicht zu den verschiedenen Münzporträts der Faustina II. in der OCRE Datenbank: Online Coins of the Roman Empire. Faustina II., <http://numismatics.org/ocre/results?q=%28portrait_facet%3A%22Faustina+II%22+OR+portrait_facet%3A%22Faustina+II+Diva%22+OR+portrait_facet%3A%22Faustina+die+J%C3%BCngere%22%29> (16.10.2021).

Eine Trainingsmenge mit einer größeren Bandbreite an verschiedenen Darstellungsstilen und mit schlechter erhaltenen Porträts kann ein Modell auf einen Bilddatensatz mit einer größeren Anzahl Münzen in verschiedenen Erhaltungszuständen und lokaleren Darstellungsarten besser vorbereiten, da das Modell weniger detailfokussiert ist und somit bessere Ergebnisse erreicht werden können.

Fazit Die Auswertung des Einsatzes des Modells 13 auf den CN Daten hat folgende Gründe für die schlechtere Leistung erbracht:

- Der vorwiegend schlechtere Erhaltungszustand und die abweichende Darstellungsweise, die durch verschiedene Stempelschneider der einzelnen Städte aus dem CN Gebiet bedingt ist, ist hauptsächlich für diese Ergebnisse verantwortlich.
- Klassen, deren OCRE Trainingsdaten relativ unausgewogen sind (vorwiegend gut erhaltene Porträts und uniformer Darstellungsstil) schneiden weitaus schlechter ab, als die wenigen Klassen, die über verschiedene Darstellungsweisen der Porträts in unterschiedlichem Erhaltungszustand verfügen.
- Es konnte sogar ein gewisser „Bias“ bezogen auf die Klasse Augustus, die über eine große Trainingsbildmenge mit beschädigten und unterschiedlich dargestellten Porträts verfügt, festgestellt werden.
- Eine mögliche Lösung dieses Problem wäre es, die eher uniformen Trainingsmengen der anderen Klassen mit heterogenen Bildern aus dem CN Datensatz anzureichern, um den Fokus des Modells beim Lernen von den Details stärker auf das ganze Porträt zu lenken, damit auch schlechter erhaltene Porträts besser erkannt werden können.
- Im derzeitigen Zustand kann das Modell für Numismatiker, die mit regionalen Datensätze wie den von CN arbeiten, nur eine sehr eingeschränkte Hilfe sein, da nur wenige Klassen zuverlässig erkannt werden. Jedoch weisen auch die falschen Ergebnisse recht oft in die richtige Zeitstellung der korrekten Klasse.

c) Einsatz auf vollständigen Münzen

Mit dem Einsatz des Modells 13 auf den vollständigen Münzen der Testmenge in der Masterarbeit von A. Loyal wurden ebenfalls deutlich schlechtere Accuracy Werte (Top-1: 34% und Top-5: 58%) erreicht als bei den Testbildern von Bildmenge 2.3 (siehe Abschn. 9.3 Auflistung 5)[162]. Worin dieser starke Unterschied begründet liegt, soll hier untersucht werden.

[162] Die Testmenge von A. Loyal wurde für diesen Test auf die in dieser Arbeit trainierten 66 Klassen beschränkt (siehe Abschn. 4.2). Es kann hier nicht ausgeschlossen werden, dass Porträts der Testmenge von A. Loyal als Trainingsbilder in der neu erstellten Bildmenge 3.2 dienten.

Eine naheliegende Antwort wäre wohl, dass das Modell durch die neu hinzukommenden Bestandteile der Münze, die nicht Teil der Trainingsbilder waren (Legende, Beizeichen usw.). von den eigentlichen Porträts abgelenkt wird. Zum Testen dieser Theorie sollen wieder mit Grad-CAM erstellte Heatmaps von Klassen mit einer aussagekräftigen Menge an Bildern, die durch ihre Accuracy Werte auffallen, herangezogen werden.

Diokletian Die Klasse *Diokletian* hat mit einer Top-1 Accuracy von 0% und einer Top-5 Accuracy von ebenfalls annähernd 0% mit Abstand die schlechtesten Werte aller Klassen in diesem Test. Die Daten der OCRE Testbilder waren dagegen recht gut (Top-1: 77% und Top-5: 99%). Bei diesen Bildern kommt auch das bereits oben beobachtete Problem mit den Kaisern der Tetrarchie, die von Diokletian begründet wurde, zum Tragen, da das Modell diese oft untereinander verwechselt. So bestehen die falschen Klassifizierungen auf den ausgeschnittenen OCRE Testbildern zu einem sehr großen Teil aus Mitgliedern der Tetrarchie, wobei die Klasse Maximian mit einem Anteil von 30% an den Gesamtdaten eindeutig dominiert. Bei den richtigen Klassifizierungen zeigen die Heatmaps als primäre Zonen der Identifizierungen vor allem Haar- und Bartpartie, die mit dem Augen- und Nasenbereich ergänzt werden können (siehe Abb. 45 Links). Auf den vollständigen Münzen nimmt die Verwechslung der Tetrarchen untereinander nun extreme Formen an. Von den 225 falsch klassifizierten, vollständigen Münzen werden von dem Modell alleine 81 Münzen der Klasse Konstantin I., 48 der Klasse Licinius und 16 der Klasse Constantinus Chlorus zugeschrieben. Alle drei Kaiser waren Mitglieder der besagten Tetrarchie. Bei einem Blick auf die Heatmaps der vollständigen Münzen, wurde festgestellt, dass bei 75% der Münzen die Legende vollständig oder teilweise zur Identifizierung der Klasse herangezogen wurde (siehe Abb. 45 Mitte). Bei 15% diente die Legende sogar als alleiniges Merkmal (siehe Abb. 45 Rechts). Hier kann jedoch ein Erkennen des Kaisernamens wie bei dem von A. Loyal trainierten Modell eher ausgeschlossen werden. Bei falsch klassifizierten Bilder mit einem Fokus auf den Legenden wird meistens der Name „Diokletianius“ mit einer Wärmezone markiert (siehe Abb. 45 Mitte und Rechts). Jedoch kommt dieser bei den Münzen mit Porträts von Konstantin I. und Licinius, die am zahlreichsten anstelle von Diokletian hier identifiziert wurden, nicht in der Legende vor. Es wurde bei diesen drei Klassen außerdem überprüft, ob die Porträtbereichserkennung in den Trainingsdaten vielleicht größere Bereiche der Legende im ausgeschnittenen Bild belassen hat. Dies kann bei Konstantin I., Licinius und Diokletian selbst aber ausgeschlossen werden. Vielmehr ist es sogar so, dass die Porträtbereichserkennung mit diesen Porträts sehr gut zurechtkommt und sie mit lediglich minimalen Legendenresten ausschneidet. Daher kann an dieser Stelle als Erklärung für das Verhalten des Modells lediglich die zwar im Training nur in

sehr kleinen Ausschnitten vorkommende Bestandteile wie die Legende, die das Modell von dem eigentlichen Porträt ablenkt, herangezogen werden.

Abb. 45: Links: Heatmap des Typus RIC VI Cyzicus 12a; Mitte: Heatmap des Typus RIC VI Aquileia 7a; Rechts: Heatmap des Typus RIC VI Cyzicus 10a.

Augustus Die vollständigen Münzen mit Porträts des Kaisers *Augustus* sind mit Accuracy Werten von 33% (Top-1) und 68% (Top-5) deutlich korrekter identifiziert worden als die der Klasse Diokletian. Auch liegen sie leicht über dem Durschnitt des Modells für alle vollständigen Münzen. Die Werte der Klasse auf den OCRE Testbildern der Bildmenge 2.3 sind sehr gut und wie bereits oben festgestellt werden als falsche Klassen hauptsächlich Kaiser des 1 Jhs. n. Chr. erkannt (siehe Abschn. 5.3.b). Der Kaiser wird bei eher beschädigten Porträts mit großflächigen Wärmezonen, die sich über das Porträt erstrecken, und ansonsten an Gesichts und Haarpartie erkannt (siehe Abb. 43 Links und Mitte). Die falsch erkannten Klassen auf den vollständigen Münzbildern verteilen sich dagegen auf die Kaiser des 1. – 4 Jhs. n. Chr., wobei der Anteil der Kaiserinnen mit ca. 29% recht hoch ist. Die meisten fälschlich genannten Klassen stehen zeitlich in keiner Beziehung zu Augustus. Bei den Heatmaps von richtig und falsch identifizierten Bildern fällt wiederrum die starke Einbeziehung der Inschrift in die Entscheidungsfindung des Modells auf. Bei knapp 50% der korrekt erkannten Bilder wird die Klassifizierung teilweise anhand der Legende durchgeführt. Eine vollständige Erkennung anhand der Legende kommt hier nicht vor. Der Anteil der falsch klassifizierten Münzen mit einem zumindest teilweisen Legendenfokus (siehe Abb. 46 Links) liegt bei ca. 63%. Vollständig anhand der Inschrift oder anderen Partien der Münze (siehe Abb. 46 Mitte) erkannte Bilder machen einen Anteil von ca. 7% aus. Dieser ist damit deutlich kleiner als bei der Klasse Diokletian. Ebenso werden andere Bestandteile wie der Perlkreis hier miteinbezogen (siehe Abb. 46 Rechts). Somit wird die Aufmerksamkeit des Modells 13 auch bei dieser Klasse wieder sehr stark von der Legende und anderen Münzbestandteilen von dem eigentlichen Porträt abgelenkt, was sich somit wohl auch bei den schlechteren Ergebnissen im Vergleich zu den Testdaten von Bildmenge 2.3 niederschlägt.

Abb. 46: Links: Heatmap des Typus RIC I (second edition) Augustus 174; Mitte: Heatmap des Typus RIC I (second edition) Gaius/Caligula 56; Rechts: Heatmap des Typus RIC I (second edition) Augustus 173B.

Konstantin I. Die vollständigen Münzen mit einem Porträt des Konstantin I. (der Große) wurden wesentlich besser von dem Modell 13 erkannt als alle anderen Klassen. Die Top-1 Accuracy liegt bei 86% und die Top-5 Accuracy bei 98%. Diese Werte sind nur um weniges schlechter als die Werte der Klasse auf den Testbildern von Bildmenge 2.3 (Top-1: 90% und Top-5: 99%). Somit stellt sich hier die Frage, wie man dieses deutlich bessere Abschneiden der Klasse begründen kann. Bei den falsch klassifizierten OCRE Porträts werden fast ausschließlich andere Mitglieder der Tetrarchie und die Söhne Konstantins als Top-1 Ergebnis vorhergesagt. Somit ist auch hier das bereits festgestellte Problem der Verwechslung der Kaiser dieser Dynastie vorhanden (s.o.). Die korrekt erkannten Porträts werden vorwiegend an dem Gesichtsbereich und der Stirn mit dem Haaransatz erkannt. In sehr seltenen Fällen werden minimale Bestandteile der Legende miteinbezogen (siehe Abb. 47 Links). Wenn der Kaiser im Priester Habitus mit über den Kopf gezogener Toga („*capite velato*“) dargestellt ist, wird er auch vorwiegend an diesem Teil der Toga erkannt (siehe Abb. 48 Mitte rechts). Die falschen Klassifizierungen auf den vollständigen Münzen verteilen sich ebenfalls relativ ausgeglichen auf Tetrachen, Mitglieder der konstantinischen Dynastie und die spätere valentianische Dynastie. Trotz der guten Werte hier bieten die Heatmaps der Klasse ein ähnliches Bild wie die beiden oben untersuchten Klassen. Mindestens 84% der Heatmaps der vollständigen Münzen zeigen eine teilweise Konzentration des Modells auf die Legende oder andere Münzbestandteile, die nicht zum Porträt gehören. Die Heatmaps, die einen ausschließlichen Fokus auf die Inschrift oder andere Münzbestandteile legen, liegen jedoch deutlich unter einem Prozent. Selbst bei ausgeschnittenen Porträts, die in der Trainingsmenge des Modells 13 vorhanden sind und somit dem Modell bestens bekannt sein sollten, wird bei der vollständigen Münze mit diesem Porträt ein größerer Fokus auf die Legende gelegt (siehe Abb. 48 Links und Mitte links). Ansonsten verteilen sich die Wärmezonen auch hier vor allem auf das Gesicht und die Haarpartie (siehe Abb. 47 Mitte und Rechts).

Abb. 47: Links: Heatmap des Typus RIC VII Arelate 80; Mitte: Heatmap des Typus RIC VII Rome 97; Rechts Heatmap des Typus RIC VII Arelate 110.

Abb. 48: Links Heatmaps des Typus RIC VI Thessalonica 61b aus der Trainingsmenge des Modells 13; Mitte links: Heatmap des Typus RIC VI Thessalonica 61b; Mitte Rechts: Heatmap des Typus RIC VIII Constantinople 68 (gespiegelt), Rechts: Heatmap des Typus RIC VIII Antioch 112.

