Bachelorarbeit

Clusterbildung keltischer Münzen basierend auf Convolutional Neural Networks

Robin Krause

11.12.2020

Gutachter: Prof. Dr. Roberto Zicari,
Dr. Karsten Tolle

Johann Wolfgang Goethe-Universität Frankfurt am Main
Institut für Informatik
Fachbereich 12 - Informatik und Mathematik

Bitte dieses Formular zusammen mit der Abschlussarbeit abgeben!

Erklärung zur Abschlussarbeit

gemäß § 25, Abs. 11 der Ordnung für den Bachelorstudiengang Informatik vom 06. Dezember 2010:

Hiermit erkläre ich Herr / ~~Frau~~

Robin Krause

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

Frankfurt am Main, den

Unterschrift der Studentin / des Studenten

Inhaltsverzeichnis

Abkürzungsverzeichnis

CNN ..... Convolutional Neural Network

ResNet ..... Residual Network

FLOPs ..... Floating point operations per secon

TSNE ..... t-distributedStochastic Neighbor Embedding

1 Einleitung

1.1 Motivation

Die Wissenschaft der Numismatik befasst sich mit den Zahlungsmitteln und seiner geschichtlichen, politischen sowie ökonomischen Geschichte. In diesem Zusammenhang stellt die Erforschung antiker Münzen einen relevanten Bereich dar, der beispielsweise der Erforschung der Geldgeschichte dient. Ein wichtiges Werkzeug bei der Analyse von antiken Münzen ist die Stempelanalyse. Hiermit kann ermittelt werden, wie viele Stempel zur Herstellung antiker Münzen benötigt wurden. Dieser Vorgang erfordert eine hohe Expertise und ist zeitaufwändig, insbesondere da die Auswertung und Zuordnung verschiedener Stempel und Münzen bislang überwiegend per Hand erfolgt.
Die Motivation dieser Arbeit ist daher, den Numismatiker bei der Stempelanalyse mit Hilfe von Machine Learning Algorithmen zu unterstützen und wenn möglich ein Framework zu bieten, welches die Stempelanalyse beschleunigen oder gar automatisieren kann.

1.2 Aufgabenstellung

Um die beschriebenen Schwierigkeiten bei der Stempelanalyse zu verbessern, soll diese mit Hilfe des Computers durchgeführt werden. Hierfür sollen die Münzen mittels eines Clusteringalgorithmus in Cluster unterteilt werden, welche die Stempel darstellen. Dabei wird nach dem Vorbild von Joris Guérin, Olivier Gibaru, Stéphane Thiery, und Eric Nyiri [GGTN18] gearbeitet. Guerin et al. setzen ein Convolutional Neural Network ein, welches die Merkmale der Münzen extrahiert. Diese extrahierten Merkmale werden wiederum durch Clusteringalgorithmen in Cluster eingeteilt.
Daraus ergibt sich ein zweite Teilaufgabe, das Training eines Convolutional Neural Networks, welches der Merkmalextrahierung dienen soll. Aufgrund des kleinen Datensatzes soll außerdem die Möglichkeit der Datenerweiterung durch Datenaugmentation getestet werden.

1.3 Struktur dieser Arbeit

Zur Erarbeitung der Aufgabenstellung sollen dafür zunächst in Kapitel 2 die Grundlagen erörtert werden, die für diese Arbeit erforderlich sind. Kapitel 3 soll einen kurzen Überblick über die Münzen liefern.
In den Kapiteln 4 und 5 soll eine Lösung der Problemstellung dieser Arbeit skizziert werden. Kapitel 4 befasst sich hierbei mit dem Training eines Convolutional Neural Networks auf einer kleineren Teilmenge an Stempeln, welches für die Merkmalextrahierung genutzt werden soll. Dabei wird auch die Effektivität der Datenaugmentierung für das Vergrößern des Trainingsdatensatzes geprüft.
Kapitel 5 befasst sich wiederum mit der Aufgabe der Clusterbildung und somit der Aufgabe der Zuteilung der Münzen zu verschiedenen Stempeln. Zunächst wird dabei die Effektivität eines vortrainierten CNN zur Merkmalextrahierung

evaluiert. Danach wird die ganze Methode analysiert und ausgewertet.
In Kapitel 6 erfolgt ein Vergleich der Methode dieser Arbeit mit bestehenden Methoden und in Kapitel 7 werden im Anschluss alternative Umsetzungsmöglichkeiten der hier angewandten Methode erörtert.

2 Grundlagen

2.1 Stempelanalyse

Die Stempelanalyse beschäftigt sich mit der Frage, wie viele Stempel für die Herstellung antiker Münzen gebraucht wurden, wie diese sich unterscheiden und in welcher Reihenfolge sie genutzt wurden. Daraus lässt sich viel über die Herkunft der Münze und den Geldumlauf in Erfahrung bringen[Coi, 3.2].
Die Münzen, die in dieser Arbeit betrachtet werden, wurden mittels Hammerprägung hergestellt. Bei der Hammerprägung wird ein Münzrohling zwischen einen Oberstempel und einen Unterstempel gelegt und durch mehrere Hammerschläge auf den Oberstempel geprägt[Kah05, S.181]. Dabei ist der Unterstempel fest fixiert, zum Beispiel in einem Holzblock, der Oberstempel wird mit der Hand festgehalten, ist also freistehend. Durch den freistehenden Oberstempel traten häufig Fehlprägungen auf, wenn etwa der Stempel den Münzrohling nicht vollständig traf[Kah05, S.506], oder es entstanden Doppelprägungen durch die wiederholten Schläge auf den Oberstempel. Dies erschwert einem Laien die Identifizierung verschiedener Stempel, wie Abbildung 1 zu entnehmen ist.

(a) Münze ohne offensichtliche Fehlprägung

(b) Münze welche nicht vollständig geprägt wurde.

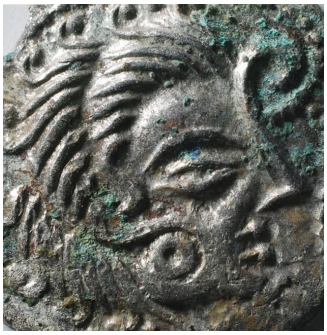
Abbildung 1: Keltische Münzen geprägt mit dem Stempel *O*. Die Fehlprägung von (b) erschwert die Zuordnung.
Bilder vom Jerseys Celtic Coin Hoard[Mue] wurden bereitgestellt durch Philip De Jersey.

Darüber hinaus nutzen sich diese Stempel mit der Zeit ab und müssen entweder nachgeschliffen oder sogar vollständig ersetzt werden. Die Suche und Zuordnung der besagten Stempel ist daher aufwendige Handarbeit. Dabei wird sowohl der Stempel der Vorderseite als auch der Stempel der Rückseite untersucht und im besten Fall ergibt sich eine chronologische Reihenfolge der Stempel[Coi, 3.2]. Eine durch ein Gespräch mit einem Numismatiker in Erfahrung gebrachte Methode zur Suche und zum Abgleich eines Stempels einer Münze lässt sich wie folgt beschreiben: Zunächst werden Bilder der Münzen angefertigt, welche alle in der selben Größe ausgedruckt werden. Danach werden auf einem Pauspapier

oder ähnlichem die wichtigsten Merkmale des Stempels der Münze abgezeichnet. Dieses Pauspapier wird nun auf die Bilder anderer Münzen gelegt, um die Merkmale miteinander zu vergleichen.
Dieser Vorgang erfordert nicht nur Expertise bezüglich der Münzen, sondern kostet dazu auch viel Zeit.

Abbildung 2: Beispiel keltischer Münzen von verschiedenen Stempeln. Dabei sind unten zwei Münzen des gleichen Stempels abgebildet. Der Unterschied von Stempel L zu den beiden anderen ist sehr deutlich. Bei den Stempeln A und B werden die Unterschiede jedoch bereits geringer.
Bilder vom Jerseys Celtic Coin Hoard[Mue] wurden bereitgestellt durch Philip De Jersey.

(a) Münze CATII-H-05775 des Stempel L



(b) Münze CATII-H-08745 des Stempels B.

(c) Münze CATII-H-09434 des Stempels A

(d) Münze CATII-H-09830 des Stempels A.

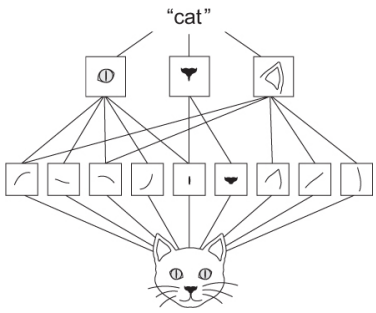
2.2 Convolutional Neural Networks

Die Inspiration für künstliche neuronale Netze entstammt aus der Natur, genauer dem Gehirn von Tieren. Die Weitergabe von Informationen in einem künstlichen neuronalen Netz spiegelt die Reizweiterleitung zwischen Neuronen in einem tierischen Gehirn wider. Diese Neuronen werden im künstlichen neuronalen Netz durch ein künstliches Neuron ersetzt, welches die Ausgabe des Neurons mit Hilfe einer Transferfunktion und einer Aktivierungsfunktion berechnet. In einem künstlichen neuronalen Netz sind diese Neuronen in Schichten zusammengefasst und die Informationen werden von einer zur nächsten Schicht weitergeleitet, wobei der Ausgang eines Neurons der vorherigen Schicht mit jedem Eingang des darauf folgenden verbunden ist.
Ein künstliches neuronales Netz besteht üblicherweise aus einer Eingabeschicht, mehreren versteckten Schichten sowie einer Ausgabeschicht[JP18, Kapitel 1].
Convolutional Neural Networks (CNN) sind spezielle künstliche neuronale Netze, welche für die Aufgabe der Objekterkennung und Bildklassifizierung die wohl meist genutzten Deep learning Modelle sind[Cho18, Kapitel 5].
Die typische CNN Architektur setzt sich aus folgenden drei Bestandteilen zusammen:

1. einer Eingabeschicht
2. ein oder mehreren Convolutional-Schichten
3. einer Klassifizierungsschicht

Die Convolutional-Schichten zeichnen sich durch die Faltungsoperation aus. Dabei werden Filter, welche nur kleine Teile des Bildes überdecken, über die Eingabe der Convolutional-Schicht gelegt und dann extrahieren diese mittels einer Faltungsoperation lokale Merkmale der Eingabe, welche in einer *feature map* gespeichert werden. Eine Convolutional-Schicht kann dabei mehrere Filter besitzen, welche unterschiedliche Merkmale aus der Eingabe herauslesen können. Die Faltungsoperationen sind translationsinvariant, das heißt sie erkennen gelernte Merkmale nicht nur beispielsweise in einem bestimmten Teilausschnitt des Bildes, sondern diese werden überall im Bild wiedererkannt. Dies stellt einen Vorteil gegenüber klassischen neuronalen Netzen dar, die Merkmale nur global erkennen können.
Durch mehrere aufeinander folgende Convolutional-Schichten können auch komplexere Strukturen in einem Bild erkannt werden. Erkennt beispielsweise die erste Schicht einfache Formen wie Kanten und Ecken, können die darauf folgenden Convolutional-Schichten Strukturen im Bild erkennen, welche wiederum sich aus den vorherigen zusammensetzten[Cho18, Kapitel 5.1.1]. Diese komplexeren Merkmale werden letztlich von einer oder mehreren verbundenen Schichten für die Klassifizierung ausgewertet.