Erstaunlicherweise kommen die Porträts der vollständigen Münzen mit einer Darstellung mit über den Kopf gezogener Toga weitestgehend ohne einen Fokus auf die Inschrift oder andere Bestandteile aus (22 von 40 Münzen, siehe Abb. 48 Rechts). Dieses bei der Kaiserdarstellung allgemein eher selten vorkommende Merkmal des Porträts scheint auch auf vollständigen Münzen für das Modell gut erkennbar und somit ausreichend für eine richtige Klassifikation zu sein. Solche Porträts wurden bei diesem Test lediglich dreimal falsch klassifiziert. Die falsch identifizierten Bilder vollständiger Münzen zeigen auch zu ca. 70% Wärmezonen, die auf der Legende oder anderen Münzbestandteilen liegen. Ansonsten sind Haare und Gesicht hier Grundlagen der Erkennung und damit unterscheiden sich die falsch klassifizierten Münzen kaum von denen, die richtig zugeordnet wurden. Eine Durchsicht der Trainingsbilder des Konstantin I. aus Bildmenge 2.3 erbrachte auch dieses Mal keine erwähnenswerte Menge an ausgeschnittenen Porträts mit größeren Resten der Legende. Da diese Klasse die meisten Trainingsbilder (6871) in Bildmenge 2.3 besitzt, wurde ein möglicher „Bias“ aufgrund dieser vielen Bilder in Erwägung gezogen. Auch die Klasse Hadrian (6673 Trainingsbilder, Top-1: 75%), Constantius II. (4798 Trainingsbilder, Top-1: 61%), Commodus (4109 Trainingsbilder, Top-1: 86%) schneiden mit ihren Werten jeweils überdurchschnittlich auf vollständigen Münzbildern ab (siehe Abschn. 9.3 Auflistung 5). Jedoch kann die Klasse mit den fünftmeisten Trainingsbildern, Antoninus Pius (3392 Trainingsbilder), diese These nicht weiter stützen, da sie mit einer Top-1 Accuracy von unter zwei Prozent völlig aus dem Rahmen fällt. Die Klasse

Tetricus II. erreicht dagegen mit 1413 Trainingsbildern eine Top-1 Accuracy von 79%. Somit kann eine hohe Anzahl der Trainingsbilder lediglich eine größere Wahrscheinlichkeit für gute Ergebnisse bieten, aber eine definitive Erklärung für das bessere Abschneiden bestimmter Klassen kann sie nicht liefern. Als eine mögliche andere Erklärung bleibt die Tatsache, dass trotz der Einbeziehung der Münzlegende bei den allermeisten Münzen immer noch Teile des Porträts des Konstantin I. hier in der Aufmerksamkeit des Modells stehen. Somit reichen diese Zonen möglicherweise aus um die Klasse weitestgehend richtig vorhersagen zu können. Für weitere mögliche Erklärungen ist die Aussagekraft der Heatmaps jedoch nicht mehr ausreichend.

Fazit Die Überprüfung eines Einsatzes des Modells 13 auf den vollständigen Münzbildern hat folgende Gründe für die schlechtere Erkennung erbracht:

- Es wurde ein starker Fokus des Netzes auf die Legenden der Münzen anhand der Heatmaps festgestellt, obwohl die Legenden nur in sehr geringem Umfang auf den Trainingsdaten des Modells 13 vorhanden waren.
- Dabei kann der Fokus in einer kleineren Zahl von Bildern vollständig oder bei dem Großteil der Bilder nur partiell auf der Legende liegen. Eine Begründung für dieses Verhalten kann jedoch mit den Heatmaps nicht gegeben werden.
- Dieses Verhalten kann außerdem zu völlig unterschiedlichem Accuracy Werten bei den verschiedenen Klassen führen. Unterschiedliche Klassen, die darstellungstechnisch relativ ähnlich sind, erreichen sehr gute wie auch schlechte Werte.
- Der Einsatz von mehr Trainingsdaten pro Klasse erhöht lediglich die die Wahrscheinlichkeit für bessere Ergebnissen.
- Eine Lösung könnte ein Training mit einer Standardmünze ohne Porträt aber mit einer Legende sein. Dies könnte dem Modell helfen sich im Falle einer Anwendung auf vollständige Münzen auf die Features des Porträts zu konzentrieren.
- Solche eine Weiterentwicklung wäre vor allem für das Modell 19, das in diesem Bereich bessere Werte gezeigt hat, denkbar (siehe Tab. 2).
- Insgesamt gesehen ist die Anwendung der Portbereichserkennung auf Münzfotos und der anschließenden Klassifizierung durch ein Modell aufgrund seiner sehr guten Werte bei den Metriken ein besseres Hilfsmittel für Numismatiker.

6. Resümee und Ausblick

In dieser Arbeit wurden die Möglichkeiten und Probleme eines auf ausgeschnittenen Münzporträts römischer Kaiserinnen und Kaiser trainierten *VGG16* Netzes als Hilfsmittel für Numismatiker untersucht. Diese Porträts wurden mithilfe der *Online Coins of the Roman Empire* Datenbank gefunden. Weiterhin wurde auch eine Verwendung bei regionalen Beständen von Münzen mit Kaiserporträts, die im *Corpus Nummorum* Projekt gesammelt und typologisiert werden, und auf vollständigen Münzen aus der OCRE Datenbank untersucht.

Vor der eigentlichen Implementation wurde die sogenannte Porträtbereichserkennung, die ein Ausschneiden des Porträtbereiches einer Münze durch ein *Regional Convolutional Neural Network* ermöglicht, im Vorfeld dieser Arbeit in Zusammenarbeit mit P. Bonack umgesetzt. Sie stellt den ersten Schritt zur Lösung des in der Bachelor Arbeit des Autors festgestellten Problems der Miteinbeziehung der Münzlegende in die Identifizierung des Kaisers durch das VGG16 Netz aus der Masterarbeit von A. Loyal dar.

Im Rahmen dieser Arbeit wurden als erstes zwei verschieden große Trainingsdatensätze (DS 1 und 2) mit via OCRE gefunden, mithilfe der Porträtbereichserkennung ausgeschnittenen Porträts erstellt. Ein weiterer Datensatz (DS 3) enthält die mit der CN Datenbank gefundenen Porträts. Er wurde zum Testen der Übertragbarkeit der trainierten Modelle erstellt. Das hier verwendete, auf einem *Convolutional Neural Network* basierende VGG16 Netz wurde mittels *Tensorflow Keras* umgesetzt. Dabei wurden mehrere Endsegmente des Netzes, die Möglichkeit zum Testen der verschiedenen Hyperparameter und Auswertungsfunktionen zur Erfassung der verwendeten Machine Learning Metriken implementiert.

Der Hauptteil der vorliegenden Arbeit widmet sich dann drei verschiedenen Experimenten, die mit den trainierten Modellen auf Basis des VGG16 Netzes durchgeführt wurden.

Experiment 1 In Experiment 1 wurde die Möglichkeit untersucht mithilfe von Keras das originale VGG16 Netz mit seiner 1000 Klassen Ausgabe für die hier trainierten 66 Klassen zu verwenden. Dabei konnte festgestellt werden, dass es eine Abweichung der von Keras am Ende des Trainingsvorgangs ausgegeben Top-1 Accuracy von dem selbst berechneten Wert gibt und sich dieses Problem bei allen in dieser Arbeit trainierten Modellen finden lässt. Außerdem muss auf die korrekte Umwandlung der Eingabebilder in ein Array mit Float32 Werten geachtet werden. Die 1000 Klassen Ausgabe dieses Netzes wurde für den hier verfolgten Zweck der Kaisererkennung als nicht hilfreich eingestuft, da es nicht ausgeschlossen werden kann, dass es bei der aktuellen Implementation zur Schlüsselfehlern und der Ausgabe von unbekannten Klassen als Top-1 oder Top-5 Ergebnis kommen könnte.

Experiment 2 Das zweite Experiment beschäftigte sich mit der Suche nach der besten Kombination aus Endsegment des VGG16 Netzes, der Trainingsmenge und der Hyperparameter Lernrate und Epochenanzahl. Zuerst wurden Modelle mit dem Endsegment 1, welches über drei vollständig verbundene Schichten verfügt und deshalb mehr Speicherplatz benötigt, getestet. Hierbei stellte sich heraus, dass dieses Endsegment mit einer kleineren Anzahl von sechs Epochen, einer daran angepassten Lernrate (Lernrate 3) und der größten Bildmenge 2.3 die besten Ergebnisse auf den OCRE Testbildern und den CN Porträts lieferte (Modell 13). Die verwendete Bildmenge besteht aus Porträts, deren ursprünglich hochrechteckige Form mit Hilfe von schwarzen Seitenbalken zu einem quadratischen Format, dass auch dem Eingabeformat des VGG16 Netzes entspricht, modifiziert wurde. Diese Bearbeitung der Porträts verursachte nicht nur keine Einbußen bei den Accuracy Werten, sondern konnte auch die Leistung auf Bildern von vollständigen Münzen steigern. Gleichzeitig wurde die Größe der Trainings- und Testbilder durch eine Spiegelung sämtlicher Porträts verdoppelt. Dies ermöglicht es dem Modell die selteneren nach links gerichteten Porträts besser zu erkennen. Bei der Auswertung der Modelle mit dem schlankeren Endsegment 2 stellte sich heraus, dass hier eine Epochenanzahl von zehn mit der Lernrate 1 und der Bildmenge 2.3 zu den Modellen mit Endsegment 1 vergleichbare, sehr gute Ergebnisse liefert. Jedoch ist das beste Modell 19 auf den CN Daten etwas schwächer als Modell 13, kann dieses jedoch bei den vollständigen Münzen übertreffen. Das Modell 19 lässt sich durch seinen geringeren Speicherplatzverbrauch gut für Webapplikationen einsetzen. Da die Übertragbarkeit auf andere Datensätze als wichtiger eingeordnet wurde und auch die besseren Ergebnisse von Modell 19 auf vollständigen Münzbildern noch nicht ausreichend gut waren, wurde Modell 13 als bevorzugtes Modell für das anschließende Experiment ausgewählt.

Experiment 3 Im letzten Experiment wurde das Modell 13 und sein Vorhersageverhalten einer genaueren Untersuchung unterzogen. Zuerst wurde das Modell mit dem finalen Modell von A. Loyal verglichen. Die Top-1 Accuracy (86%) von Modell lag leicht unter der von A. Loyal (91%), dafür war die Top-5 Accuracy des Modells jedoch noch einmal besser (99% zu 92%). Für ein Abfallen bei der Top-1 Accuracy konnten vor allem die Klassen der konstantinischen, valentianischen und theodosianischen Kaiserhäuser ausgemacht werden, da diese aus eher abstraktem, einander sehr ähnlichen Porträts zusammensetzt sind. A. Loyals Modell konnte hier noch die Münzlegende als Feature zur Identifizierung der Kaiser einsetzen und somit bei diesen Klassen bessere Werte erreichen.

Im zweiten Schritt wurde das Verhalten auf den CN Bildern mit Hilfe von Grad-CAM Heatmaps ausgewertet. Bei mehreren Beispielklassen konnte festgestellt werden, dass der

schlechtere Erhaltungszustand der Münzen aus diesem Gebiet, die hauptsächlich aus Bronzestücken und Abgüssen davon bestehen, und der abweichende regionale Darstellungsstil der lokalen Stempelschneider für das schlechtere Abschneiden des Modells 13 auf diesen Daten verantwortlich ist. Dabei spielt vor allem die Uniformität der Trainingsdaten der einzelnen Klassen eine große Rolle. Klassen wie *Augustus*, die über deutlich mehr Münzen mit schlechterem Erhaltungszustand und verschiedenen Arten der Darstellung verfügen, werden deutlich besser erkannt als Kaiser mit eher gleichaussehenden Porträts und vielen gut erhaltenen Münzen. Zur Verbesserung der Übertragbarkeit des Modells sollten die OCRE Trainingsdaten der Kaiser mit Münzen aus den wichtigsten Münzstätten des römischen Reiches mit weniger gut erhaltenen lokalen Städteprägungen angereichert werden.

Der letzte Test in Experiment 3 beschäftigte sich mit den eher schlechten Ergebnissen des Modells auf Bildern vollständiger Münzen. Im Zuge dessen wurden wiederrum Heatmaps der Grad-CAM Methode für drei Beispielklassen ausgewertet. Dabei konnte ein für viele Bilder individuell mehr oder weniger starker Fokus des Modells auf die Münzlegende festgestellt werden. Das Modell von A. Loyal konnte durch das Training auf vollständigen Münzen die Legende noch zur Identifizierung nutzen. Das Modell 13 kann dies jedoch nicht leisten, da die Trainingsdaten nur minimale Ausschnitte der Legende beinhalten. Somit ist wohl der Grund für die deutlich schlechteren Ergebnisse auf dieses Abweichen der Aufmerksamkeit von dem eigentlichen Porträt zurückzuführen. Hierbei kommt der Aussagewert der Heatmaps an seine Grenzen, da es nicht ersichtlich wird, warum beispielsweise auch bei Münzen, deren Porträt schon in den Trainingsdaten des Modells 13 vorhanden ist, die Legende miteinbezogen wird. Eine mögliche Lösung für dieses Verhalten könnte ein Einsatz von einer standardisierten, mit einer Legende versehenen Münze ohne Porträt, in welche das individuelle Porträt eingefügt wird. So könnte das Modell lernen im Falle einer Anwendung auf vollständige Münzbilder hauptsächlich das Porträt zur Identifizierung zu nutzen. Die Erhöhung der Anzahl der Trainingsbilder mit ausgeschnittenen Porträts erhöht zwar die Wahrscheinlichkeit für bessere Ergebnisse, kann diese aber nicht garantieren.

Ergebnisse und Ausblick Folgende Ergebnisse konnten in dieser Arbeit gewonnen werden:

- Ein Modell, welches auf den Porträts römischer Kaiser trainiert wurde, kann im Gegensatz zu einem auf vollständigen Münzen trainierten Modell diese Porträts auch mit sehr hoher Wahrscheinlichkeit richtig identifizieren.
- Die Pipeline aus Porträtbereichserkennung und dem Modell 13 hat bei den Experimenten die besten Top-1 und Top-5 Ergebnisse geliefert. Von der Performance bezogen auf die OCRE Datensätze kann sie bereits als Hilfsmittel für Numismatiker dienen.
- Die Übertragung auf andere Datensätze mit Münzen aus lokaler Produktion ist dagegen noch eingeschränkt und noch nicht als zuverlässige Unterstützung von Numismatikern tauglich. Durch weitere Porträts aus den Städteprägungen der einzelnen Regionen des römischen Reiches im Training könnte hier Abhilfe geschaffen werden.
- Ebenso wenig sind die hier trainierten Modelle auf Bildern von vollständigen Münzen eine zuverlässige Hilfe, da hier die anderen Münzbestandteile das Modell zu falschen Ergebnissen führen. Hier könnten noch mit einer Augmentation der Porträts mit Standardmünzen bessere Ergebnisse erreicht werden.

Folgende Erkenntnisse wurden außerdem bei der Bearbeitung des Themas erlangt:

- Die internen, beim Trainingsvorgang angezeigten Metriken von Keras liefern meist bessere Ergebnisse als ein erneuter, manuell gestarteter Durchlauf der jeweiligen Testmengen der Porträts. Daher muss darauf geachtet werden diese *zu guten* Ergebnisse mit dem Nachtest notfalls zu korrigieren, damit keine falschen Zahlen präsentiert werden.
- Die Problematik des Integer / Float32 Eingabearrays und der damit verbundenen unterschiedlichen Ergebnisse eines Modells zeigt, dass auch eher kleinteilige Arbeitsschritte wie das Umwandeln eines Bildes in ein Array größere Probleme verursachen können. Dieses Problem musste mit einer zeitaufwändigen Suche erst einmal händisch identifiziert werden.
- Die Erstellung von verschiedenen Datensets aus den OCRE und CN Datenbanken war sehr arbeitsaufwändig, da neben Problemen beim Download der Bilder auch das händische Aussortieren von einzelnen Münzen aufgrund von Fehlern in den Datenbanken nötig geworden war. Viele Fehler davon konnten an die Betreiber rückgemeldet werden.

- Das Training und das Testen von 19 verschiedenen Modellen ist ebenfalls mit einem großen Zeitaufwand verbunden, da einerseits ein Trainingsvorgang immer mehrere Stunden in Anspruch nimmt und andererseits jedes dieser Modelle noch auf unterschiedliche Bildermengen getestet werden muss. Die Erstellung und die händische Auswertung von Tausenden von Heatmaps zu den Ergebnissen von Modell 13 trug ebenfalls enorm zu diesem Aufwand bei.

Ausblick Als Ausblick für den Einsatz von CNNs auf Bildern von antiken Münzen ist zu sagen, dass diese Arbeit das Potential dieser Netze für die Numismatik deutlich zeigen konnte. Vor allem für lokal beschränkte Münzdaten ist aber durch Hinzunahme von Bildern aus diesen Datensätzen jedoch noch Raum für Verbesserungen vorhanden. Jedoch ist die Anwendung nicht nur auf antike Porträts beschränkt. Die Typerkennung von Münzen, sowie der Einsatz auf anderen Feldern der antiken Flächenkunst wie beispielsweise den griechischen Bildvasen sind weitere Möglichkeiten des Einsatzes, der auch auf andere Epochen der Geschichte ausgeweitet werden kann. Letztendlich können solche CNNs, wie das hier verwendete VGG16, in Zusammenarbeit mit Domainexperten zu sehr hilfreichen Werkzeugen im jeweiligen Arbeitsgebiet entwickelt werden.

7. Literaturverzeichnis

Bishop 2006

C. M. Bishop, Pattern Recognition and Machine Learning (New York 2006).

Chollet 2018

F. Chollet, Deep Learning mit Python und Keras. Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek (Frechen 2018).

Davidsson 1998

P. Davidsson, Coin Classification Using A Novel Technique For Learning Characteristic Decision Trees By Controlling The Degree Of Generalization, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.899.5401&rep=rep1&type=pdf> (15.09.2021).

Fukumi u. a. 1992

M. Fukumi – S. Omatu – F. Takeda – T. Kosaka, Rotation-Invariant Neural Pattern Recognition Systems with Application to Coin Recognition, <https://www.sice.jp/e-trans/papers/E1-23.pdf> (15.09.2021).

Gampe 2019

S. Gampe, Kombination maschineller Lernmethoden der Bild- und Texterkennung auf antiken Münzdaten, Bachelorarbeit (Frankfurt a. M. 2019).

Géron 2019

A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Concepts, Tools and Techniques to Build Intelligent Systems (Sebastopol 2019).

Girshick u. a.

Rich feature hierarchies for accurate object detection and semantic segmentation. Tech Report (v5), <https://arxiv.org/pdf/1311.2524v5.pdf> (08.10.2021).

Goodfellow u. a. 2016

I. Goodfellow – Y. Bengio – A. Courville, Deep Learning (Cambridge 2016).

Gruber 2018

E. Gruber, Linked Open Data and Hellenistic numismatics, in: S. Glenn – F. Duyrat – A. Meadows (Hrsg.), Alexander the Great. A Linked Open World (Pessac 2018) 17–34.

Hölscher 2016

T. Höscher, Consensus universorum. Die Akzeptanz der Herrschaft des Augustus in Bau- und Bildwerken, öffentlich und privat, in: E. Baltrusch – C. Wendt (Hrsg.), Der Erste. Augustus und der Beginn einer neuen Epoche (Darmstadt 2016) 43–65.

Howgego 1995

C. Howgego, Ancient History from Coins (London 1995).

Jia u. a. 2014

Y. Jia – E. Shelhamer – J. Donahue – S. Karayev – J. Long – R. Girshick – S. Guadarrama – T. Darrell, Caffe. Convolutional Architecture for Fast Feature Embedding, <https://arxiv.org/pdf/1408.5093.pdf> (15.09.2021).

Jostock 2016

M. Jostock. Automatische Münzerkennung mit OpenCV, Bachelorarbeit (Frankfurt 2016).

Kim –Pavlovic 2015

J. Kim – V. Pavlovic, Discovering Characteristic Landmarks on Ancient Coins using Convolutional Networks, <https://arxiv.org/pdf/1506.09174.pdf> (15.09.2021).

Kiourt – Evangelidis 2021

C. Kiourt – V. Evangelidis, AnCoins: Image-Based Automated Identification of Ancient Coins Through Transfer Learning Approaches, in: A. Del Bimbo – R. Cucchiara – S. Sclaroff – G.M. Farinella – T. Mei – M. Bertini – H.J. Escalante – R. Vezzani (Hrsg.), Pattern Recognition. ICPR International Workshops and Challenges 2021, Lecture Notes in Computer Science 12667, 54 – 67.

Klinger 2018

P. Klinger, Natural Language Processing to enable semantic search on numismatic descriptions, Bachelorarbeit (Frankfurt a. M. 2018).

Klinger u. a. 2018

P. Klinger – S. Gampe – K. Tolle – U. Peter, Semantic search based on Natural Language Processing - A numismatic example, Journal of Ancient History and Archaeology 5.3, 2018, 68–79.

Krizhevsky u. a. 2012

A. Krizhevsky – I. Sutskever – G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, <https://arxiv.org/pdf/1506.09174.pdf> (15.09.2021).

Liu u. a. 2018

H. Liu – Q. Yin – W. Y. Wang, Towards Explainable NLP. A Generative Explanation Framework for Text Classification, <https://arxiv.org/pdf/1811.00196.pdf> (15.09.2021).

Loyal 2018

A. Loyal, Maschinelles Lernen angewendet auf Bilder antiker Münzen. Masterarbeit (Frankfurt a. M. 2018).

Peschlow 1983

U. Peschlow, Zum Kaiserporträt des 4. bis 6. Jh. n. Chr., in: H. Beck – Peter.C. Bol (Hrsg.), Spätantike und frühes Christentum. Ausstellungskatalog Frankfurt am Main (Frankfurt a. M. 1983) 61–69.

Petkovic u. a. 2018

D. Petkovic – R. Altman – M. Wong – A.Vigil, Improving the explainability of Random Forest classifier – user centered approach, in: R. Altman – A. Dunker – L. Hunter (Hrsg), Biocomputing 2018. Proceedings of the Pacific Symposium. Internationales Symposion Biocomputing Kohala Coast, Hawaii 3–7.1.2018, 204–215.

Saleh 2018

H. Saleh, Machine Learning Fundamentals, <https://www-oreilly-com.proxy.ub.uni-frankfurt.de/library/view/machine-learning-fundamentals/9781789803556/?ar> (20.09.2021).

Selvaraju u. a. 2017

R. R. Selvaraju – M. Cogswell – A. Das – R. Vedantam – D. Parikh – D. Batra, Grad-CAM. Visual Explanations from Deep Networks via Gradient-based Localization, <https://arxiv.org/pdf/1610.02391.pdf> (31.08.2021).

Shukla - Fricklas 2018

N. Shukla – K. Fricklas, Machine Learning with Tensorflow (Shelter Island 2018).

Simonyan – Zisserman 2014

K. Simonyan – A. Zisserman, Very Deep Convolutional Networks for Largescale Image Recognition, <https://arxiv.org/pdf/1409.1556.pdf> (31.08.2021).

8. Abbildungsnachweis

Abb. 1: American Numismatic Society. Silver Denarius of Antoninus Pius, Rome, AD 139 1956.127.352, <http://numismatics.org/collection/1956.127.352> (24.07.2021).

Abb. 2: American Numismatic Society, Bronze As of Augustus, Nemausus, 9 BC - 3 BC. 1944.100.68873, <http://numismatics.org/collection/1944.100.68873> (24.07.2021).

Abb. 5: Diagram nach: Loyal 2018, 14 Abb. 2.7.

Abb. 8: Heatmap nach: American Numismatic Society, 1944.100.39040 <http://numismatics.org/collection/1944.100.39040> (27.07.2021).

Abb. 9: Münzbilder im Diagramm: RIC I (second edition) Augustus 412: American Numismatic Society. Silver Denarius of Augustus, Rome, 12 BC 1937.158.395, <http://numismatics.org/collection/1937.158.395> (14.09.2021);

RIC I (second edition) Augustus 43B: American Numismatic Society, 1944.100.39040 <http://numismatics.org/collection/1944.100.39040> (27.07.2021).

Abb. 10: Münzbilder im Diagramm: RIC I (second edition) Augustus 9A: Münzkabinett Berlin. 18202546, <https://ikmk.smb.museum/object?id=18202546> (14.09.2021);

RIC I (second edition) Augustus 412: American Numismatic Society. Silver Denarius of Augustus, Rome, 12 BC 1937.158.395, <http://numismatics.org/collection/1937.158.395> (14.09.2021).

Abb. 11: American Numismatic Society. Silver Denarius of Augustus, Caesaraugusta, 19 BC - 18 BC 1944.100.39040, <http://numismatics.org/collection/1944.100.39040> (27.07.2021).

Abb. 12: Münzkabinett Berlin. 18202546, <https://ikmk.smb.museum/object?id=18202546> (14.09.2021).

Abb. 14: American Numismatic Society. Bronze AE1 of Maximian, Trier, AD 294 1944.100.5872, <http://numismatics.org/collection/1944.100.5872> (15.09.2021).

Abb. 15: American Numismatic Society. Gold Aureus of Augustus, Colonia Patricia, 18 BC 1968.39.1, <http://numismatics.org/collection/1968.39.1> (15.09.2021).

Abb. 32:

Links: American Numismatic Society. Gold Solidus of Valens, Antioch, AD 364 - AD 367 1978.64.580, <http://numismatics.org/collection/1978.64.580> (11.10.2021).

Mitte: American Numismatic Society. Gold Solidus of Valens, Antioch, AD 364 - AD 367 1965.4.6, <http://numismatics.org/collection/1965.4.6> (11.10.2021).

Rechts: American Numismatic Society. Gold Solidus of Gratian, Trier, AD 375 - AD 378 1954.237.379, <http://numismatics.org/collection/1954.237.379> (11.10.2021).

Abb. 38: Corpus Nummorum. Unveröffentlichte Münzabgüsse, <https://www.corpus-nummorum.eu> (11.10.2021).

Abb. 39:

Links: Heatmap nach: American Numismatic Society. Silver Antoninianus of Philip the Arab, Antioch, AD 244 - AD 249. 1989.117.1, <http://numismatics.org/collection/1989.117.1> (14.10.2021).

Mitte: Heatmap nach: Corpus Nummorum. cn coin 9259, <https://www.corpus-nummorum.eu/coins/9259> (14.10.2021).

Rechts: Heatmap nach: Corpus Nummorum. cn coin 8969, <https://www.corpus-nummorum.eu/coins/8969> (14.10.2021).

Abb. 40:

Links: Heatmap nach: American Numismatic Society. Silver Denarius of Severus Alexander, Rome, AD 231 - AD 235 0000.999.23499, <http://numismatics.org/collection/0000.999.23499> (14.10.2021).

Mitte links: Heatmap nach: Corpus Nummorum, cn coin 1022, <https://www.corpus-nummorum.eu/coins/1022> (14.10.2021).

Mitte rechts: Heatmap nach: Corpus Nummorum. Unveröffentlichte Münze, <https://www.corpus-nummorum.eu/> (14.10.2021).

Rechts: Heatmap nach: Corpus Nummorum. cn coin 7167, <https://www.corpus-nummorum.eu/coins/7167> (14.10.2021).

Abb. 41:

Links: Heatmap nach: American Numismatic Society. Silver Denarius of Hadrian, Rome, AD 125 - AD 128 1974.26.990, <http://numismatics.org/collection/1974.26.990> (16.10.2021).

Mitte links: Heatmap nach: Münzkabinett Wien. RÖ 88603, <https://www.ikmk.at/object?id=ID1419> (16.10.2021).

Mitte rechts: Heatmap nach: Corpus Nummorum. cn coin 31741, <https://www.corpus-nummorum.eu/coins/31741> (16.10.2021).

Abb. 42:

Links: Heatmap nach: Corpus Nummorum. Unveröffentlichte Münze, <https://www.corpus-nummorum.eu/> (16.10.2021).