Abbildung 3: Vereinfachtes Beispiel der möglichen Zusammensetzung solcher Strukturen. Bild aus [Cho18, Kapitel 5.1.1]

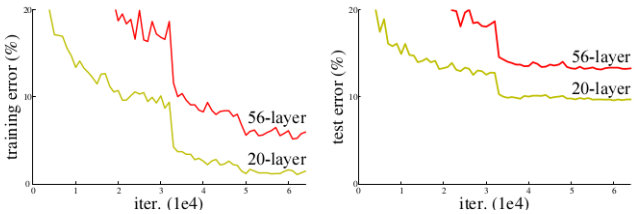


Für weitergehende mathematische Erklärungen sowie einen tieferen Einblick verweise ich auf die Masterarbeit von A.Loyal [Loy18, S.6-18], welche sich mit der Klassifizierung von römischen Münzen durch CNN befasst hat.

2.3 ResNet Architektur

Eine spezifische CNN Architektur sind die Deep Resiudal Networks, auch bekannt als ResNet. Sie zeichnen sich durch ihre hohe Anzahl an gestapelten Schichten aus. Die Architektur wurde 2015 von Kaiming He, Xiangyu Zhang, Shaoqinq Ren und Jia Sun vorgestellt und hat auch im selben Jahr viele Preise für die Aufgaben der ImageNet-Erkennung, ImageNet-Lokalisierung, COCO-Erkennung und COCO-Segmentierung gewonnen[HZRS15, S. 1]. He et al. beschäftigten sich mit der Frage, ob durch das Stapeln von immer mehr Schichten in einem einfachen Convolutional Neural Network die Performance verbessert werden kann. Wie in Abbildung 4 auf der rechten Seite zu sehen ist, ist der Testfehler für das CNN mit 56 Schichten im Vergleich zu dem mit nur 20 Schichten deutlich höher, die Annahme erweist sich somit als nicht richtig. He et al. vermuteten, dass durch die steigende Tiefe des Netzwerks und auch durch die größere Anzahl an Parametern das CNN zu *overfitten*[1] beginnt. Diese Vermutung lässt sich jedoch durch den höheren Trainingsfehler (links in Abbildung 4) widerlegen, da dieser beim *overfitting* geringer sein müsste.

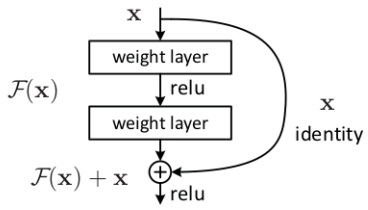
Abbildung 4: Trainings und Testfehlers von CNNs mit 20 und 56 Schichten. Niedrigere Werte sind besser. Bild aus [HZRS15]



He et al. kamen zu folgender Hypothese: *"Die Verschlechterung (der Trainingsgenauigkeit) deutet darauf hin, dass nicht alle Systeme ähnlich leicht zu optimieren sind. Betrachten wir die flachere Architektur und ihr tieferes Pendant, das weitere Schichten hinzufügt. Es gibt eine konstruktive Lösung für das tiefere Modell: die hinzugefügten Schichten sind Identitätsabbildung, und die anderen Schichten werden vom gelernten flacheren Modell kopiert. Die Existenz dieser konstruierten Lösung deutet darauf hin, dass ein tieferes Modell keinen höheren Trainingsfehler produzieren sollte als sein flacheres Gegenstück"*[HZRS15, S. 1]. Das Erlernen der Identitätsabbildungen durch das CNN erwies sich jedoch nicht so einfach wie angenommen. He et al. sind dieses Problem mit der Einführung des Residual Blocks angegangen.

[1] *Overfitting* tritt auf, wenn eine Überanpassung an den Trainingsdatensatz stattfindet. Dadurch wird keine gute Generalisierung der Daten gelernt, wodurch die Genauigkeit auf unbekannten Daten schlechter wird.

Abbildung 5: Residual learning Block. Bild aus [HZRS15]



Im Residual Block wird eine *shortcut connection* eingeführt, welche die Ausgabe der letzten Schicht direkt auf die Ausgabe der darauf folgenden Schicht addiert. Dadurch ergibt sich die residuale Abbildung $F(x) + x$ [HZRS15]. He et al. nehmen an, dass es einfacher ist, die residuale Abbildung $F(x) = H(x) - x$ zu optimieren als die originale Abbildung $H(x)$. Denn damit $F(x) + x$ die Identitätsabbildung abbildet, muss das neuronale Netz lediglich lernen, dass $F(x) = 0$ ist, anstatt das $H(x)$ die Identitätsabbildung lernt.
Folglich sollte, wie zuvor von He et al. angenommen, die Performance eines Modells durch das Hinzufügen eines Residual Blocks sich gegenüber seinem flacheren Gegenspieler nicht verschlechtern. Vielmehr ergaben die Tests von He et al., dass diese Residual Blocks sogar die Performance erhöhen können[HZRS15, S.6].
In Abbildung 6 ist ein ResNet mit 34 Schichten zu sehen und ein schlichtes CNN mit genau so vielen Schichten. Dabei besteht die ResNet Architektur aus mehreren aufeinander folgenden Convolutional Blöcken, welche aus mehreren aufeinander folgenden Residual Blöcken bestehen. Ein Residual Block umfasst zwei 3x3 Convolutional-Schichten, wobei die Eingabe x des Blocks ist über eine *shortcut connection* mit der Ausgabe $F(x)$ des Blocks verbunden. Wenn die Dimensionen von $F(x)$ und x gleich sind, kann x direkt auf $F(x)$ aufaddiert werden. Erhöhen sich die Dimensionen am Anfang eine Convolutional Blocks hingegen(in Abbildung 6 ist das der Fall bei den gestrichelten Linien), wird die Dimension von x beispielsweise durch *zero padding* erhöht, indem Nullwerte hinzugefügt werden, um die gleiche Dimension zu erreichen.
Des Weiteren besitzt die ResNet Architektur eine Poolingschicht am Ende des letzten Convolutional Blocks und lediglich eine verbundene Schicht zur Klassifizierung der ausgegebenen Merkmale der Convolutional Blöcke.

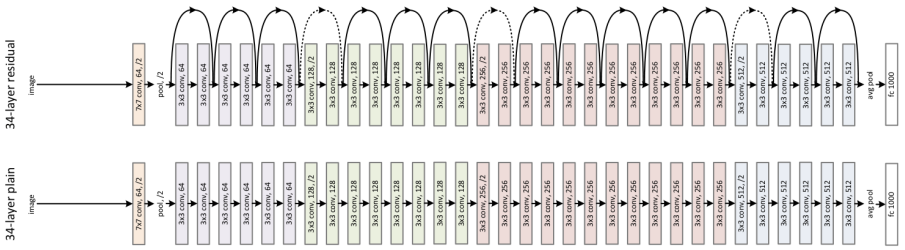


Abbildung 6: Einfaches CNN mit 34 Schichten und ResNet mit 34 Schichten. Bild aus [HZRS15]

Die ResNet Architektur hat eine Implementation für die Tiefen[2] 34 ,50,101 und 152 . Abbildung 7 enthält eine Übersicht über den Aufbau der Architekturen mit unterschiedlicher Tiefe.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Abbildung 7: Bild aus [HZRS15]

Der Aufbau der Residual Blocks unterscheidet sich für die tieferen Modelle. Diese sogenannten *deeper bootleneck architectures* bestehen aus drei anstatt zwei Convolutional-Schichten mit zwei 1x1 Schichten und einer 3x3 Schicht. Dabei verkleinert die erste 1x1 Schicht die Dimensionen, wodurch die 3x3 Schicht eine kleinere Eingabe- und Ausgabedimension hat, welches die Effizienz erhöht und zu weniger Rechenoperationen führt. Die letzte 1x1 Schicht stellt die Dimension der Eingabe wieder her oder erhöht diese.
Dadurch hat die ResNet Architektur für ihre Tiefe relativ wenige floating point operations per second (FLOPs) im Vergleich zu deutlich flacheren Architekturen. So hat etwa die VGG19 Architektur mit 19 Schichten 19,6 Milliarden Flops, die ResNet 152 Architektur mit 152 Schichten weist hingegen gerade mal 11,3 Milliarden Flops auf. Dies führt auch zu einem deutlich schnelleren Trainingsvorgang.

[2]Die Tiefe ist die Anzahl an Convolutional-Schichten plus eine verbundene Schicht

2.4 Gradient-weighted Class Activation Mapping

Ein Problem, welches bei der Nutzung eines CNN für die Klassifizierung von Bildern auftritt ist, dass die Entscheidung des Netzes, warum ein Bild X der Klasse Y zugeordnet wurde, dem Benutzer nicht wirklich klar wird.
Gradient-weighted Class Activation Mapping, kurz Grad CAM, ist eine Methode, die eine Visualisierung für den Entscheidungsprozess des CNN bietet. In voherigen Arbeiten mit Münzdaten von A. Loyal[Loy18, S.20] und S. Gampe [Gam19, S.7-9] hat sich die Visualisierungsmethode Grad-CAM als sehr hilfreich erwiesen. Bei Grad-Cam wird mittels der Gradienten der Ausgabe der letzten Convolutional-Schicht eines CNNs die Pixel im Bild hervorgehoben, welche für die Klassifizierung zu einer festgelegten Klasse wichtig sind. Diese Hervorhebung findet mittels sogenannter Heatmaps statt, ein Beispiel einer solchen Heatmap in Abbildung 8

Abbildung 8: Beispiel einer Heatmap der Münze CATII-H-15560 der Klasse A
Bilder vom Jerseys Celtic Coin Hoard[Mue] wurden bereitgestellt durch Philip De Jersey.

Diese Heatmaps können sehr hilfreich für die Evaluation eines trainierten CNN und der gelernten Merkmale für die spätere Merkmalextrahierung sein, da auch ein Numismatiker so einen Überblick erhalten kann, ob die ausgewählten Merkmale auch mit den bestimmten Merkmalen eines Stempels übereinstimmen.
Für einen genaueren mathematischen Hintergrund für die Berechnung der Heatmaps wird auf [SCD+19, S.4-6] verwiesen. Die Implementierung von Grad-Cam, die in dieser Arbeit benutzt wurde, entstammt aus Keras[3].

[3]https://github.com/keras-team/keras-io/blob/master/examples/vision/grad_cam.py

2.5 Welchs t-Test

Der Welchs t-Test ist ein Hypothesentest für die Signifikanz zwischen Mittelwertunterschieden zwischen zwei voneinander unabhängigen Stichproben. Für die Entscheidung, ob die Hypothese abgelehnt werden kann oder nicht, wird ein p-Wert errechnet. Dabei ist der p-wert wie folgt definiert: *Unter der Annahme der Gültigkeit der Nullhypothese entspricht der p-Wert der Wahrscheinlichkeit der beobachteten oder einer extremeren Prüfgröße,welche mit der in der Alternativhypothese spezifizierten Richtung des Effekts übereinstimmt*[JB07, S.107]. Unter einem vorher festgelegten Signifikanzniveau α wird die Nullhypothese abgelehnt, wenn der p-wert kleiner als α ist. Für die Berechnung der p-Werte wird die Funktion des Python Paket SciPy namens *ttest_ind*[T-t] benutzt.

2.6 K-means Clustering

2.6.1 Beschreibung

K-means Clustering ist eine unsupervised Machine Learning Methode, bei der der Benutzer das Modell nicht überwachen muss. Da eine vorangegangene Klassifizierung für die Anwendung nicht nötig ist, ist diese Methode sehr vorteilhaft wenn mit Daten gearbeitet wird, welche noch nicht oder nur zum Teil klassifiziert sind.
K-means Clustering dient der Identifizierung und Einteilung von Datenpunkten des mehrdimensionalen Raumes in eine bestimmte Anzahl von Cluster.
Angenommen es gibt eine Datenmenge X mit N Objekten eines $D-$dimensionalen Euklidischen Raumes, kurz $X = \{x_1, .., x_n\}$, wobei $x_t = (x_{t_1}, x_{t_2}, .., x_{t_D})\ \forall t \in N$. Gesucht ist nun eine Unterteilung dieser N Punkte in K Cluster, wobei die einzelnen Cluster durch ihre Clusterzentren μ_k beschrieben werden. Ziel ist es, dass der Abstand der Datenpunkte in jedem Cluster zu dem dazugehörigen Clusterzentrum minimal ist. Um dies zu erreichen, soll folgende Funktion minimiert werden:

$$J = \sum_{l=1}^{N} \sum_{m=1}^{K} r_{lm} \parallel x_l - \mu_m \parallel^2 \text{ [Bis06]}$$

wobei $r_{lm} = 1$, wenn der Punkt x_l dem Cluster m zugeordnet ist, sonst $r_{lm} = 0$ ist und $\parallel x_l - \mu_m \parallel^2$ der Euklidische Abstand zwischen einem Punkt und dem dazugehörigen Clusterzentrum ist. Um J zu minimieren wird Llyods Algorithmus verwendet. Zuerst werden K initiale Clusterzentren gewählt und danach wird jeder Datenpunkt dem Clusterzentrum mit dem geringsten Abstand zugeordnet. Formal:

$$C_k = \{x_i : \parallel x_i - \mu_k \parallel^2 < \parallel x_i - \mu_j \parallel^2 \ \forall j \in \{\mu_1, .., \mu_K\} \setminus \mu_k\}$$

Danach wird in jedem Cluster das Zentrum neu gesetzt und zwar zum Mittelpunkt aller Punkte, welche in diesem Cluster liegen. Formal:

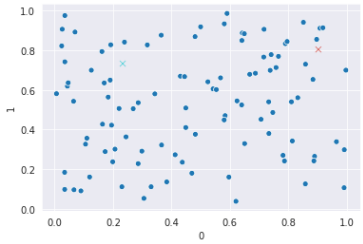
$$\mu_k = \frac{1}{|C_k|} \cdot \sum_{x_i \in C_k} x_i$$

Diese zwei Vorgänge werden solange wiederholt, bis sich keine neue Zuordnung der Punkte finden lässt[Bis06, S.423-427].

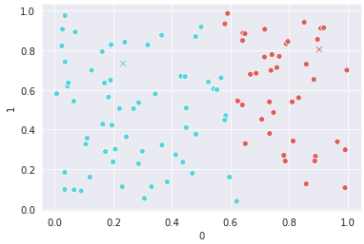
2.6.2 Beispiel

Abbildung 9: Einfaches Beispiel des K-means Clustering.

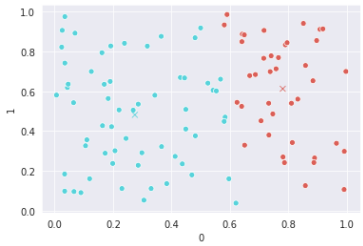
(a) Türkises und rotes Kreuz markieren die Clusterzentren. Datenpunkte noch nicht zugeordnet.



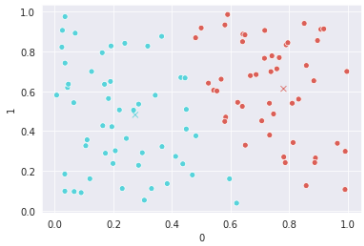
(b) Zuordnung der Punkte zum Clusterzentrum mit der geringsten Euklidischen Distanz



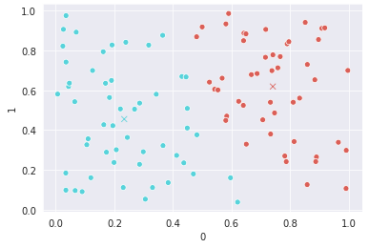
(c) Clusterzentren werden neu berechnet



(d) wie (b)



(e) Keine Veränderung der Clusterzentren. Finale Clusterzentren gefunden



2.7 Hierarchisches Clustering

Bei der hierarchischen Clustering-Methode gibt es zwei Ansätze, die Cluster zu bilden, die top-down Methode und die bottom-up Methode[MR10, 278]. Bei der top-down Methode befinden sich alle Datenpunkte zunächst in einem Cluster und werden dann sukzessive in kleine Untercluster aufgeteilt, bis die gewünschte Clustergröße erreicht wird. Die bottom-up Methode geht genau gegenteilig vor, da alle Datenpunkte zunächst eigene Cluster repräsentieren und dann schrittweise zu größeren Clustern vereinigt werden. In dieser Arbeit wird die bottom-up Methode verwendet, das sogenannte *Agglomerative Hierachical Clustering*.
Hierbei werden nach einer gegeben Abstandsmetrik werden zunächst alle Distanzen zwischen den einzelnen Datenpunkten berechnet. Danach werden jene Datenpunkte nach einem vordefinierten Ähnlichkeitsmaß zu einem Cluster vereint und alle Abstände neu berechnet. Dieser Vorgang wird solange wiederholt, bis alle Punkte in einem Cluster vereint sind.
Dabei gibt es folgende Gruppen der Ähnlichkeitsmaße:[MR10, S.279]

1. Single-link clustering: Bei der single-linkage Methode wird das Ähnlichkeitsmaß zwischen zwei Clustern definiert als der geringste Abstand zwischen jedem Datenpunkt des einen Clusters zu jedem Datenpunkt des anderen Clusters.

2. Complete-link clustering: Ähnlich zu single-linkage, nur wird hier der größte Abstand verwendet.

3. Average-link clustering: Bei average-linkage wird der Abstand als Durchschnittsdistanz zwischen jedem Datenpunkt des einen Clusters zu jedem Datenpunkt des anderen Clusters berechnet.

Das Ähnlichkeitsmaß *Ward*, welches in der Arbeit verwendet wird, gehört zur Kategorie des average-link Clustering. Dabei wird das Ähnlichkeitsmaß *Ward* wie folgend berechnet: Sei u ein neu entstandenes Cluster, welches aus s und t besteht, dann ergibt sich das Ähnlichkeitsmaß zu jedem anderen Cluster v aus

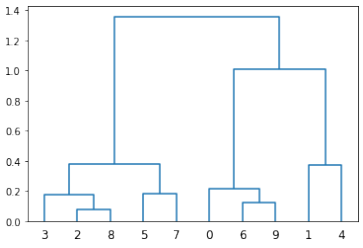
$$d(u,v) = \sqrt{\frac{|v|+|s|}{|v|+|s|+|t|} \cdot d(v,s)^2 + \frac{|v|+|t|}{|v|+|s|+|t|} \cdot d(v,t)^2 + \frac{|v|}{|v|+|s|+|t|} \cdot d(s,t)^2}$$

[Sci] Die Ergebnisse eines hierarchischen Clusters werden mittels eines Dendrogrammes (siehe Abbildung 10a) dargestellt. Dabei stellt die Wurzel das Cluster dar, welches alle Datenpunkte enthält. Ein innerer Knoten im Dendrogramm repräsentiert ein Cluster, welches aus der Vereinigung aller seiner Kindknoten besteht.

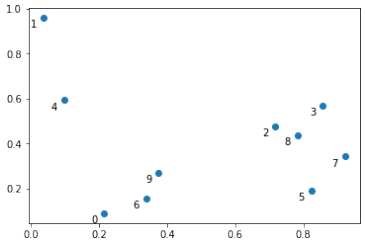
2.7.1 Beispiel

Abbildung 10: Einfaches Beispiel des Agglomerative Clustering mit Ward.

(a) Dendrogram



(b) Datenpunkte



2.8 Cluster Evaluationsverfahren

Die erzeugten Cluster sind als Laie schwer auszuwerten, da die Unterschiede der Stempel teilweise minimal sind, sodass die Zuordnung zu einem Stempel somit selbst für Numismatiker kein trivialer Vorgang ist. Daher wird in dieser Arbeit eine Metrik benutzt, welche die Auswertung eines Clusters anhand einer gegebenen Menge an klassifizierten Daten berechnen kann. Da nicht alle Bilder klassifiziert sind, kann auf diese Weise nur die Metrik über eine Teilmenge der Bilder im Cluster errechnet werden.
Für diese Arbeit wurde der V-Measure Score, welcher sich aus dem Homogeneity Score und Completeness Score zusammensetzt, gewählt. Die Implementierungen der Scores entstammen dem Python Paket sckit-learn Version 0.23.2[Clu].