Mitte: Heatmap nach: Münzkabinett. Wien. RÖ 40913, <https://www.ikmk.at/object?id=ID170820 > (16.10.2021).

Rechts: Heatmap nach: Corpus Nummorum. Unveröffentlichte Münze, <https://www.corpus-nummorum.eu> (16.10.2021).

Abb. 43:

Links: Heatmap nach: American Numismatic Society. Bronze As of Augustus, Lugdunum, 10 BC - 6 BC. 1944.100.39124, <http://numismatics.org/collection/1944.100.39124> (16.10.2021).

Mitte links: Heatmap nach: American Numismatic Society. Silver Quinarius of Augustus, Emerita, 25 BC - 23 BC. 1944.100.39026, <http://numismatics.org/collection/1944.100.39026> (16.10.2021).

Mitte rechts: American Numismatic Society. Silver Quinarius of Augustus, Emerita, 25 BC - 23 BC 1969.222.1274, <http://numismatics.org/collection/1969.222.1274> (16.10.2021).

Rechts: Corpus Nummorum. Unveröffentlichte Münze, <https://www.corpus-nummorum.eu> (16.10.2021).

Abb. 44:

Links: Heatmap nach: Münzkabinett der Universität Göttingen, Sesterz, 161 - 176 n. Chr., <https://www.kenom.de/id/record_DE-MUS-062622_kenom_186352> (16.10.2021).

Mitte links: Heatmap nach: American Numismatic Society. Silver Denarius of Antoninus Pius, Rome, AD 145 - AD 161. 1956.127.914, <http://numismatics.org/collection/1956.127.914> (16.10.2021).

Mitte rechts: Heatmap nach: Heatmap nach: Corpus Nummorum. Unveröffentlichte Münze, <https://www.corpus-nummorum.eu> (16.10.2021).

Rechts: Heatmap nach: Corpus Nummorum , cn coin 13021, <https://www.corpus-nummorum.eu/coins/13021> (16.10.2021).

Abb. 45:

Links: Heatmap nach: American Numismatic Society. Bronze AE1 of Diocletian, Cyzicus, AD 297 - AD 299. 1944.100.5749, < http://numismatics.org/collection/1944.100.5749> (16.10.2021).

Mitte: Heatmap nach: American Numismatic Society. Gold Aureus of Maximian, Aquileia, AD 294 - AD 303 1944.100.5538, <http://numismatics.org/collection/1944.100.5538> (16.10.2021).

Rechts: Heatmap nach: American Numismatic Society. Bronze AE1 of Diocletian, Cyzicus, AD 295 - AD 296. 1944.100.5715, <http://numismatics.org/collection/1944.100.5715> (16.10.2021).

Abb. 46:

Links: Heatmap nach: American Numismatic Society. Silver Denarius of Augustus, Lugdunum, 12 BC 1944.100.39098, <http://numismatics.org/collection/1944.100.39098> (16.10.2021).

Mitte: Heatmap nach: American Numismatic Society. Bronze Dupondius of Gaius/Caligula, Rome, AD 37 - AD 41. 1957.172.1512, <http://numismatics.org/collection/1957.172.1512> (16.10.2021).

Rechts: Heatmap nach: American Numismatic Society. Silver Denarius of Augustus, Lugdunum, 15 BC - 13 BC 1944.100.39097, <http://numismatics.org/collection/1944.100.39097> (16.10.2021).

Abb. 47:

Links: Heatmap nach: Münzsammlung des Seminars für Alte Geschichte der Albert-Ludwigs-Universität Freiburg i. Br. 08057, <https://ikmk.uni-freiburg.de/object?id=ID9242> (16.10.2021).

Mitte: Heatmap nach: American Numismatic Society. Bronze AE3 of Licinius, Rome, AD 317 - AD 318. 1944.100.6294, <http://numismatics.org/collection/1944.100.6294> (16.10.2021).

Rechts: Heatmap nach: American Numismatic Society. Bronze AE3 of Constantine I, Arelate, AD 316 - AD 317 1944.100.11192, <http://numismatics.org/collection/1944.100.11192> (16.10.2021).

Abb. 48:

Links: Heatmap nach: American Numismatic Society. Bronze AE3 of Licinius, Thessalonica, AD 312 - AD 313. 1944.100.3781, <http://numismatics.org/collection/1944.100.3781> (16.10.2021).

Mitte links: Heatmap nach: American Numismatic Society. Bronze AE3 of Licinius, Thessalonica, AD 312 - AD 313. 1944.100.3781, <http://numismatics.org/collection/1944.100.3781> (16.10.2021).

Mitte rechts: Heatmap nach: American Numismatic Society. Bronze AE3 of Constantius II, Constantinople, AD 347 - AD 348. 1944.100.23750, <http://numismatics.org/collection/1944.100.23750> (16.10.2021).

Rechts: Heatmap nach: American Numismatic Society. Bronze AE3 of Constantius II, Antioch, AD 347 - AD 348. 1944.100.22279, <http://numismatics.org/collection/1944.100.22279> (16.10.2021).

9. Anhang

9.1 MySQL Abfragen

Listing 1: Abfrage zur Anzeige der 66 trainierten Kaiserinnen und Kaiser auf dem CNT Datenbank Dump

```
SELECT i.ImageID, i.CoinID, i.ObverseImageFilename, i.Path, i.ObjectType, p.PersonID, per.PersonName, count(per.PersonName)
FROM Images i, PersonsHelper p, Persons per
WHERE i.CoinID IN (SELECT a.CoinID from Images a GROUP BY a.CoinID HAVING COUNT(a.CoinID) = 1) AND i.CoinID = p.CoinID AND p.FunctionID = '4'
AND p.PersonID = per.PersonID AND per.Position = 'Roman Emperor' AND per.PersonName IN ('Antoninus Pius', 'Arcadius', 'Augustus', 'Aurelian', 'Caracalla', 'Carinus', 'Claudius', 'Claudius II. Gothicus', 'Commodus', 'Constans', 'Constantine I.', 'Constantine II.', 'Constantius Chlorus', 'Constantius Gallus', 'Constantius II.', 'Cornelia Salonina', 'Crispus', 'Diocletian', 'Domitian', 'Elagabal', 'Faustina I.', 'Faustina II.', 'Galeria Valeria', 'Galerius', 'Gallienus', 'Geta', 'Gordian III.', 'Gratian', 'Hadrian', 'Helena', 'Honorius', 'Iulia Domna', 'Iulia Mamaea', 'Iulian Apostata', 'Licinius', 'Licinius II.', 'Lucilla', 'Lucius Verus', 'Magnentius', 'Marcus Aurelius', 'Maxentius', 'Maximian', 'Maximinus Daia', 'Maximinus Thrax', 'Nero', 'Nerva', 'Philippus Arabs', 'Philippus II.', 'Postumus', 'Probus', 'Septimius Severus', 'Severus Alexander', 'Tetricus I.', 'Tetricus II.', 'Theodosius I.', 'Theodosius II.', 'Tiberius', 'Titus', 'Trajan', 'Trajan Decius', 'Trebonianus Gallus', 'Valens', 'Valentinian I.', 'Valentinian II.', 'Vespasian', 'Victorinus') GROUP BY per.PersonName ORDER BY per.PersonName;
```

9.2 Confusion Matrizen

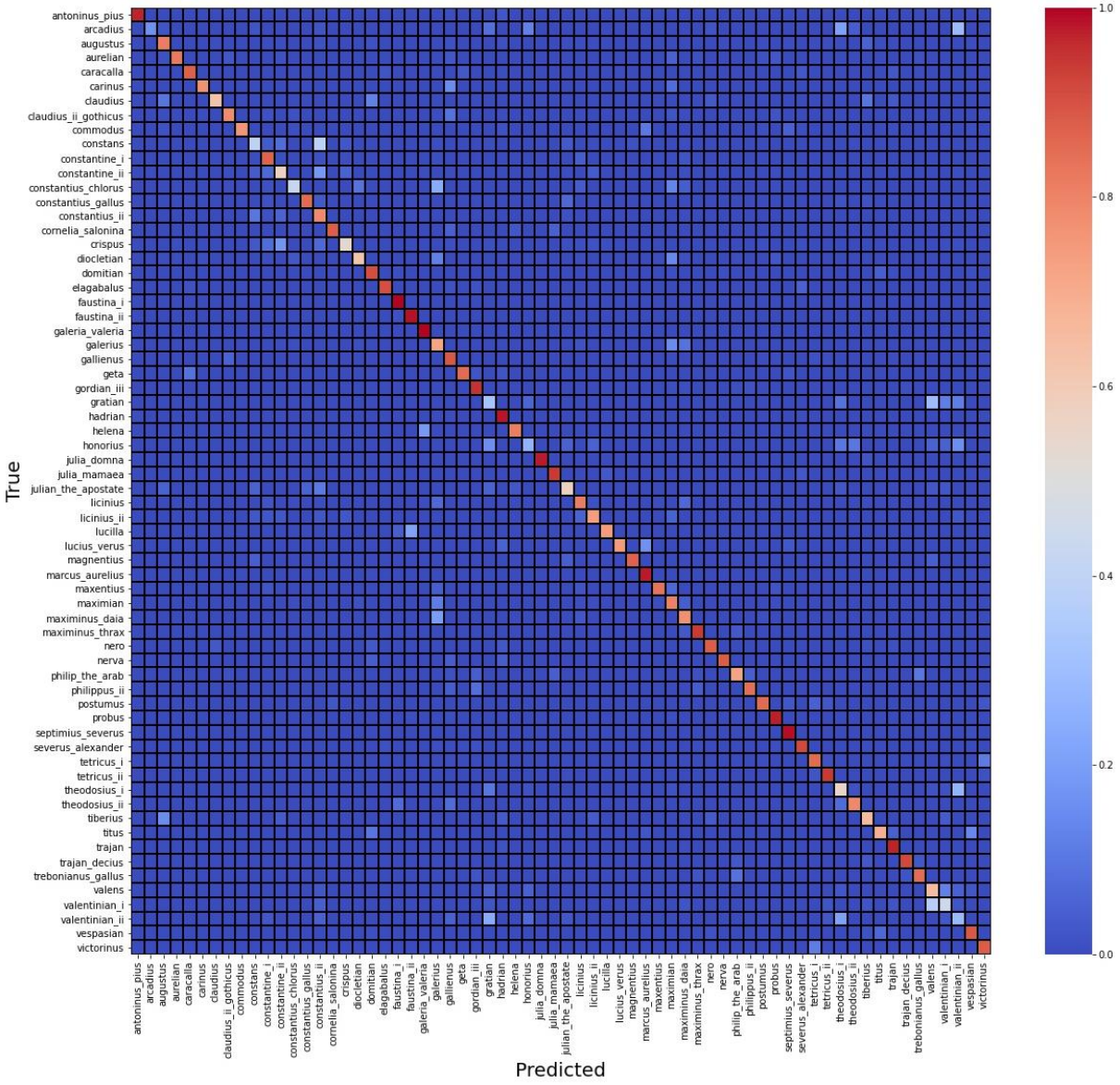

Abb. 49: Confusion Matrix des Modells 2.

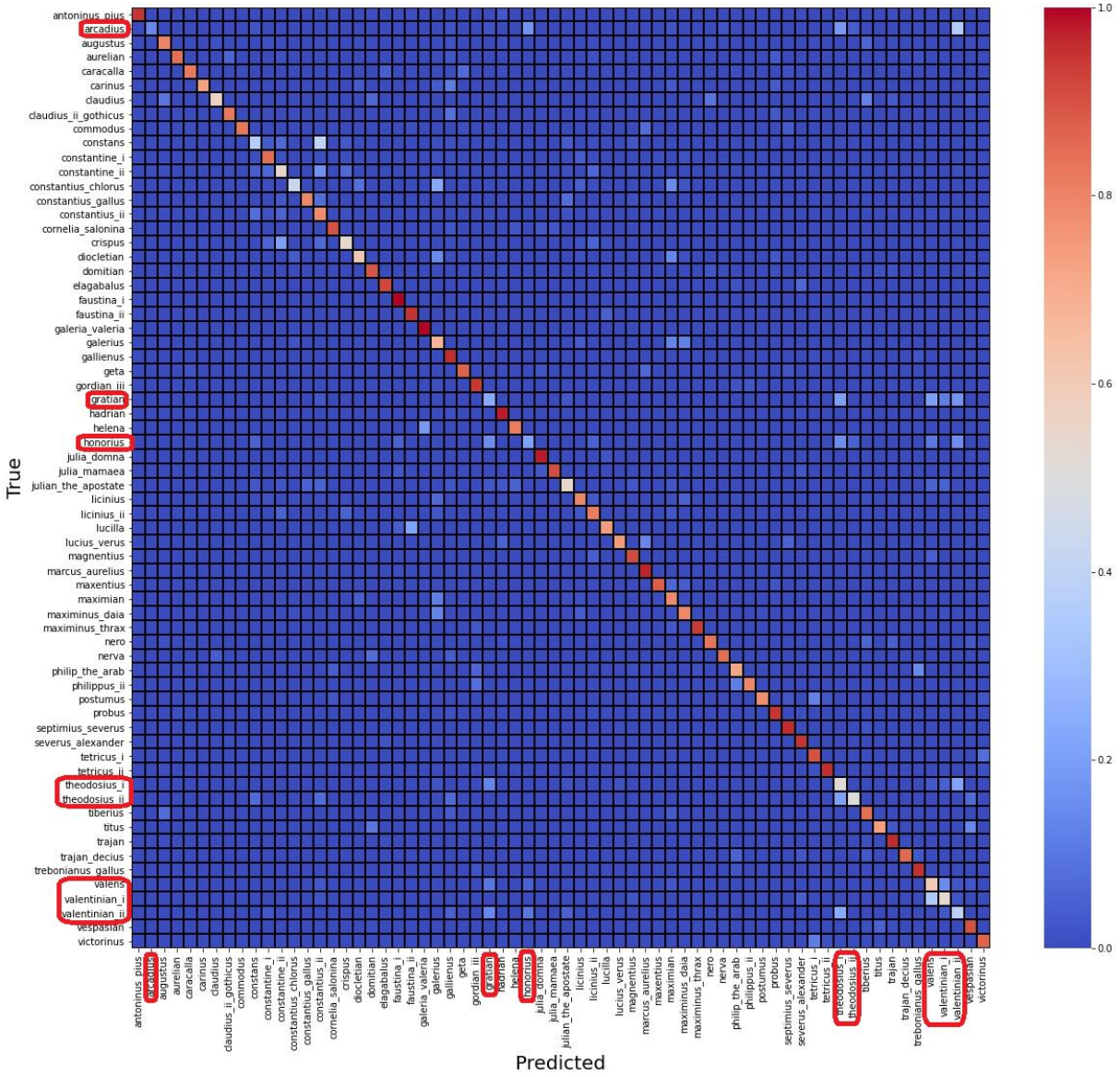

Abb. 50: Confusion Matrix des Modells 3. Die Mitglieder der valentinianischen und der theodosianischen Dynastie sind rot markiert.