2.8.1 V-Measure Score

Der 2007 von A. Rosenberg und J. Hirschberg vorgestellte V-Measure Score [RH07] beschreibt ein Maß, welches sich aus den Kriterien der Homogenität und der Vollständigkeit zusammensetzt und angibt, inwiefern diese erfüllt wurden.
Um das Homogenitätskriterium zu erfüllen, dürfen Datenpunkte in einem Cluster auch nur aus einer Klasse stammen. Das Vollständigkeitskriterium verläuft symmetrisch zu dem Homogenitätskriterium und wird erfüllt, wenn alle Datenpunkte einer Klasse in einem Cluster vorkommen.
Die beiden Kriterien weisen jedoch auch Schwächen auf. Würde man etwa alle Datenpunkte in ein einziges Cluster aufnehmen, so hätte man das Vollständigkeitskriterium gänzlich erfüllt, da alle Datenpunkte jeder Klasse immer in einem Cluster vorkommen. Allerdings wäre das Homogenitätskriterium verletzt, da in dem Cluster Daten aus allen Klassen vorkommen.
Unter der Annahme, dass n die Anzahl der Datenpunkte ist, würde umgekehrt durch die Zuweisung jedes Datenpunktes in ein eigenes Cluster, also ist die Anzahl an Cluster n, das Homogenitätskriterium erfüllt werden, nicht jedoch das

Vollständigkeitskriterium, da die Datenpunkte der Klassen in einzelnen Clustern vorkommen.
Ein Erhöhen der Homogenität führt folglich häufig zu einem Verringern der Vollständigkeit [RH07, S.411].
Daher wird der V-Measure Score als "*harmonischer Mittelwert der verschiedenen Homogenitäts- und Vollständigkeitswerte* "bezeichnet[RH07, S.410].
Die Berechnung erfolgt nach folgender Formel:

$$v_\beta = \frac{(1+\beta) \cdot Homogenität \cdot Vollständigkeit}{\beta \cdot Homogenität \cdot Vollständigkeit}$$

wobei $\beta > 1$ der Vollständigkeit ein größeres Gewicht geben, und $\beta < 1$ wiederum der Homogenität. In dieser Arbeit wird zur Berechnung von V-Measure β immer auf 1 gesetzt.
Dabei ist Wertebereich für den V-Measure zwischen 0 und 1, wobei 1 für den perfekten Wert steht und 0 für den Schlechtesten. Dadurch lässt sich ein Clustering relativ einfach bewerten und mit anderen vergleichen.
Ein großer Nachteil des V-Measure ist hingegen, dass eine Grundwahrheit gegeben sein muss, da sonst eine Berechnung nicht möglich ist.
Für einen genaueren mathematischen Hintergrund der Berechnung der Homogenität sowie der Vollständigkeit wird auf [RH07] verwiesen.

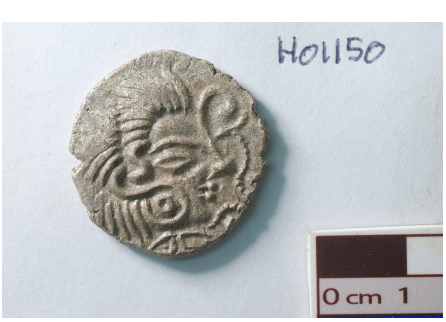
3 Daten

3.1 Beschreibung der Daten

Die keltischen Münzen, die in dieser Arbeit betrachtet werden, entstammen einem großen Münzschatz, welcher 2012 auf der Insel Jersey gefunden wurde[Mue]. Die Insel Jersey liegt nördlich der Normandie und ist als Kronbesitz direkt der Britischen Krone unterstellt. In dem Münzschatz wurden 70.000 keltische Münzen gefunden, welche hauptsächlich dem keltischen Volksstamm der Curiosoliten zuzuordnen sind, sowie Münzen von einigen anderen Stämmen und einige andere Schätze, wie zum Beispiel goldene Halsringe.
Der Stamm der Curiosoliten lebte auf der bretonischen Halbinsel im Westen des heutigen Frankreichs. Die Münzen fanden möglicherweise ihren Weg zur Insel Jersey, als vor ca. 2000 Jahren Römische Legionen Richtung dem heutigen Nordfrankreich zogen und die dort ansässigen keltischen Stämme ihren Reichtum vor den einfallenden Legionen beschützen wollten.
Die Münzdaten dieser Arbeit entstammen jenem Stamm der Curiosoliten und belaufen sich auf insgesamt 2658 Stück wobei für jede Münze ein Bild für die Vorderseite sowie ein Bild für die Rückseite gemacht wurde. Auf der Vorderseite der Münzen ist ein Kopf abgebildet, welcher nach rechts schaut, und auf der Rückseite ein Tier, welches einem Pferd ähnelt. Andere Arbeiten, die ebenfalls in die Vorbereitungen auf diese Arbeit einbezogen wurden[Loy18], haben zumeist Münzen verwendet, auf denen ein Portrait abgebildet ist, weshalb der Fokus auch in dieser Arbeit auf die Vorderseite der Münzen gelegt wurde, da hierdurch ein besseres Ergebnis erwartet wird. Die Methodiken dieser Arbeit lassen sich auch auf die Rückseite der Münzen anwenden.

3.2 Vorverarbeitung der Daten

Da auf den Bildern nicht nur die Münze, sondern auch ein Maßstab sowie eine Beschriftung mit dem Namen der Münze zu sehen sind, wurde ein Tool verwendet, welches dem Verfasser dieser Arbeit zur Verfügung gestellt wurde. Dieses erkennt den Kopf auf der Münze und schneidet das Bild so zu, sodass im besten Falle die Münze selbst zu sehen ist. Ein Vergleich vor und nach der Verarbeitung ist in Abbildung 11 zu sehen.



(a) Unverändertes Bild

(b) Durch das Tool verarbeitetes Bild

Abbildung 11: Münze H-01150 vor und nach der Verarbeitung
Bilder vom Jerseys Celtic Coin Hoard[Mue] wurden bereitgestellt durch Philip De Jersey.

Nach der Vorverarbeitung waren 1217 Bilder im Datensatz vorhanden. Dass dies nicht der Anzahl der ursprünglich vorhandenen Bilder entspricht, ist wohl darauf zurück zu führen, dass das verwendete Tool keinen Kopf erkannt und das Bild dann dementsprechend auch nicht zugeschnitten hat. Zu berücksichtigen ist, dass das Zuschneiden der Bilder durchaus die Gefahr bergen kann, dass für eine korrekte Klassifizierung relevante Teile des Bildes abgeschnitten werden. Eine kleinere Stichprobe hat jedoch gezeigt, dass die Zuschnitte relativ passend erfolgten. Aufgrund der gewonnenen Zeitersparnis und der zu erwartenden geringen Fehlerquote wurden etwaige Fehlzuschnitte daher in Kauf genommen. Eine weitere Vorverarbeitung der Bilder war nicht nötig.

4 Convolutional Neural Network

Dieser Teil der Arbeit beschäftigt sich mit dem Training eines Convolutional Neural Network sowie der Datenaugmentation. Dabei soll erörtert werden, welchen Einfluss verschiedene augmentierte Daten auf das Training des CNN und seiner Performance haben. Der Aufbau der Experimente wird in Kapitel 4.1 beschrieben und in Kapitel 4.2 die Ergebnisse evaluiert.
Als Programmiersprache wurde Python gewählt, da für diese mehrere Machine Learning Frameworks zur Verfügung stehen. Speziell für diese Arbeit wird das Framework Keras benutzt, welches in Tensorflow 2.3 inbegriffen ist.

4.1 Aufbau und Durchführung

4.1.1 Data Augmentation

Aufgrund der relativ kleinen Menge an klassifizierten Münzen wurde in einem ersten Schritt der Datensatz mit Hilfe von Datenaugmentierung vergrößert. Bei der Datenaugmentierung von Bildern werden diese beispielsweise durch Drehung oder Spiegelung an einer Bildachse verändert.
Für die Generierung der Bilder wurde die Tensorflow Klasse ImageDataGenerator genutzt[Imab]. Der ImageDataGenerator generiert während des Trainings eines CNN eine vorbestimmte Menge an augmentierten Bildern, welche aus einem gegebenen Verzeichnis erzeugt werden, sodass das CNN während des Trainings kein Bild doppelt sieht. Dies kann dem Problem des *overfitting* beim Training entgegenwirken [PW17].
Da in den einzelnen Klassen die Anzahl an Bildern so unterschiedlich ist, schien es sinnvoll, die Bilder nicht während des Trainings zu generieren. Der ImageDataGenerator bietet nämlich noch die Möglichkeit, die generierten Bilder abzuspeichern, wodurch man die Anzahl an generierten Bildern pro Klasse besser steuern kann.
Der ImageDataGenerator bietet eine Vielzahl an Augmentationsmöglichkeiten. Da nicht jede dieser Möglichkeiten für den Datensatz der keltischen Münzen passend gewesen wäre, ist diese Arbeit auf folgende sechs Möglichkeiten beschränkt:

1. Rotations Range: Reichweite für eine Rotation des Bildes. Das Bild wird zufällig zwischen $[0, X]$ rotiert. Abbildung 12 zeigt eine Rotation links mit dem Wert $X = 45$ und eine mit $X = 180$. Da bei höheren Werten die Münzen auf dem Kopf stehen, wurden in dieser Arbeit für die Experimente nur die Werte 45 und 90 betrachtet.

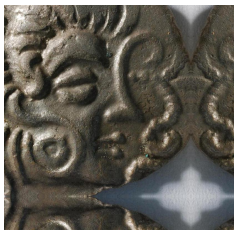
(a) Rotation mit Reichweite 45

(b) Rotation mit Reichweite 180

Abbildung 12: Beispiele für die Rotation

2. Höhen- und Breitenverschiebung, welche ein Bild in Reichweite $[-X, X]$ verschiebt. Wie im rechten Bild der Abbildung 13 zu erkennen, wird dort ein Teil des Bildes abgeschnitten, der für die Klassifizierung wichtig sein könnte. Um dies zu verhindern, wird als maximaler Wert 0.1 gewählt.

(a) Höhen- und Breitenverschiebung mit Reichweite $[-0.1, 0.1]$

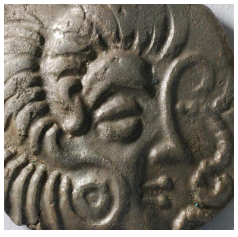


(b) Höhen- und Breitenverschiebung mit Reichweite $[-0.2, 0.2]$

Abbildung 13: Beispiele Höhen- und Breitenverschiebung

3. Zoom Range: zoomt auf die Reichweite $[1 - X, 1 + X]$ heran oder heraus. Auch hier wird als maximaler Wert 0.1 gewählt, damit nicht zu viel abgeschnitten wird, wie es etwa in Abbildung 14b der Fall ist.

(a) Zufälliger Zoom mit Reichweite $[-0.9, 1, 1]$



(b) Zufälliger Zoom mit Reichweite $[-0.8, 1.2]$

Abbildung 14: Beispiele Zoom

4. Fill mode: Gibt an, wie die Pixel außerhalb der Grenzen des Bildes, welche beispielsweise durch die Rotation des Bildes entstanden sind, aufgefüllt werden sollen. In Abbildung 15 sind die vier verschiedenen Möglichkeiten abgebildet.
   (a) *Constant* füllt die Pixel mit einer konstanten Zahl *cval*.
   (b) *Nearest* füllt die freien Pixel bis zum Rand des Bildes mit dem angrenzenden Wert auf.
   (c) *Reflect* reflektiert das Bild an der Bildgrenze.
   (d) *Wrap* legt das Bild an jeder Bildgrenze an.

(a) Fill mode *constant* mit $cval = 0$

(b) Fill mode *nearest*

(c) Fill mode *reflect*

(d) Fill mode *wrap*

Abbildung 15: Beispiele der fill modes anhand eines augmentierten Bildes

5. Horizontal Flip: Zufällige Spiegelung an der horizontalen Achse des Bildes. Ein Beispiel ist zu sehen in Abbildung 15a
6. Brightness range: Ändert die Helligkeit eines Bildes in einer vorgegebenen Reichweite. Für die Versuche wurde die Reichweite zwischen $[0.5, 1.5]$ gesetzt, da Voruntersuchungen gezeigt haben, dass größere und kleinere Werte das Bild zu stark erhellen oder verdunkeln können, wie in Abbildung 16b zu sehen ist.

(a) Helligkeitsänderung mit Reichweite $[0.5, 1.5]$

(b) Helligkeitsänderung mit Reichweite $[0, 2]$.

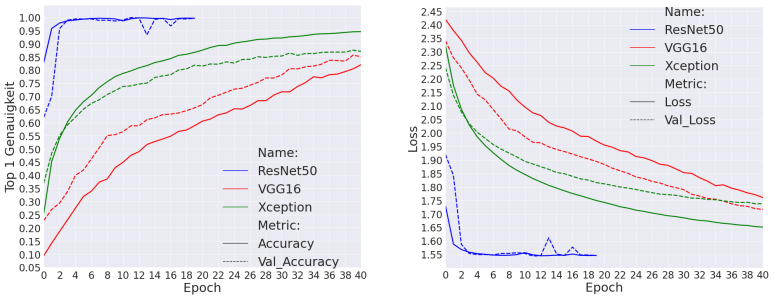
Abbildung 16: Beispiele für die Helligskeitsveränderung

4.1.2 Wahl der Stempel für das Training

Für die Seite 'A' der keltischen Münze gibt es laut der csv Datei, welche die Klassifizierung der Münzen beinhaltet, 54 verschiedene Stempel. Allerdings ist die Anzahl an gefundenen Münzen pro Stempel sehr unterschiedlich. Es gibt Stempel, wie den Stempel K mit 46 Bildern, aber auch Stempel mit nur einem einzigen Bild, wie Stempel U. In Tabelle 3 ist eine detaillierte Übersicht für die Anzahl an Bildern pro Stempel zu sehen.
Für das Training wurden die Klassen $A, B, D, F, I, K, L, O, R, S, W$ ausgewählt, da zu jedem dieser Stempel wenigstens 19 Bilder existieren. Dabei wurden die Stempel, welche mit einem Fragezeichen versehen waren, nicht in die Auswahl einbezogen, da vermutet wird, dass diese keinen eigenen Stempel darstellen. Es wird angenommen, dass die Zuordnung jener Münzen nicht eindeutig feststehen. Der Stempel G wurde versehentlich nicht aufgenommen, obwohl er für das Training geeignet gewesen wäre. Aufgrund des hohen Zeitaufwands wurde das Experiment jedoch nicht nochmals mit dem Stempel G wiederholt.

4.1.3 Wahl der CNN Architektur

Für die Experimente wurden die Xception und ResNet Architektur in die engere Auswahl gezogen, da diese bei den Experimenten von Guerin et al.[GGTN18, S.5] am Besten abgeschnitten haben, sowie die VGG16 Architektur, mit der A. Loyal bei der Klassifizierung von Münzen bereits gute Erfahrungen gemacht hat[Loy18]. Eine erste Voruntersuchung hat gezeigt, dass die ResNet Struktur deutlich schneller eine hohe Top 1 Genauigkeit[4] erreicht als die beiden anderen, wie in Abbildungen18a zu sehen ist. Dabei wurden für diese Voruntersuchungen vortrainierte CNN genutzt, welche in Keras enthalten sind.



(a) Trainings- und Validations- Top 1 Genauigkeit. Höhere Werte sind besser.

(b) Trainings- und Validations-Loss. Niedrige Werte sind besser.

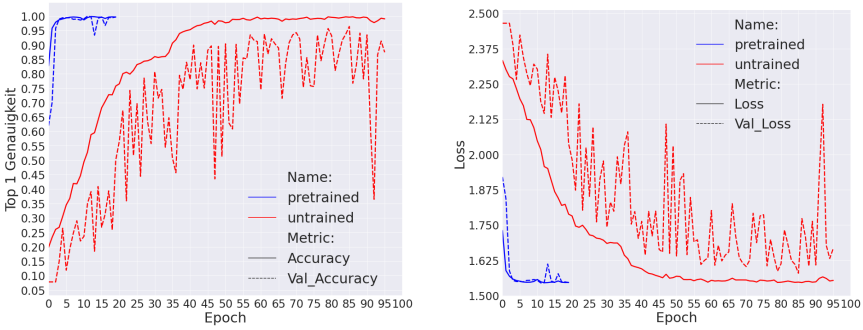
Abbildung 17: Vergleich des Trainings mit der VGG16, Xception und der ResNet Architektur. Alle drei mit vortrainierten Gewichten auf dem ImageNet Datensatz. ResNet und Xception Optimizer Adam mit Lernrate 0,001. VGG16 Optimizer SGD mit Lernrate 0.001. Daten augmentiert wie in Experiment 1 in Tabelle 2

[4]Die Top 1 Genauigkeit errechnet sich aus der Anzahl der richtig klassifizierten Bildern durch die Anzahl aller Bilder

Aufgrund der begrenzten Zeit für diese Arbeit fiel die Wahl letztlich auf die ResNet Architektur. Tensorflow stellt mehrere verschiedene Versionen von ResNet mit unterschiedlicher Tiefe zur Verfügung. Hier wurde die ResNet Version 2 [HZRS16] mit 50 Schichten gewählt, da durch die geringere Anzahl an FLOPS (Vergleich Abbildung 7) die Trainingszeit geringer ausfällt als bei den tieferen Varianten.

4.1.4 Aufbau des Trainings des CNN

Für das Training bestand die Möglichkeit, ein CNN von Grund auf zu trainieren oder ein CNN mit vortrainierten Gewichten zu nehmen. Die Wahl eines vortrainierten CNN ist bei einem kleinen Datensatz eine effektive und häufig gewählte Methode[Cho18, Kapitel 5.3]. Daher wurden beide Methoden in Betracht gezogen und getestet. Wie in Abbildung 18 zu sehen ist, ist das Training mit einem vortrainierten CNN deutlich schneller. Eine Trainings- Top 1- Genauigkeit von größer als $0,95$ wird schon nach zwei Epochen erreicht im Gegensatz zu dem nicht vortrainierten CNN, welcher für dieselbe Genauigkeit ungefähr 40 Epochen benötigt.



(a) Trainings- und Validations- Top 1- Genauigkeit. Höhere Werte sind besser.

(b) Trainings- und Validations-Loss. Niedrige Werte sind besser.

Abbildung 18: Vergleich des Trainings auf den Datensatz der keltischer Münzen eines CNN mit vortrainierten Gewichten und einem CNN mit zufälligen Startgewichten. Der Optimizer ist Adam mit einer Lernrate von 0,001. Daten augmentiert wie in Experiment 1 in Tabelle 2.

Außerdem sieht man, dass die Validationsgenauigkeit und der Validations-Loss beim vortrainierten CNN deutlich stabiler bleibt. Für diese Arbeit war daher, auch wegen des Zeitvorteils, ein vortrainiertes CNN sinnvoller. Das vortrainierte CNN wurde auf dem ImageNet[Imaa] Datensatz trainiert. Der Datensatz enthält 14 Millionen Daten und das CNN wurde auf 1000 Klassen trainiert.

Es gibt zwei Techniken, die man bei einem vortrainierten CNN anwenden kann, nämlich Feature Extraction und Fine-Tuning[Cho18, Kapitel 5.3.1]. Beim Fea-

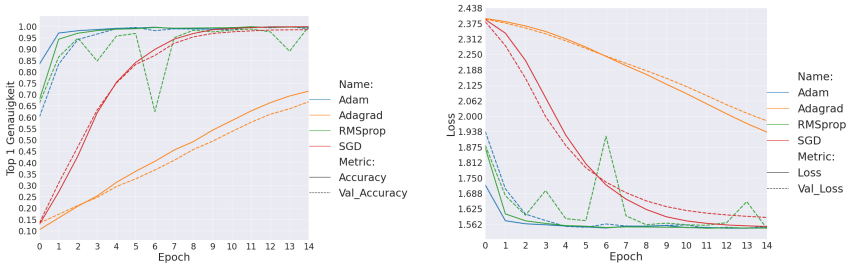
ture Extraction macht man sich die vorgelernten Merkmale zu Nutze. Man entfernt den trainierten Klassifizierer, welcher aus vollverbundenen Schichten und einer Ausgabeschicht besteht, von der letzten Convolutional-Schicht und fügt einen nicht trainierten Klassifizierer hinzu. Für das Training werden die Convolutional-Schichten "eingefroren", so dass diese nicht durch das Training verändert werden können. Dieser Ansatz ist für diese Arbeit jedoch wenig zielführend, da für das Clustering später nur die Merkmale benutzt werden sollen, welche die letzte Convolutional-Schicht ausgibt. Da bei einem Training wiederum nur die Klassifizierungsschicht verändert wird, nicht jedoch die Convolutional-Schichten, wäre ein Training obsolet.
Die zweite Technik ist das sogenannte Fine-Tuning. Beim Fine-Tuning wird zuerst ein CNN wie bei der Feature Extraction erstellt, d.h. es wird der trainierte Klassifizierer durch einen nicht trainierten ersetzt. Im Gegensatz zur Feature Extraction werden nicht alle Convolutional-Schichten eingefroren, sodass beim Training neue Merkmale gelernt werden können. Dieser Ansatz bietet sich für diese Arbeit an, da so eine bessere Repräsentation der Münzen gelernt werden kann. Betrachtet man die Heatmaps, welche mit einer Grad-Cam erstellt wurden, sieht man, dass beim Fine-Tuning auf mehr Details eingegangen wird im Gegensatz zu der Feature Extraction Methode. Wie in Abbildung 19a zu sehen ist, liegt hier der Fokus auch auf dem Rand, welcher in Abbildung 19b hingegen nicht aufgezeigt wird. Bei diesem Bild wäre der Rand für die Zuordnung des Stempels nicht sinnvoll, da der Bruch in der Münze kein Merkmal für einen Stempel darstellt.