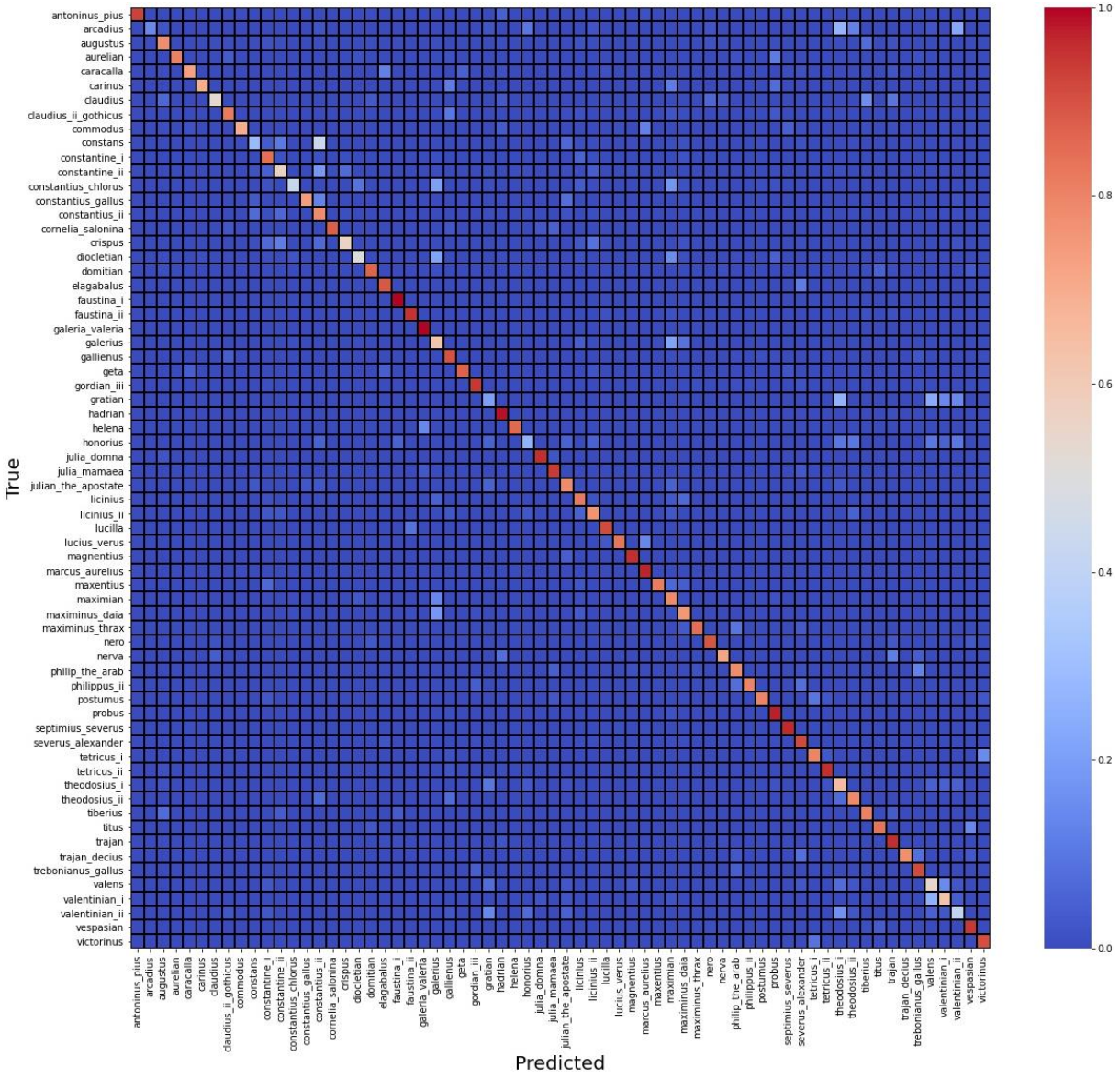

Abb. 51: Confusion Matrix des Modells 5.

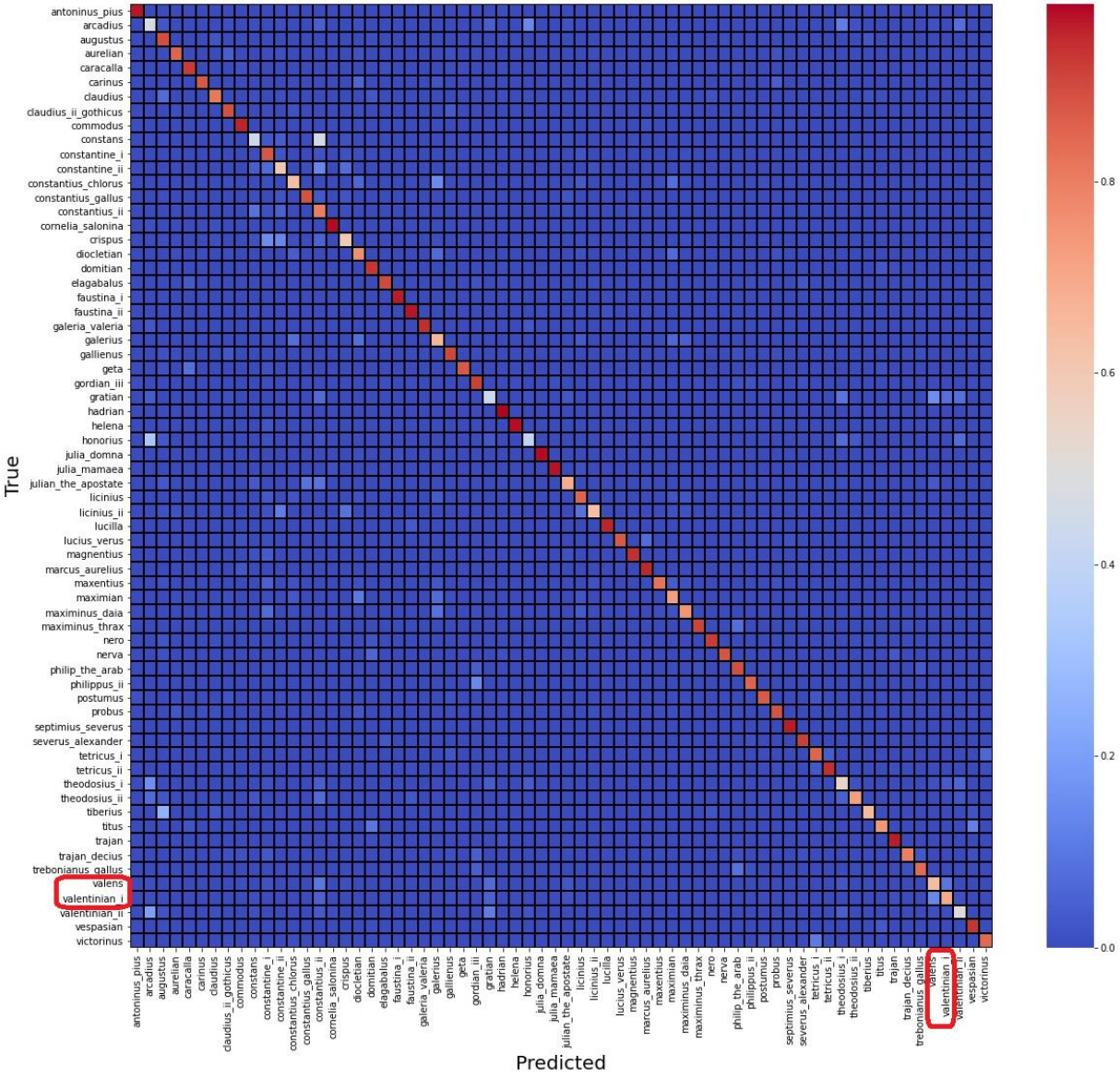

Abb. 52: Confusion Matrix des Modells 6. Die Kaiser der Valantianischen Dynastie, deren Werte sich leicht verbessert haben, sind rot markiert.

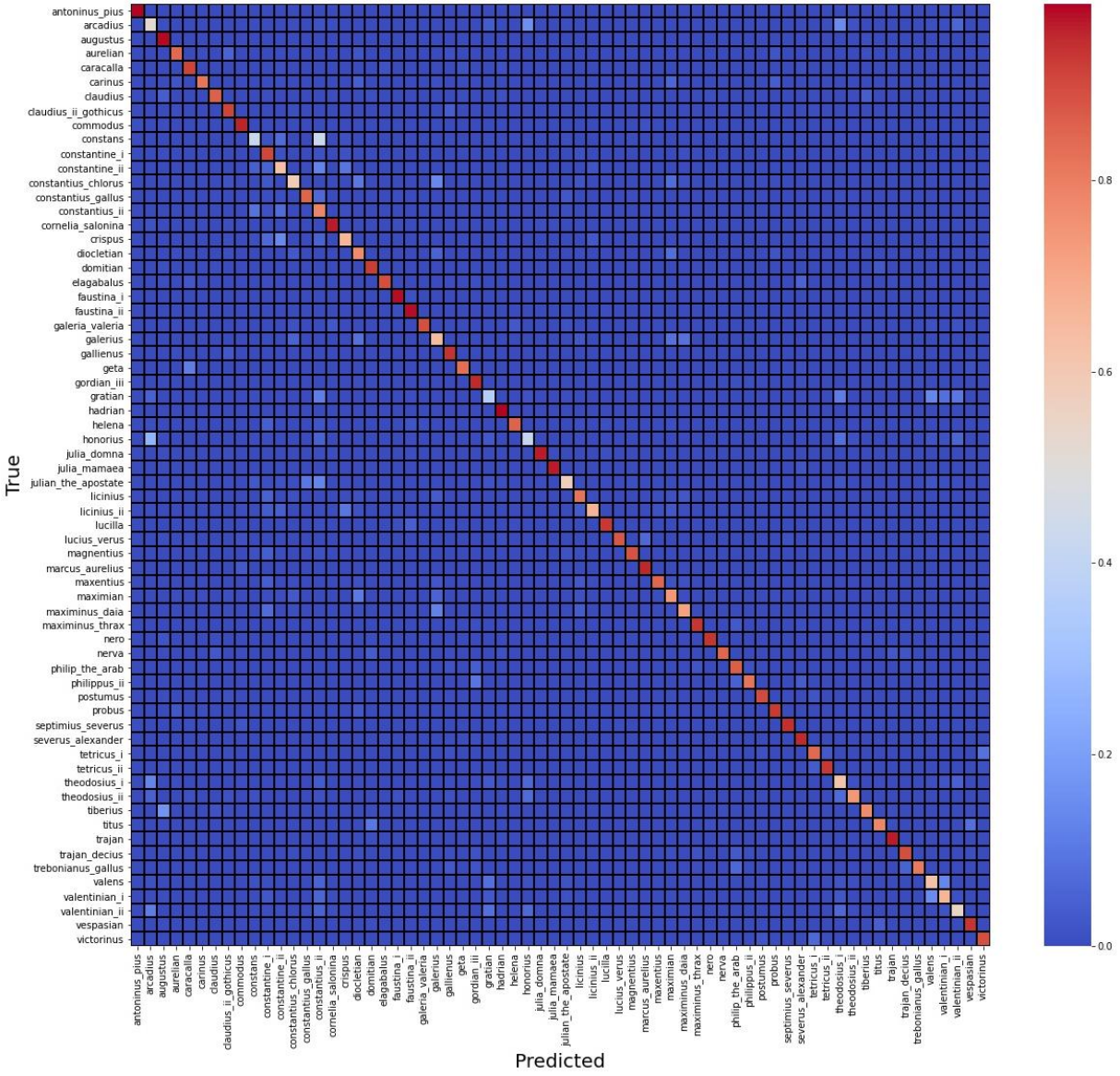

Abb. 53: Confusion Matrix des Modells 7.

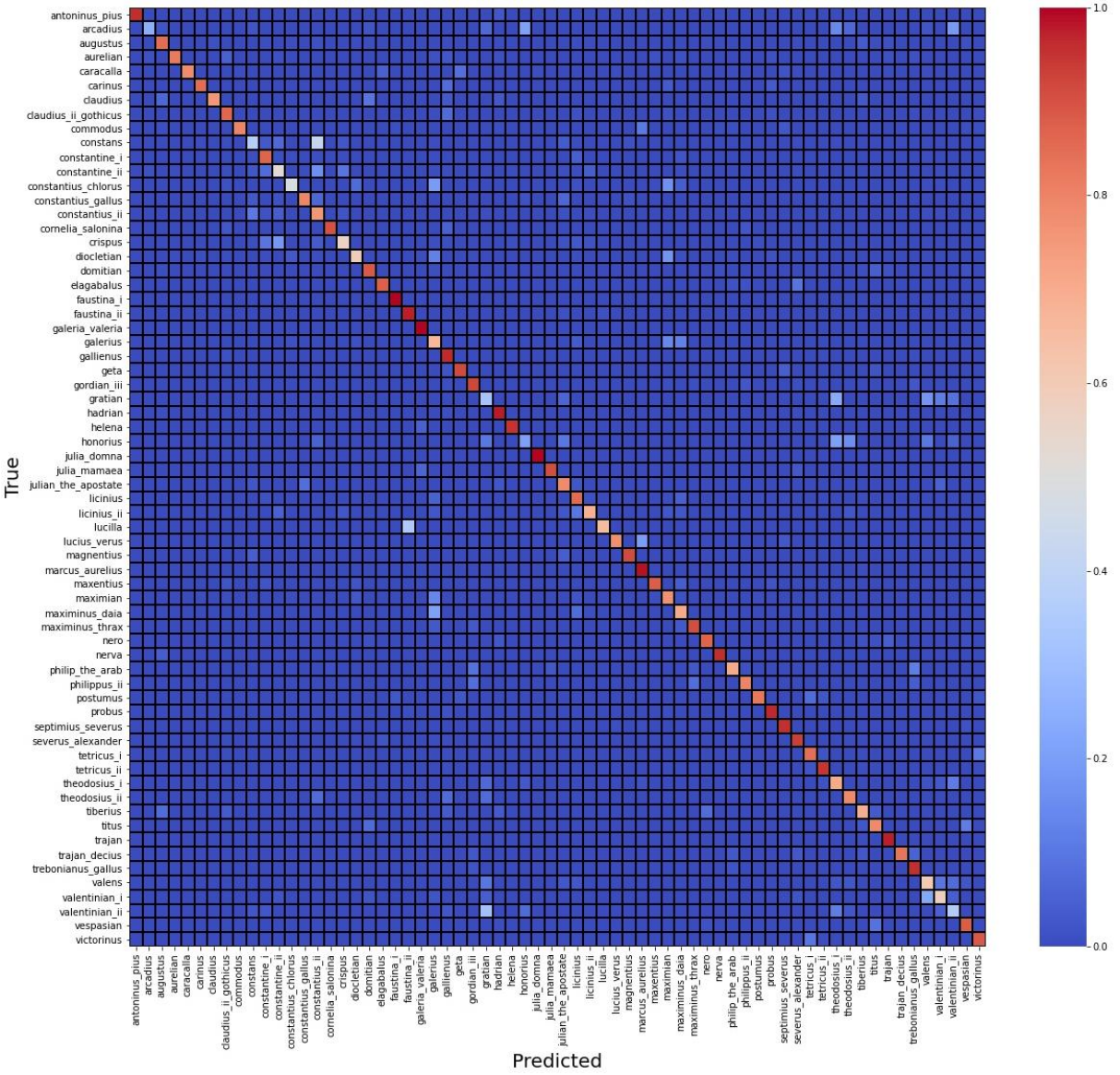

Abb. 54: Confusion Matrix des Modells 8.

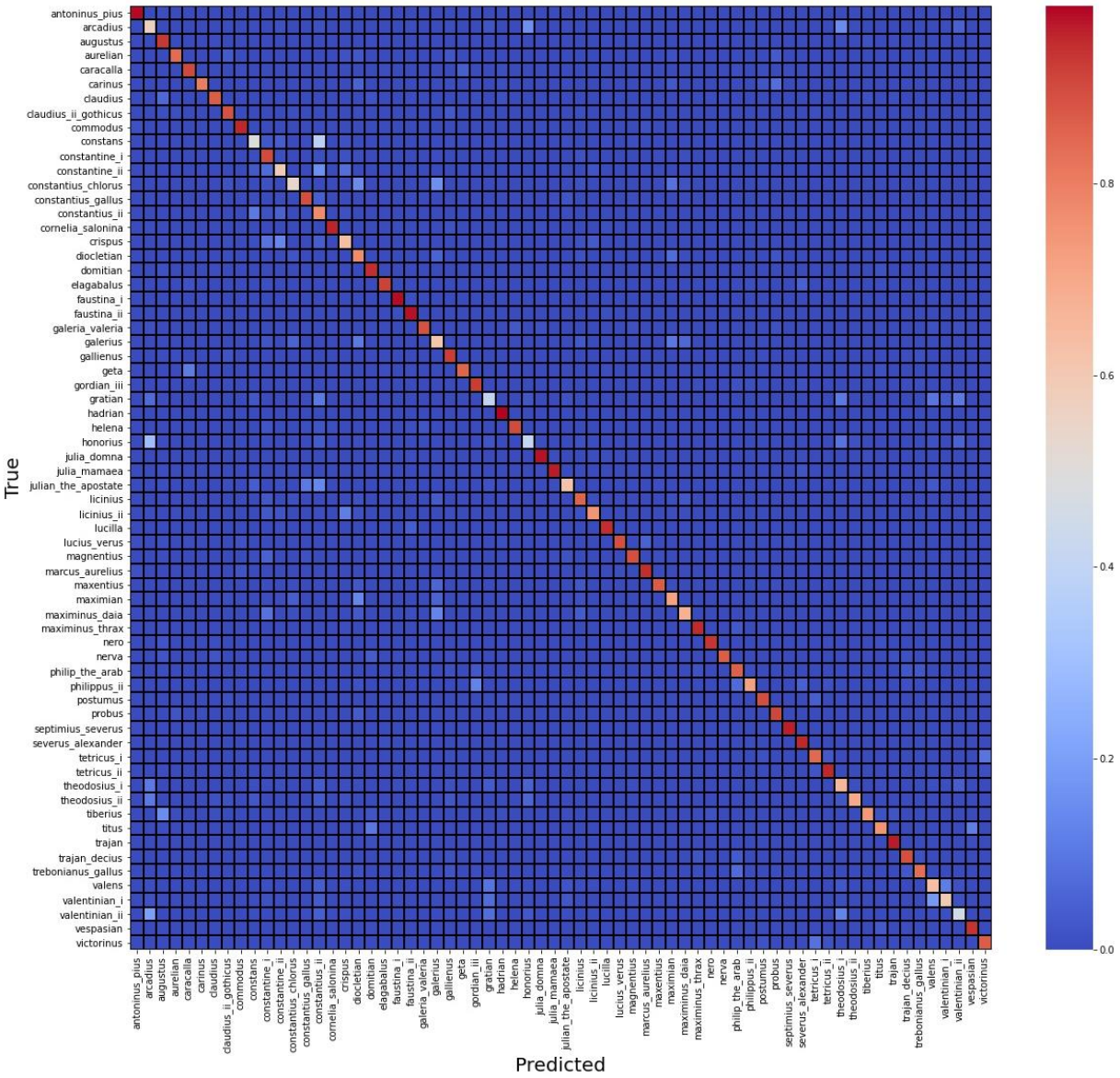

Abb. 55: Confusion Matrix des Modells 13.

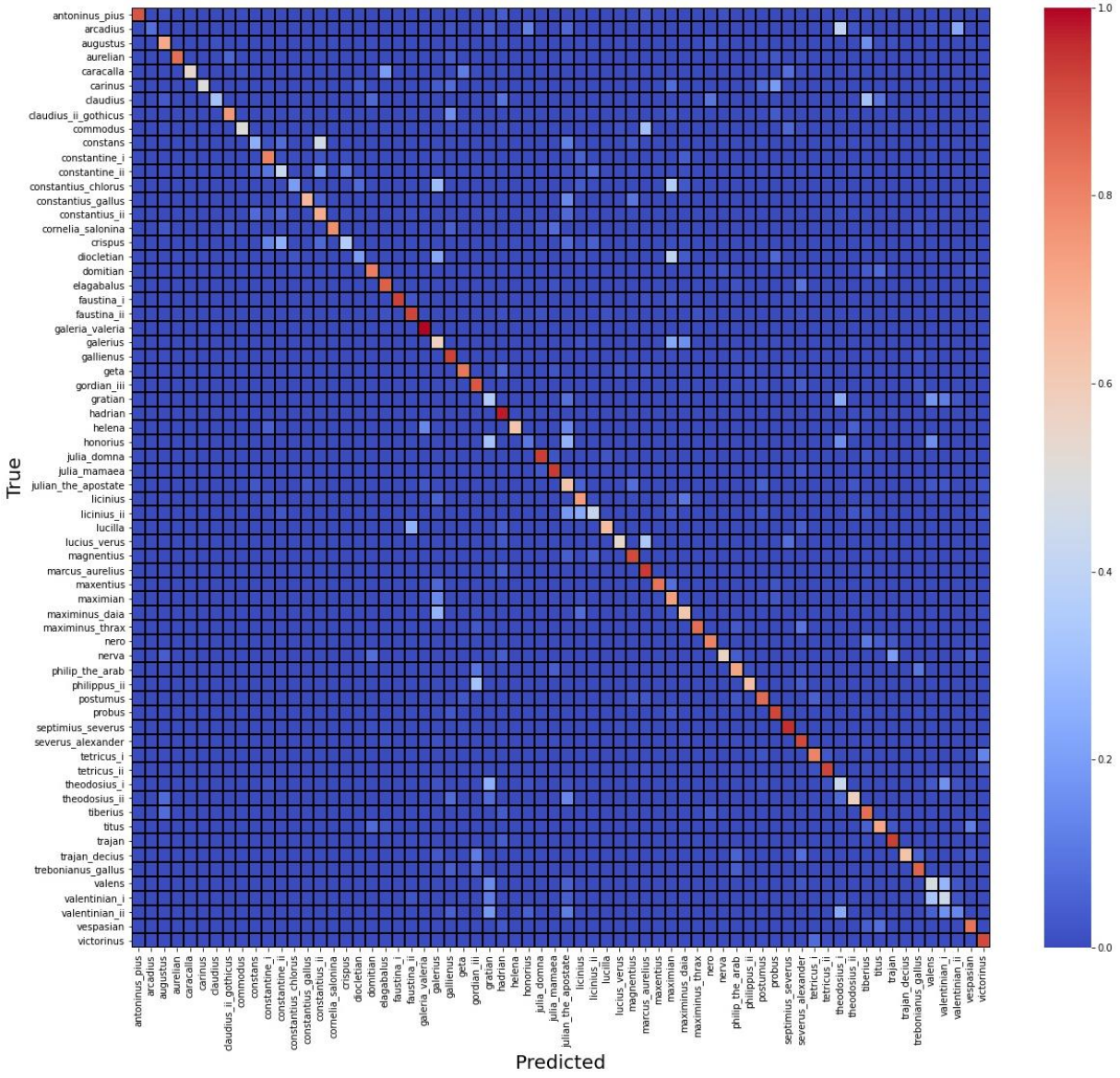

Abb. 56: Confusion Matrix des Modells 14.