(a) CNN mit Feature Extraction (b) CNN mit Fine-Tuning

Abbildung 19: Vergleich der Heatmaps der Münze CATII-H-15560 der Klasse A von einem CNN mit Feature Extraction und Fine-Tuning

Für das Training des ResNet 50 wurden die ersten vier Convolutional Blöcke (siehe Abbildung 7) eingefroren und der letzte für das Training frei gelassen. Zudem wurde eine Ausgabeschicht mit der Ausgabegröße 11 (Anzahl an Stempeln) hinzugefügt.

Mit den augmentierten Daten aus Experiment 1 wurden vier verschiedene Optimierungsalgorithmen getestet, um herauszufinden, welcher sich für das Training mit den keltischen Münzen am besten eignet. Wie aus Abbildung 20 ersichtlich wird, erreicht man entweder mit RMSprop oder Adam innerhalb der ersten vier Epochen eine Top 1 Genauigkeit von über 95%. Sowohl die Top 1 Genauigkeit als auch der Loss auf dem Validationsdatensatz scheinen jedoch stabiler mit Adam zu sein. Es kommen keine Sprünge vor wie bei RMSprop.
Als Optimizer wurde daher Adam genutzt.



(a) Trainings- und Validations- Top 1- Genauigkeit.
Höhere Werte sind besser.

(b) Trainings- und Validations- Loss.
Niedrige Werte sind besser

Abbildung 20: Vergleich des Trainings auf den keltischen Münzen mit verschiedenen Optimierungsalgorithmen. Daten wurden augmentiert wie in Experiment 1 in Tabelle 2 . Für alle Optimierungsalgorithmen wurde die Default Lernrate von Keras gewählt.

Voruntersuchungen haben außerdem gezeigt, dass sich für das Training eine Lernrate von 0.001 für fünf Epochen und dann eine Reduzierung der Lernrate auf 0.00001 für weitere 100 Epochen am besten eignet, da durch diese Reduzierung die Top 1 Genauigkeit um einige Prozentpunkte gesteigert werden konnte. Damit die CNNs beim Training nicht *overfitten*, wird *EarlyStopping*[TfE] genutzt, welches das Training stoppt, wenn eine ausgewählte Metrik sich nicht mehr um einen festgelegten Wert ändert. Als Metrik wurde der Validation Loss gewählt. Wenn sich dieser für 10 Epochen nicht um einen Wert von 0.001 ändert, wird der Trainingsvorgang abgebrochen.

4.1.5 Aufbau der Experimente

Um die verschiedenen Augmentationsmethoden miteinander zu vergleichen, wurden insgesamt zwölf Experimente durchgeführt. Dabei wurden zuerst die Daten mit den Parametern augmentiert, welche der Tabelle 2 zu entnehmen sind. Es wurde ein fester Seed für die Augmentation gesetzt, sodass man die Experimente replizieren kann. Für die Ermittlung eines späteren durchschnittlichen Top 1 Fehlers[5] wurden pro Experiment zwanzig CNN trainiert und die Top 1

[5]Der Top 1 Fehler errechnet sich folgendermaßen: Anzahl der falsch zugeordneten Münzen geteilt durch die Anzahl aller Münzen. Weitere Möglichkeit 1 - (Top 1 Genauigkeit)

Genauigkeit auf dem Testdatensatz jedes CNN in einer csv-Datei gespeichert, woraus sich der Top 1 Fehler einfach berechnen lässt. Des weiteren wurde dasjenige CNN abgespeichert, welches von den zwanzig Durchläufen die beste Top 1 Genauigkeit besitzt, da dieses für das spätere Clustering benötigt wird. Da ein CNN vergleichsweise viel Speicherplatz benötigt und das Testen 240 verschiedener CNN für das Clustering zeitaufwendig ist, wurde jeweils nur ein CNN pro Experiment abgespeichert. Es wurde letztlich für das CNN mit der besten Top 1 Genauigkeit abgespeichert, da davon auszugehen ist, dass sich dieses am besten für das Clustering eignet. Das Training eines CNN dauert im Schnitt ca. 12 Minuten[6], somit ergab sich eine Trainingszeit von 4 Stunden pro Experiment.

4.2 Ergebnisse des Trainings

In folgendem Abschnitt soll erarbeitet werden, ob die Datenaugmentation einen Vorteil für das Training des CNN erbracht hat. Insgesamt wurden zwölf Experimente durchgeführt mit verschieden augmentierten Bildern. Tabelle 2 zeigt die Parameter auf, die für die Datenaugmentierung benutzt wurden. Dabei ist das Ausgangsexperiment die Nummer 1, welches im Folgenden mit den übrigen Experimenten verglichen wird. In der Tabelle 2 werden daher die Unterschiede zu Experiment 1 rot markiert dargestellt.
Abbildung 21 zeigt den durchschnittlichen Top 1 Fehler von Experiment 1 im Vergleich mit dem Durchschnittswert von Experiment 0. Bei Experiment 0 wurde das CNN nur mit den ursprünglichen Daten trainiert. Es wird deutlich, dass der durchschnittliche Top 1 Fehler bei Experiment 1 niedriger ist. Daraus lässt sich folgern, dass das Augmentieren der Daten eine bessere Performance des CNN ermöglicht.

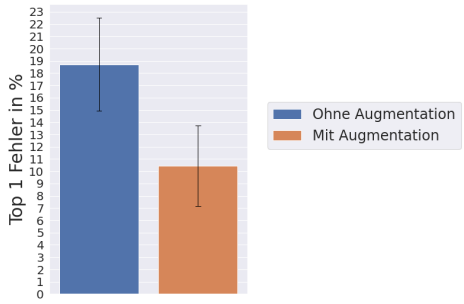
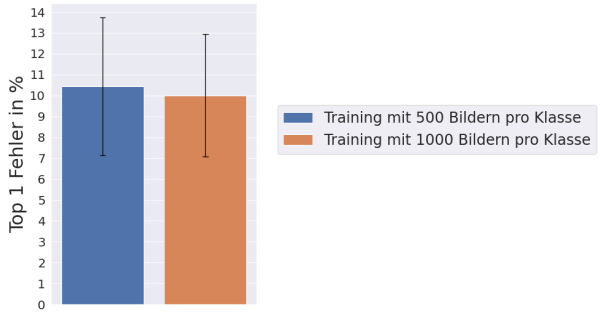


Abbildung 21: Durchschnittlicher Top 1 Fehler der Experimente 0 und 1. Niedrigere Werte sind besser. Schwarze Linien geben die Standardabweichung an.

Als nächstes wurde sich mit der Frage befasst, ob eine Generierung von mehr Bildern auch eine bessere Performance zur Folge hat. Daher wurde ein Experiment 1.2 durchgeführt, für welches 1000 Bilder pro Klasse generiert wurden.
In Abbildung 22 sind die durchschnittlichen Top 1 Fehler abgebildet.

[6]CPU: Amd Ryzen 3600X (3,8 GHz),GPU: Nvidia RTX 2070 8GB vRAM, RAM: 32 GB

Abbildung 22: Durchschnittlicher Top 1 Fehler in % der Experimente 1 und 1.2. Niedrigere Werte sind besser. Schwarze Linien geben die Standardabweichung an



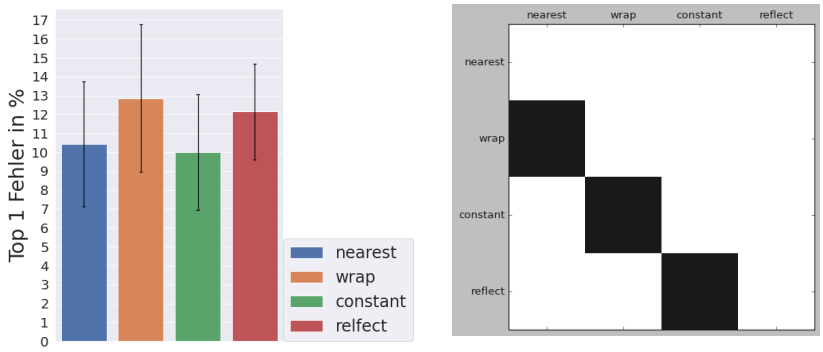
Um herauszufinden, ob die Unterschiede der Mittelwerte zwischen den einzelnen Experimenten signifikant sind, wurde der Welch-t Test genutzt (vgl hierzu Kapitel 2.5). Unter der Nullhypothese, dass die Mittelwerte gleich sind, der Alternativhypothese, dass die Mittelwerte ungleich sind, und mit einem Signifikanzniveau von $\alpha = 0.05$ ergibt sich ein p-Wert von 0.6732. Somit lässt sich die Nullhypothese nicht ablehnen. Die Performance sollte durch das Generieren einer größeren Anzahl von Bildern weder verbessert noch verschlechtert werden. Der wahre durchschnittliche Top 1 Fehler sollte bei beiden Experimenten statistisch gesehen gleich sein. Aufgrund der begrenzten verfügbaren Zeit wurden alle weiteren Experimente daher nur mit 500 Bildern pro Klasse durchgeführt.

Ein weiterer Faktor beim Augmentieren ergibt sich daraus, wie die leeren Ränder, welche zum Beispiel durch das Drehen oder Verschieben eines Bildes entstehen, gefüllt werden. Die verschiedenen in Kapitel 4.1.1 vorgestellten Methoden wurden in den Experimenten 1 bis 4 getestet, wobei alle anderen Parameter in den Experimenten gleich gelassen wurden. In Abbildung 23a ist der durchschnittliche Top 1 Fehler jedes Experiments zu sehen.

Abbildung 23

(a) Durchschnittlicher Top 1 Fehler der Experimente 1 bis 4. Niedrigere Werte sind besser. Schwarze Linie entspricht jeweils der Standardabweichung

(b) Welch t-test. Schwarze Box steht dafür das der p-Wert kleiner als $\alpha = 0.05$ ist.

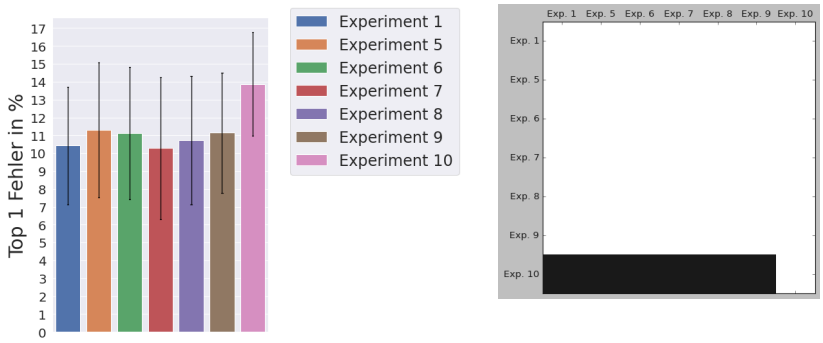


Um zu ermitteln, ob die Abweichung zwischen den einzelnen Experimenten signifikant ist, wurde erneut einen Welch-t Test durchgeführt mit der Nullhypothese, dass die Mittelwerte gleich sind, der Alternativhypothese, dass die Mittelwerte ungleich sind, und dem Signifikanzniveau von $0,05$. Die Ergebnisse wurden in Abbildung 23b visualisiert. Daraus ergibt sich, dass die Unterschiede zwischen *nearest* und *wrap*, *constant* und *wrap* sowie *constant* und *reflect* als signifikant anzusehen sind. Folglich ist zu erwarten, dass sich die Füllmethode *constant* zusammen mit *nearest* am besten für das Training eignet. Zwischen *wrap* und *reflect* sowie zwischen *constant* und *nearest* kann die Nullhypothese, dass die wahren durchschnittlichen Top 1 Fehler gleich sind, nicht abgelehnt werden.
Als letztes werden die durchschnittlichen Top 1 Fehler aller restlichen Experimente mit Experiment 1 und untereinander in Abbildung 24a verglichen. Die Experimente 5 bis 9 schneiden dabei alle sehr ähnlich ab und unterscheiden sich im Top 1 Fehler um höchstens einen Prozentpunkt Punkt. Mit dem Welch-t Test kann eine Signifikanz dieser Unterschiede auch nicht gezeigt werden(Vgl. Abbildung 24b). Daraus folgt für das Experiment, dass die Bildaugmentierung mit der Höhen- und Breitenverschiebung, das Zoomen auf sowie weg vom Bild und die Veränderung der Helligkeit keine relevanten Vorteile für das Training eines CNN mit keltischen Münzdaten erbringt.

Abbildung 24

(a) Durchschnittlicher Top 1 Fehler in %. Kleinere Werte sind besser. Schwarze Linie entspricht jeweils der Standardabweichung

(b) Welch t-test. Schwarze Box steht dafür, dass der p-Wert kleiner als α ist.



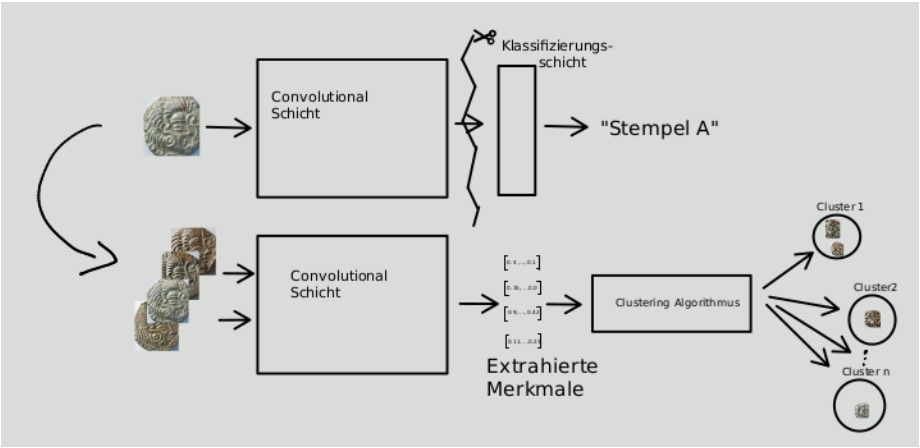
Experiment 10 sticht in Abbildung 24 mit einem Top 1 Fehler von 13.86% gegenüber den anderen Werten deutlich heraus. Diese Abweichung ist signifikant zu den anderen Experimenten in Abbildung 24b. Das zeigt auf, dass eine deutlich höhere Rotationsreichweite von 90 nicht vorteilhaft ist.
Darüber hinaus könnte man natürlich noch viele weitere Tests machen, wie z.B. die Rotationsreichweiten zwischen 0 und 45 sowie 45 und 90 oder etwa andere Werte bei anderen Argumenten. Allerdings war hiervon keine deutlich bessere Performance zu erwarten, da die Unterschiede zwischen den Experimenten in Abbildung 24a schon so gering sind. So ergibt sich aus Abbildung 24a auch, dass die Art der Augmentation nicht der entscheidende Faktor ist, sondern vielmehr die Anzahl an Bildern.
Eine mögliche Fehlerquelle in den Experimenten könnte sich daraus ergeben, dass der Testdatensatz für die Berechnung des Top 1 Fehlers relativ klein ist. Dieser enthält lediglich 35 Bilder, also ca. zwei pro Klasse, sodass die geringen Unterschiede zwischen den Mittelwerten auch aus der Größe des Testdatensatzes resultieren könnten. Da die Menge an klassifizierten Münzen sehr begrenzt ist, lässt sich die Größe des Testdatensatzes nicht ohne weiteres verändern, vor allem nicht erhöhen. Eine Augmentation der Bilder im Testdatensatz wird auch nicht als sinnvoll erachtet, da die Art der Augmentation einen Einfluss auf den Top 1 Fehler des jeweiligen CNN haben könnte. Experimente, deren Trainingsdatensatz ähnlich augmentiert wurden wie der augmentierte Testdatensatz, könnten einen Vorteil bei der Erkennung der neu generierten Bilder haben und somit das Ergebnis verfälschen.
Weitere Experimente sind daher wohl erst zielführend, wenn eine größere Menge an klassifizierten Münzen zur Verfügung steht.
Ein weiterer negativer Einfluss auf die Ergebnisse der Experimente könnte sich schließlich auch aus einer fehlerhaften Grundwahrheit der Bilder ergeben. Da diese per Hand von einem Numismatiker klassifiziert wurden, kann man mögliche Fehler nicht zu 100 % ausschließen.

5 Clustering

5.1 Extrahierung der Merkmale

Die Extrahierung der Merkmale erfolgt mit Hilfe eines CNN. Dabei wird die Klassifikationsschicht des CNN entfernt, sodass das CNN die gelernten Merkmale der Filter der Convolutional-Schichten ausgibt. Da die Ausgabe der letzten Convolutional-Schicht die Dimension $(H \times W \times F)$ beträgt, wird diese Ausgabe mittels einer Poolingschicht[7] auf einen Vektor der Länge F reduziert, sodass über diese ein Clustering gebildet werden kann. H und W entsprechen dabei der Höhe bzw. Breite der *feature map* und F gibt die Anzahl an Filtern an. Die ResNet Architektur liefert für jede Münze einen Vektor der Größe 2048.
Diese Vektoren werden mit einem Clusteringalgorithmus in Cluster eingeteilt, wobei die Münzen in einem Cluster sich möglichst ähnlich sein sollten.
Diese Idee des Clusterings entstammte aus dem Forschungsprojekt zur Clusterbildung von antiken Münzen[For] und dem wissenschaftlichen Artikel von Guerin et. al [GGTN18].

Abbildung 25: Visualisierung der Methode.



[7] Für diese Arbeit wird average Pooling benutzt. Dabei wird der Mittelwert der $H \times W$ *feature map* errechnet und ausgegeben.

5.2 Aufbau und Implementierung

5.2.1 Wahl des Clusteringalgorithmus

Für die Auswahl der Clusteringalgorithmen wurden die Ergebnisse von Guerin et.al [GGTN18] mit einbezogen. Dort haben K-means und Agglomerative Hierarchical Clustering am besten abgeschnitten, weshalb diese auch als Grundlage für diese Arbeit dienen.

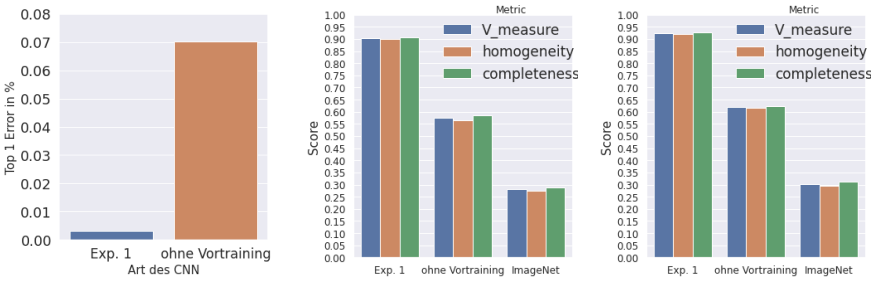
5.2.2 Implementierung

Für die Implementierungen der Algorithmen werden zwei Python Pakete genutzt. Für die K-means Clusterbildung wurde das Paket scikit-learn Version 0.23.1.[Kme] und für das hierarchische Clustering das Python Paket SciPy Version 1.5.1.[Sci] gewählt, da SciPy die Erstellung eines Dendrogramms besser unterstützt im Vergleich zu scikit-learn. Es wurden bestehende Implementierungen gegenüber einer eigenen Implementierung vorgezogen, da diese optimiert und weitaus weniger fehleranfällig sind.
Im Gegensatz zum hierarchischen Clustering muss für K-means die Größe des Clusterings angegeben werden. Bei genauerer Betrachtung der bereits gefundenen Stempel (siehe Tabelle 3) fällt auf, dass viele Stempel mit einem Fragezeichen versehen sind. Wie in Kapitel 4.1.2 werden diese aus der Menge an gesuchten Stempeln herausgenommen. Auch den Stempel HB wurde nicht mit einbezogen, da angenommen wird, dass dieser dem Stempel Hb entspricht und es sich nur um einen Tippfehler handelt.
So bleiben 36 Stempel übrig, welche als Größe für das K-means Clustering festgelegt werden. Weitere Implementierungen, beispielsweise für die Erstellung der Plots, befinden sich im Anhang.

5.3 Clusterbildung und Evaluation

5.3.1 Voruntersuchung

Im Folgenden soll zunächst die Frage erörtert werden, ob das Training eines CNN mit den keltischen Münzen einen Vorteil für die spätere Clusterbildung erbringt. Dafür wurden drei verschiedene Cluster mit K-means gebildet, wobei der Unterschied zwischen den Clustern darin besteht, dass die extrahierten Merkmale jeweils von einem anderen CNN erstellt wurden. Konkret handelt es sich um das gespeicherte CNN von Experiment 1, das CNN, welches ohne Vortraining trainiert wurde, und zuletzt ein CNN mit Gewichten, welche auf dem ImageNet Datensatz vortrainiert wurden. Die Cluster wurden nur mit den Test- und Trainingsdaten aus Kapitel 4 gebildet, da so eine genaue Clustergröße und Grundwahrheit zugrunde liegt. Es wurden alle Default Argumente übernommen und der Seed wurde für die Replizierbarkeit der Clusterings auf 1 festgesetzt.
Aus Abbildung 26b wird ersichtlich, dass das CNN aus Experiment 1 am besten abschneidet. Zusätzlich wurden drei hierarchische Cluster gebildet, deren Ergebnisse decken sich mit denen von K-means(Vergleich Abbildung 26c).



(a) Vergleich des Top 1 Fehlers.
Niedrigere Werte sind besser.