9.3 Auflistungen der Ausgaben des Modells

Auflistung 1: Testbilder der Bildmenge 1.1 auf Modell 4 angewandt:

```
antoninus_pius: Files: 169, correct predictions: 4, Top-5 correct:61
arcadius: Files: 48, correct predictions: 0, Top-5 correct:4
augustus: Files: 168, correct predictions: 120, Top-5 correct:157
aurelian: Files: 82, correct predictions: 3, Top-5 correct:17
caracalla: Files: 140, correct predictions: 7, Top-5 correct:33
carinus: Files: 26, correct predictions: 0, Top-5 correct:1
claudius: Files: 32, correct predictions: 9, Top-5 correct:25
claudius_ii_gothicus: Files: 137, correct predictions: 27, Top-5 correct:79
commodus: Files: 103, correct predictions: 0, Top-5 correct:3
constans: Files: 160, correct predictions: 26, Top-5 correct:85
constantine_i: Files: 603, correct predictions: 59, Top-5 correct:265
constantine_ii: Files: 220, correct predictions: 57, Top-5 correct:151
constantius_chlorus: Files: 141, correct predictions: 14, Top-5 correct:64
constantius_gallus: Files: 62, correct predictions: 0, Top-5 correct:8
constantius_ii: Files: 480, correct predictions: 94, Top-5 correct:294
cornelia_salonina: Files: 40, correct predictions: 0, Top-5 correct:0
crispus: Files: 103, correct predictions: 13, Top-5 correct:51
diocletian: Files: 225, correct predictions: 24, Top-5 correct:125
domitian: Files: 111, correct predictions: 34, Top-5 correct:84
elagabalus: Files: 63, correct predictions: 0, Top-5 correct:0
faustina_i: Files: 85, correct predictions: 35, Top-5 correct:72
faustina_ii: Files: 80, correct predictions: 8, Top-5 correct:29
galeria_valeria: Files: 21, correct predictions: 0, Top-5 correct:0
galerius: Files: 215, correct predictions: 78, Top-5 correct:197
gallienus: Files: 198, correct predictions: 25, Top-5 correct:99
geta: Files: 46, correct predictions: 0, Top-5 correct:10
gordian_iii: Files: 124, correct predictions: 0, Top-5 correct:0
gratian: Files: 34, correct predictions: 1, Top-5 correct:5
hadrian: Files: 217, correct predictions: 37, Top-5 correct:143
helena: Files: 21, correct predictions: 0, Top-5 correct:1
honorius: Files: 19, correct predictions: 0, Top-5 correct:4
julia_domna: Files: 46, correct predictions: 2, Top-5 correct:14
julia_mamaea: Files: 33, correct predictions: 0, Top-5 correct:0
julian_the_apostate: Files: 37, correct predictions: 7, Top-5 correct:16
licinius: Files: 240, correct predictions: 54, Top-5 correct:188
licinius_ii: Files: 38, correct predictions: 6, Top-5 correct:24
lucilla: Files: 23, correct predictions: 0, Top-5 correct:1
lucius_verus: Files: 35, correct predictions: 2, Top-5 correct:11
magnentius: Files: 23, correct predictions: 0, Top-5 correct:0
marcus_aurelius: Files: 134, correct predictions: 6, Top-5 correct:31
maxentius: Files: 48, correct predictions: 22, Top-5 correct:41
maximian: Files: 202, correct predictions: 47, Top-5 correct:161
maximinus_daia: Files: 120, correct predictions: 10, Top-5 correct:54
maximinus_thrax: Files: 32, correct predictions: 0, Top-5 correct:6
nero: Files: 66, correct predictions: 33, Top-5 correct:56
nerva: Files: 25, correct predictions: 0, Top-5 correct:5
philip_the_arab: Files: 68, correct predictions: 0, Top-5 correct:1
philippus_ii: Files: 25, correct predictions: 0, Top-5 correct:0
postumus: Files: 40, correct predictions: 0, Top-5 correct:1
probus: Files: 114, correct predictions: 82, Top-5 correct:111
septimius_severus: Files: 112, correct predictions: 2, Top-5 correct:19
severus_alexander: Files: 133, correct predictions: 0, Top-5 correct:3
tetricus_i: Files: 178, correct predictions: 33, Top-5 correct:139
tetricus_ii: Files: 134, correct predictions: 37, Top-5 correct:114
theodosius_i: Files: 60, correct predictions: 3, Top-5 correct:9
theodosius_ii: Files: 14, correct predictions: 0, Top-5 correct:4
```

```
tiberius: Files: 26, correct predictions: 12, Top-5 correct:24
titus: Files: 42, correct predictions: 17, Top-5 correct:39
trajan: Files: 192, correct predictions: 7, Top-5 correct:35
trajan_decius: Files: 35, correct predictions: 0, Top-5 correct:0
trebonianus_gallus: Files: 45, correct predictions: 0, Top-5 correct:2
valens: Files: 40, correct predictions: 0, Top-5 correct:1
valentinian_i: Files: 43, correct predictions: 0, Top-5 correct:2
valentinian_ii: Files: 42, correct predictions: 0, Top-5 correct:7
vespasian: Files: 109, correct predictions: 86, Top-5 correct:105
victorinus: Files: 141, correct predictions: 17, Top-5 correct:96
All files: 6868, Top-1 correct: 1160, Top-5 correct:3387
Top-1 Accuracy: 0.168899242865463
Top-5 Accuracy: 0.49315666860803725
```

Auflistung 2: Testbilder der Bildmenge 2.3 auf Modell 13 angewandt:

```
antoninus_pius: Files: 1130, correct predictions: 1109, Top-5 correct:1126
arcadius: Files: 223, correct predictions: 125, Top-5 correct:214
augustus: Files: 897, correct predictions: 836, Top-5 correct:892
aurelian: Files: 316, correct predictions: 265, Top-5 correct:307
caracalla: Files: 436, correct predictions: 393, Top-5 correct:433
carinus: Files: 99, correct predictions: 79, Top-5 correct:96
claudius: Files: 225, correct predictions: 195, Top-5 correct:222
claudius_ii_gothicus: Files: 624, correct predictions: 555, Top-5
correct:609
commodus: Files: 1369, correct predictions: 1305, Top-5 correct:1358
constans: Files: 482, correct predictions: 238, Top-5 correct:471
constantine_i: Files: 2291, correct predictions: 2061, Top-5 correct:2257
constantine_ii: Files: 711, correct predictions: 419, Top-5 correct:691
constantius_chlorus: Files: 430, correct predictions: 229, Top-5
correct:417
constantius_gallus: Files: 195, correct predictions: 175, Top-5 correct:191
constantius_ii: Files: 1600, correct predictions: 1232, Top-5 correct:1584
cornelia_salonina: Files: 183, correct predictions: 175, Top-5 correct:182
crispus: Files: 442, correct predictions: 281, Top-5 correct:430
diocletian: Files: 826, correct predictions: 639, Top-5 correct:819
domitian: Files: 734, correct predictions: 693, Top-5 correct:729
elagabalus: Files: 247, correct predictions: 226, Top-5 correct:246
faustina_i: Files: 535, correct predictions: 522, Top-5 correct:532
faustina_ii: Files: 478, correct predictions: 464, Top-5 correct:476
galeria_valeria: Files: 72, correct predictions: 64, Top-5 correct:70
galerius: Files: 645, correct predictions: 394, Top-5 correct:640
gallienus: Files: 951, correct predictions: 878, Top-5 correct:941
geta: Files: 144, correct predictions: 123, Top-5 correct:142
gordian_iii: Files: 473, correct predictions: 436, Top-5 correct:467
gratian: Files: 169, correct predictions: 68, Top-5 correct:157
hadrian: Files: 2225, correct predictions: 2194, Top-5 correct:2222
helena: Files: 90, correct predictions: 81, Top-5 correct:89
honorius: Files: 132, correct predictions: 54, Top-5 correct:115
julia_domna: Files: 210, correct predictions: 204, Top-5 correct:208
julia_mamaea: Files: 137, correct predictions: 132, Top-5 correct:134
julian_the_apostate: Files: 179, correct predictions: 111, Top-5
correct:162
licinius: Files: 686, correct predictions: 586, Top-5 correct:673
licinius_ii: Files: 135, correct predictions: 101, Top-5 correct:124
lucilla: Files: 181, correct predictions: 171, Top-5 correct:178
lucius_verus: Files: 218, correct predictions: 194, Top-5 correct:215
magnentius: Files: 173, correct predictions: 154, Top-5 correct:170
marcus_aurelius: Files: 1166, correct predictions: 1104, Top-5 correct:1162
maxentius: Files: 169, correct predictions: 147, Top-5 correct:168
maximian: Files: 721, correct predictions: 517, Top-5 correct:710
```

```
maximinus_daia: Files: 372, correct predictions: 251, Top-5 correct:363
maximinus_thrax: Files: 178, correct predictions: 169, Top-5 correct:178
nero: Files: 415, correct predictions: 388, Top-5 correct:411
nerva: Files: 138, correct predictions: 120, Top-5 correct:134
philip_the_arab: Files: 289, correct predictions: 249, Top-5 correct:284
philippus_ii: Files: 84, correct predictions: 60, Top-5 correct:84
postumus: Files: 299, correct predictions: 268, Top-5 correct:295
probus: Files: 429, correct predictions: 389, Top-5 correct:426
septimius_severus: Files: 551, correct predictions: 529, Top-5 correct:546
severus_alexander: Files: 535, correct predictions: 510, Top-5 correct:531
tetricus_i: Files: 600, correct predictions: 508, Top-5 correct:593
tetricus_ii: Files: 471, correct predictions: 449, Top-5 correct:470
theodosius_i: Files: 259, correct predictions: 171, Top-5 correct:242
theodosius_ii: Files: 85, correct predictions: 59, Top-5 correct:78
tiberius: Files: 183, correct predictions: 135, Top-5 correct:180
titus: Files: 309, correct predictions: 231, Top-5 correct:304
trajan: Files: 919, correct predictions: 886, Top-5 correct:914
trajan_decius: Files: 130, correct predictions: 116, Top-5 correct:127
trebonianus_gallus: Files: 140, correct predictions: 117, Top-5 correct:134
valens: Files: 227, correct predictions: 144, Top-5 correct:215
valentinian_i: Files: 216, correct predictions: 128, Top-5 correct:198
valentinian_ii: Files: 178, correct predictions: 80, Top-5 correct:164
vespasian: Files: 749, correct predictions: 703, Top-5 correct:748
victorinus: Files: 492, correct predictions: 427, Top-5 correct:484
All files: 31597, Top-1 correct: 27016, Top-5 correct:31132
Top-1 Accuracy: 0.8550178814444409
Top-5 Accuracy: 0.9852834129822452
```

Precision, Recall und F1-Score Werte:

	precision	recall	f1-score	support
antoninus_pius	0.96	0.98	0.97	1130
arcadius	0.47	0.56	0.51	223
augustus	0.86	0.93	0.90	897
aurelian	0.91	0.84	0.87	316
caracalla	0.91	0.90	0.91	436
carinus	0.93	0.80	0.86	99
claudius	0.89	0.87	0.88	225
claudius_ii_gothicus	0.88	0.89	0.88	624
commodus	0.97	0.95	0.96	1369
constans	0.50	0.49	0.50	482
constantine_i	0.90	0.90	0.90	2291
constantine_ii	0.67	0.59	0.62	711
constantius_chlorus	0.69	0.53	0.60	430
constantius_gallus	0.80	0.90	0.84	195
constantius_ii	0.74	0.77	0.75	1600
cornelia_salonina	0.97	0.96	0.96	183
crispus	0.65	0.64	0.64	442
diocletian	0.67	0.77	0.72	826
domitian	0.91	0.94	0.93	734
elagabalus	0.95	0.91	0.93	247
faustina_i	0.98	0.98	0.98	535
faustina_ii	0.94	0.97	0.95	478
galeria_valeria	0.96	0.89	0.92	72
galerius	0.61	0.61	0.61	645
gallienus	0.92	0.92	0.92	951
geta	0.82	0.85	0.84	144
gordian_iii	0.94	0.92	0.93	473
gratian	0.46	0.40	0.43	169
hadrian	0.98	0.99	0.98	2225

```
              helena       0.94      0.90      0.92        90
            honorius       0.43      0.41      0.42       132
         julia_domna       0.97      0.97      0.97       210
        julia_mamaea       0.95      0.96      0.96       137
julian_the_apostate       0.74      0.62      0.67       179
            licinius       0.83      0.85      0.84       686
         licinius_ii       0.68      0.75      0.71       135
             lucilla       0.92      0.94      0.93       181
        lucius_verus       0.92      0.89      0.90       218
          magnentius       0.86      0.89      0.87       173
    marcus_aurelius       0.96      0.95      0.96      1166
           maxentius       0.88      0.87      0.87       169
            maximian       0.73      0.72      0.72       721
     maximinus_daia        0.73      0.67      0.70       372
    maximinus_thrax       0.94      0.95      0.94       178
               nero       0.96      0.93      0.95       415
              nerva       0.93      0.87      0.90       138
    philip_the_arab       0.85      0.86      0.86       289
       philippus_ii       0.77      0.71      0.74        84
           postumus       0.88      0.90      0.89       299
             probus       0.88      0.91      0.90       429
  septimius_severus       0.98      0.96      0.97       551
  severus_alexander       0.93      0.95      0.94       535
        tetricus_i        0.87      0.85      0.86       600
       tetricus_ii        0.91      0.95      0.93       471
      theodosius_i        0.63      0.66      0.64       259
     theodosius_ii        0.88      0.69      0.78        85
           tiberius       0.78      0.74      0.76       183
              titus       0.82      0.75      0.78       309
             trajan       0.97      0.96      0.97       919
      trajan_decius       0.77      0.89      0.83       130
 trebonianus_gallus       0.81      0.84      0.82       140
             valens       0.65      0.63      0.64       227
      valentinian_i       0.73      0.59      0.65       216
     valentinian_ii       0.58      0.45      0.51       178
           vespasian       0.92      0.94      0.93       749
          victorinus       0.86      0.87      0.87       492

           accuracy                           0.86     31597
          macro avg       0.83      0.82      0.82     31597
       weighted avg       0.85      0.86      0.85     31597
```

Auflistung 3: Bilder der Bildmengen 3.1 und 3.2 auf Modell 13 angewandt:

Bildmenge 3.1:

```
domitian: Files: 41, correct predictions: 8, Top-5 correct:20
gallienus: Files: 63, correct predictions: 7, Top-5 correct:14
hadrian: Files: 19, correct predictions: 4, Top-5 correct:8
marcus_aurelius: Files: 68, correct predictions: 16, Top-5 correct:41
nero: Files: 33, correct predictions: 2, Top-5 correct:13
philip_the_arab: Files: 8, correct predictions: 1, Top-5 correct:5
septimius_severus: Files: 155, correct predictions: 53, Top-5 correct:119
tiberius: Files: 2, correct predictions: 0, Top-5 correct:2
trajan: Files: 107, correct predictions: 9, Top-5 correct:41
valerian: Files: 25, correct predictions: 0, Top-5 correct:0
All files: 521, Top-1 correct: 100, Top-5 correct:263
Top-1 Accuracy: 0.19193857965451055
Top-5 Accuracy: 0.5047984644913628
```

Bildmenge 3.2:

```
antoninus_pius: Files: 731, correct predictions: 285, Top-5 correct:483
augustus: Files: 259, correct predictions: 167, Top-5 correct:243
caracalla: Files: 1731, correct predictions: 357, Top-5 correct:898
claudius: Files: 37, correct predictions: 6, Top-5 correct:22
commodus: Files: 665, correct predictions: 166, Top-5 correct:394
domitian: Files: 144, correct predictions: 44, Top-5 correct:90
elagabalus: Files: 691, correct predictions: 40, Top-5 correct:163
faustina_i: Files: 17, correct predictions: 3, Top-5 correct:9
faustina_ii: Files: 313, correct predictions: 198, Top-5 correct:271
gallienus: Files: 293, correct predictions: 39, Top-5 correct:92
geta: Files: 554, correct predictions: 22, Top-5 correct:118
gordian_iii: Files: 712, correct predictions: 199, Top-5 correct:416
hadrian: Files: 224, correct predictions: 67, Top-5 correct:126
julia_domna: Files: 310, correct predictions: 146, Top-5 correct:222
julia_mamaea: Files: 113, correct predictions: 50, Top-5 correct:92
lucilla: Files: 17, correct predictions: 2, Top-5 correct:8
lucius_verus: Files: 103, correct predictions: 29, Top-5 correct:68
marcus_aurelius: Files: 332, correct predictions: 115, Top-5 correct:233
maximinus_thrax: Files: 163, correct predictions: 26, Top-5 correct:74
nero: Files: 92, correct predictions: 17, Top-5 correct:47
nerva: Files: 11, correct predictions: 1, Top-5 correct:2
philip_the_arab: Files: 135, correct predictions: 15, Top-5 correct:56
philippus_ii: Files: 65, correct predictions: 0, Top-5 correct:5
septimius_severus: Files: 938, correct predictions: 388, Top-5 correct:698
severus_alexander: Files: 563, correct predictions: 74, Top-5 correct:185
tiberius: Files: 20, correct predictions: 1, Top-5 correct:12
titus: Files: 6, correct predictions: 0, Top-5 correct:2
trajan: Files: 253, correct predictions: 34, Top-5 correct:117
trebonianus_gallus: Files: 46, correct predictions: 1, Top-5 correct:12
vespasian: Files: 24, correct predictions: 16, Top-5 correct:18
All files: 9562, Top-1 correct: 2508, Top-5 correct:5176
Top-1 Accuracy: 0.2622882242208743
Top-5 Accuracy: 0.541309349508471
```

Auflistung 4: Bilder der Bildmenge 3.2 nach Originalmünzen und Abgüssen getrennt auf Modell 13 angewandt:

Originale:

```
antoninus_pius: Files: 220, correct predictions: 127, Top-5 correct:185
augustus: Files: 217, correct predictions: 134, Top-5 correct:204
caracalla: Files: 836, correct predictions: 263, Top-5 correct:518
claudius: Files: 21, correct predictions: 3, Top-5 correct:10
commodus: Files: 404, correct predictions: 67, Top-5 correct:207
domitian: Files: 45, correct predictions: 29, Top-5 correct:40
elagabalus: Files: 234, correct predictions: 16, Top-5 correct:50
faustina_i: Files: 14, correct predictions: 1, Top-5 correct:6
faustina_ii: Files: 165, correct predictions: 108, Top-5 correct:146
gallienus: Files: 214, correct predictions: 26, Top-5 correct:63
geta: Files: 263, correct predictions: 22, Top-5 correct:93
gordian_iii: Files: 496, correct predictions: 123, Top-5 correct:261
hadrian: Files: 107, correct predictions: 41, Top-5 correct:68
julia_domna: Files: 173, correct predictions: 102, Top-5 correct:142
julia_mamaea: Files: 73, correct predictions: 32, Top-5 correct:63
lucilla: Files: 4, correct predictions: 1, Top-5 correct:3
lucius_verus: Files: 65, correct predictions: 19, Top-5 correct:48
marcus_aurelius: Files: 197, correct predictions: 83, Top-5 correct:154
maximinus_thrax: Files: 125, correct predictions: 23, Top-5 correct:68
```

```
nero: Files: 56, correct predictions: 16, Top-5 correct:33
nerva: Files: 5, correct predictions: 1, Top-5 correct:1
philip_the_arab: Files: 116, correct predictions: 11, Top-5 correct:43
philippus_ii: Files: 52, correct predictions: 0, Top-5 correct:4
septimius_severus: Files: 337, correct predictions: 106, Top-5 correct:224
severus_alexander: Files: 365, correct predictions: 54, Top-5 correct:117
tiberius: Files: 19, correct predictions: 1, Top-5 correct:11
titus: Files: 6, correct predictions: 0, Top-5 correct:2
trajan: Files: 127, correct predictions: 22, Top-5 correct:67
trebonianus_gallus: Files: 32, correct predictions: 1, Top-5 correct:7
vespasian: Files: 15, correct predictions: 11, Top-5 correct:12
All files: 5003, Top-1 correct: 1443, Top-5 correct:2850
Top-1 Accuracy: 0.28842694383369977
Top-5 Accuracy: 0.5696582050769539
```

Abgüsse:

```
antoninus_pius: Files: 511, correct predictions: 158, Top-5 correct:298
augustus: Files: 42, correct predictions: 33, Top-5 correct:39
caracalla: Files: 895, correct predictions: 94, Top-5 correct:380
claudius: Files: 16, correct predictions: 3, Top-5 correct:12
commodus: Files: 261, correct predictions: 99, Top-5 correct:187
domitian: Files: 99, correct predictions: 15, Top-5 correct:50
elagabalus: Files: 457, correct predictions: 24, Top-5 correct:113
faustina_i: Files: 3, correct predictions: 2, Top-5 correct:3
faustina_ii: Files: 148, correct predictions: 90, Top-5 correct:125
gallienus: Files: 79, correct predictions: 13, Top-5 correct:29
geta: Files: 291, correct predictions: 0, Top-5 correct:25
gordian_iii: Files: 216, correct predictions: 76, Top-5 correct:155
hadrian: Files: 117, correct predictions: 26, Top-5 correct:58
julia_domna: Files: 137, correct predictions: 44, Top-5 correct:80
julia_mamaea: Files: 40, correct predictions: 18, Top-5 correct:29
lucilla: Files: 13, correct predictions: 1, Top-5 correct:5
lucius_verus: Files: 38, correct predictions: 10, Top-5 correct:20
marcus_aurelius: Files: 135, correct predictions: 32, Top-5 correct:79
maximinus_thrax: Files: 38, correct predictions: 3, Top-5 correct:6
nero: Files: 36, correct predictions: 1, Top-5 correct:14
nerva: Files: 6, correct predictions: 0, Top-5 correct:1
philip_the_arab: Files: 19, correct predictions: 4, Top-5 correct:13
philippus_ii: Files: 13, correct predictions: 0, Top-5 correct:1
septimius_severus: Files: 601, correct predictions: 282, Top-5 correct:474
severus_alexander: Files: 198, correct predictions: 20, Top-5 correct:68
tiberius: Files: 1, correct predictions: 0, Top-5 correct:1
trajan: Files: 126, correct predictions: 12, Top-5 correct:50
trebonianus_gallus: Files: 14, correct predictions: 0, Top-5 correct:5
vespasian: Files: 9, correct predictions: 5, Top-5 correct:6
All files: 4559, Top-1 correct: 1065, Top-5 correct:2326
Top-1 Accuracy: 0.23360386049572274
Top-5 Accuracy: 0.5101996051765738
```

Auflistung 5: Vollständige Münzbilder der Testmenge von A. Loyal auf Modell 13 angewandt:

```
antoninus_pius: Files: 169, correct predictions: 3, Top-5 correct:33
arcadius: Files: 48, correct predictions: 13, Top-5 correct:38
augustus: Files: 168, correct predictions: 56, Top-5 correct:114
aurelian: Files: 82, correct predictions: 1, Top-5 correct:10
caracalla: Files: 140, correct predictions: 66, Top-5 correct:122
carinus: Files: 26, correct predictions: 0, Top-5 correct:0
claudius: Files: 32, correct predictions: 0, Top-5 correct:1
```

claudius_ii_gothicus: Files: 137, correct predictions: 36, Top-5 correct:66
commodus: Files: 103, correct predictions: 89, Top-5 correct:100
constans: Files: 160, correct predictions: 43, Top-5 correct:127
constantine_i: Files: 603, correct predictions: 517, Top-5 correct:589
constantine_ii: Files: 220, correct predictions: 9, Top-5 correct:61
constantius_chlorus: Files: 141, correct predictions: 15, Top-5 correct:77
constantius_gallus: Files: 62, correct predictions: 13, Top-5 correct:31
constantius_ii: Files: 480, correct predictions: 293, Top-5 correct:421
cornelia_salonina: Files: 40, correct predictions: 24, Top-5 correct:36
crispus: Files: 103, correct predictions: 0, Top-5 correct:4
diocletian: Files: 225, correct predictions: 0, Top-5 correct:5
domitian: Files: 111, correct predictions: 2, Top-5 correct:23
elagabalus: Files: 63, correct predictions: 3, Top-5 correct:17
faustina_i: Files: 85, correct predictions: 62, Top-5 correct:79
faustina_ii: Files: 80, correct predictions: 37, Top-5 correct:61
galeria_valeria: Files: 21, correct predictions: 17, Top-5 correct:19
galerius: Files: 215, correct predictions: 1, Top-5 correct:39
gallienus: Files: 198, correct predictions: 93, Top-5 correct:155
geta: Files: 46, correct predictions: 0, Top-5 correct:5
gordian_iii: Files: 124, correct predictions: 1, Top-5 correct:14
gratian: Files: 34, correct predictions: 14, Top-5 correct:31
hadrian: Files: 217, correct predictions: 163, Top-5 correct:198
helena: Files: 21, correct predictions: 14, Top-5 correct:19
honorius: Files: 19, correct predictions: 4, Top-5 correct:13
julia_domna: Files: 46, correct predictions: 29, Top-5 correct:42
julia_mamaea: Files: 33, correct predictions: 4, Top-5 correct:26
julian_the_apostate: Files: 37, correct predictions: 8, Top-5 correct:23
licinius: Files: 240, correct predictions: 108, Top-5 correct:211
licinius_ii: Files: 38, correct predictions: 0, Top-5 correct:0
lucilla: Files: 23, correct predictions: 2, Top-5 correct:15
lucius_verus: Files: 35, correct predictions: 15, Top-5 correct:25
magnentius: Files: 23, correct predictions: 13, Top-5 correct:20
marcus_aurelius: Files: 134, correct predictions: 18, Top-5 correct:75
maxentius: Files: 48, correct predictions: 20, Top-5 correct:46
maximian: Files: 202, correct predictions: 3, Top-5 correct:15
maximinus_daia: Files: 120, correct predictions: 0, Top-5 correct:12
maximinus_thrax: Files: 32, correct predictions: 0, Top-5 correct:0
nero: Files: 66, correct predictions: 0, Top-5 correct:9
nerva: Files: 25, correct predictions: 0, Top-5 correct:2
philip_the_arab: Files: 68, correct predictions: 10, Top-5 correct:32
philippus_ii: Files: 25, correct predictions: 0, Top-5 correct:11
postumus: Files: 40, correct predictions: 7, Top-5 correct:35
probus: Files: 114, correct predictions: 4, Top-5 correct:19
septimius_severus: Files: 112, correct predictions: 10, Top-5 correct:52
severus_alexander: Files: 133, correct predictions: 35, Top-5 correct:81
tetricus_i: Files: 178, correct predictions: 150, Top-5 correct:177
tetricus_ii: Files: 134, correct predictions: 106, Top-5 correct:130
theodosius_i: Files: 60, correct predictions: 21, Top-5 correct:53
theodosius_ii: Files: 14, correct predictions: 0, Top-5 correct:7
tiberius: Files: 26, correct predictions: 0, Top-5 correct:6
titus: Files: 42, correct predictions: 2, Top-5 correct:15
trajan: Files: 192, correct predictions: 7, Top-5 correct:32
trajan_decius: Files: 35, correct predictions: 1, Top-5 correct:13
trebonianus_gallus: Files: 45, correct predictions: 0, Top-5 correct:0
valens: Files: 40, correct predictions: 22, Top-5 correct:38
valentinian_i: Files: 43, correct predictions: 24, Top-5 correct:41
valentinian_ii: Files: 42, correct predictions: 0, Top-5 correct:4
vespasian: Files: 112, correct predictions: 78, Top-5 correct:101
victorinus: Files: 141, correct predictions: 52, Top-5 correct:134
All files: 6871, Top-1 correct: 2338, Top-5 correct:4010
Top-1 Accuracy: 0.3402707029544462
Top-5 Accuracy: 0.5836122835104061

9.4 Weitere Tabellen

Tab. 3: Eigenschaften und Größe sämtlicher in dieser Arbeit verwendeten Bildmengen

Name	Bildmenge 1.1	Bildmenge 1.2	Bildmenge 1.3	Bildmenge 2.1	Bildmenge 2.2	Bildmenge 2.3	Bildmenge 3.1	Bildmenge 3.2
Eigenschaft	Ausgeschnittene Porträts der Bildmenge von A. Loyal aus dem American Numismatic Society Bestand	Alle Porträts der Bildmenge 1.1 gespiegelt	Kombination aus Bildmenge 1.1 und 1.2	Ausgeschnittene Porträts sämtlicher OCRE Quellen	Kombination der Porträts aus Bildmenge 2.1 und der gespiegelten Porträts derselben Bildmenge	Hochrechteckige Porträts der Bildmenge 2.2 in eine quadratisches Format gebracht	Testmenge mit CN Bildern zu wenigen Klassen aus der Bachelorarbeit des Autors	Größere Testmenge mit CN Bildern zu mehr Klassen
Größe	Ca. 28.000 Bilder	Ca. 28.000 Bilder	Ca. 56.000 Bilder	Ca. 63.000 Bilder	Ca. 126.000 Bilder	Ca. 126.000 Bilder	Ca. 500 Bilder	Ca. 9.600 Bilder

9.5 Hardware und Sicherung

Die trainierten Modelle und die dafür verwendeten Jupyter Notebooks sind auf dem in Abschn. 9.3 beschriebenen Rechner der Abteilung Datenbanken und Informationssystem des Instituts für Informatik der Goethe-Universität Frankfurt am Main abgelegt.

Der Rechner besteht aus den folgenden Komponenten:

- *CPU: AMD Ryzen 5 1600 6-Kern Prozessor*
- *RAM: 16 GB*
- *GPU: NVIDIA Geforce 1080 GTX (8 GB RAM)*
- *2 * 1 TB HDDs*

Das verwendete Betriebssystem ist Ubuntu Linux in der Version 18.04.4 LTS.