(b) Werte der Metriken für K-means.
Höhere Werte sind besser.
Annahme, dass die Grundwahrheit richtig ist.

(c) Werte der Metriken für das hierarchische Clustering.
Höhere Werte sind besser.
Annahme, dass die Grundwahrheit richtig ist.

Hieraus wird ersichtlich, dass sich für das Clustering ein vortrainiertes CNN am besten eignet, da die extrahierten Merkmale die Unterschiede der Stempel deutlich hervorheben. Im Gegensatz zu einen CNN, welches nicht vortrainiert wurde, können hier anscheinend viel komplexere Merkmale gelernt werden.
Ein Tool für die Visualisierung von höher dimensionalen Daten ist das t-distributed Stochastic Neighbor Embedding, kurz TSNE[Tsn]. Dieses errechnet zuerst auf den höher dimensionalen Datensatz eine Ähnlichkeitsmatrix zwischen den Datenpunkten. Danach werden die Daten zufällig auf eine festgelegte kleinere Dimension projiziert und auf der kleineren Dimension wird ebenso eine Ähnlichkeitsmatrix errechnet. Danach wird die Ähnlichkeitsmatrix der kleineren Dimension schrittweise so angepasst, dass diese der höher dimensionalen Ähnlichkeitsmatrix entspricht.[8]
Betrachtet man die Visualisierung der extrahierten Merkmale in Abbildung 27, sieht man, dass die extrahierten Merkmale von Münzen des gleichen Stempels beim Experiment viel enger beieinander liegen als bei den anderen beiden. Die extrahierten Merkmale des CNN ohne Vortraining zeigen auch hilfreiche Gruppierungen auf, jedoch nicht annähernd so Gute wie die des Experiments 1. Dennoch sind die Ergebnisse besser als die des CNN, welches auf den den ImageNet Datensatz trainiert wurde. Daher kann davon ausgegangen werden, dass die gelernte Repräsentation für die Clusterbildung einen großen Vorteil schafft, da die Gruppierungen der Daten in Abbildung 27a und 27b deutlich besser sind als in Abbildung 27c.

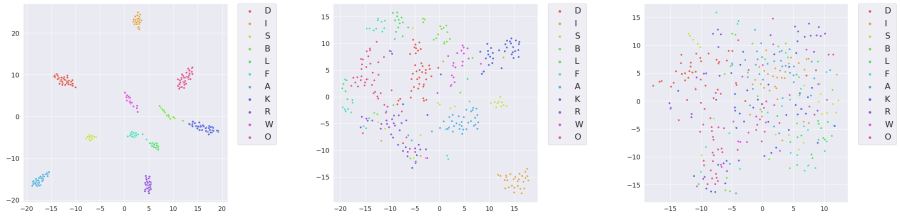
[8]Die Visualisierung durch TSNE ist nicht eindeutig. Für verschiedene Parameter können unterschiedliche Ergebnisse auftreten.

Abbildung 27: Visualisierung der extrahierten Merkmale mit TSNE.

(a) Die extrahierten Merkmale des Experiment 1

(b) Die extrahierten Merkmale des CNN ohne Vortraining

(c) Die extrahierten Merkmale des CNN mit ImageNet Gewichten



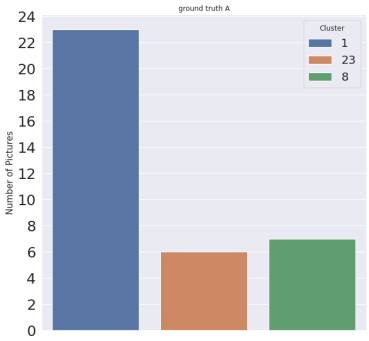
5.3.2 Clusterbildung über alle Münzen

Zur Clusterbildung über alle Münzen wurde zunächst ein Clustering mit K-means gebildet. Für die Errechnung des V-Measure Score wurden alle Münzen aus dem Cluster entnommen, die nicht zu den 36 gesuchten Stempeln gehören. Somit stellt die Grundwahrheit diese 36 Stempel und ihre bereits klassifizierten Münzen dar. K-means erreichte über diese 36 Stempel einen V-Measure Score von 0.7499 (siehe Abbildung 29). Betrachtet man die einzelnen Cluster des Clustering genauer (siehe Abbildung 36), fällt auf, dass die Anzahl der vorkommenden Stempel pro Cluster sehr stark variiert. Auffällig ist auch, dass die Cluster mit einer geringen Anzahl an vorkommenden Stempeln jene Stempel enthalten, welche auch für das Training des CNN genutzt wurden. Dabei sind diese Stempel auf mehrere Cluster verteilt, wobei sie meist nur in einem Cluster mit Münzen sind, welche noch nicht klassifiziert sind(siehe z.B. Cluster 0,1,4,5,7).

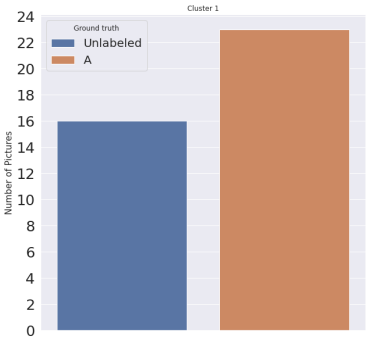
Im Folgenden soll nun der Stempel A genauer betrachtet werden im Hinblick auf die Verteilung und Zuordnung im Cluster. Abbildung 28a zeigt die genaue Verteilung des Stempels. Dabei sind knapp 64 % der Münzen des Stempels auf Cluster 1 verteilt und der Rest zu annähernd gleichen Anteilen auf Cluster 23 und 8.
Cluster 1 sowie Cluster 8 sehen von der Zuteilung der Münzen sehr vielversprechend aus. Diese könnten nicht klassifizierte Münzen enthalten, welche dem Stempel A angehören. In Cluster 8 ist allerdings auch eine Münze des Stempels P enthalten, was möglicherweise auf eine fehlerhafte Zuteilung durch den Numismatiker oder auf ein fehlerhaftes Clustering hindeutet.

Abbildung 28: Visualisierung der Verteilung des Stempels A im K-means Clustering und die dazugehörigen Cluster

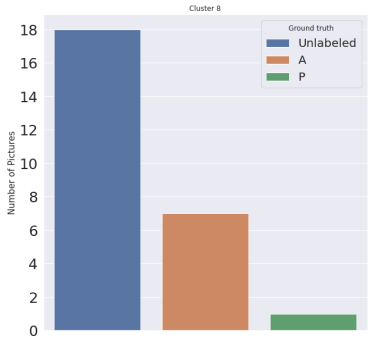
(a) Verteilung der Münzen des Stempels A im Clustering



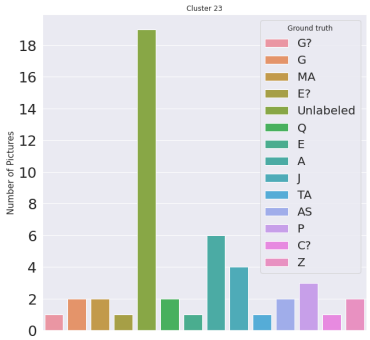
(b) Cluster 1 mit den vorkommenden Stempeln



(c) Cluster 8 mit den vorkommenden Stempeln



(d) Cluster 23 mit den vorkommenden Stempeln



Cluster 23 hingegen enthält zum größten Teil Münzen, die noch nicht klassifiziert sind, sowie sonstige Münzen von mehreren verschiedenen Stempeln. Hier stellt sich die Frage, warum die sechs Münzen des Stempels A diesem Cluster zugeordnet wurden. Es könnte wieder auf eine falsche Zuteilung durch den Numismatiker hindeuten oder aber die Extraktion der Merkmale durch das CNN ist noch nicht divers genug, um über die elf gelernten Klassen hinaus die Unterschiede hervorzuheben. Es wird von letzterer Annahme ausgegangen, da wie zuvor angemerkt, die Cluster mit geringer Streuung jene sind, mit denen das CNN trainiert wurde. Für ein genaueres Urteil über das Clustering wäre jedoch eine Überprüfung durch einen Numismatiker sehr sinnvoll.
Da aus dem K-means Clustering nicht deutlich gemacht wird, wieso beispielsweise der Stempel A sowohl in Cluster 1 als auch in Cluster 8 vorkommt und was diese Cluster genau voneinander unterscheidet, wurde ein hierarchisches Clustering erstellt. Ein solches bietet weiterreichende Möglichkeiten, die einzelnen Cluster genauer zu untersuchen, da durch das Dendrogramm ersichtlicher wird, wie sich einzelne Cluster zusammensetzten.

Beim hierarchischen Clustering wurden zunächst 36 Cluster erstellt und diese untersucht. Der V-Measure Score (siehe Abbildung 29) entspricht etwa dem von K-means.

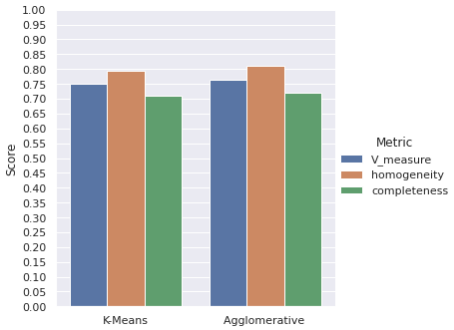
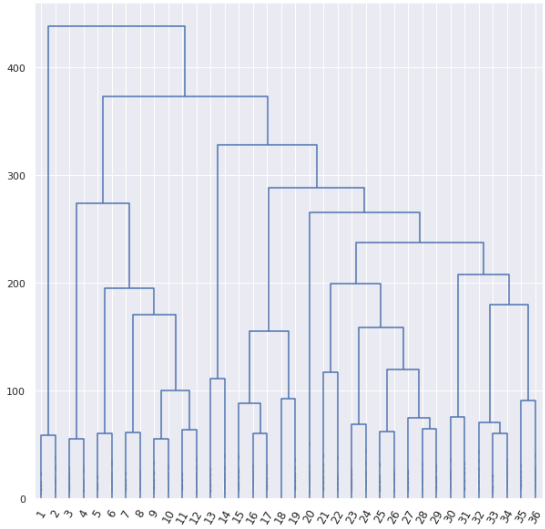


Abbildung 29: Werte der Metriken für beide Clusterings. Höhere Werte sind besser. Annahme, dass die Grundwahrheit richtig ist.

Betrachtet man nun das Dendrogramm (Abbildung 30) und vergleicht dieses mit der Verteilung der Münzen in die einzelnen Cluster (Abbildung 37), erkennt man, dass die Cluster, welche eine größere Anzahl an gleichen Stempeln enthalten, näher beieinander liegen(Vergleich Cluster 1 und 2, 3 und 4, 5 und 6).

Abbildung 30: Dendrogramm mit den Clustern als Label



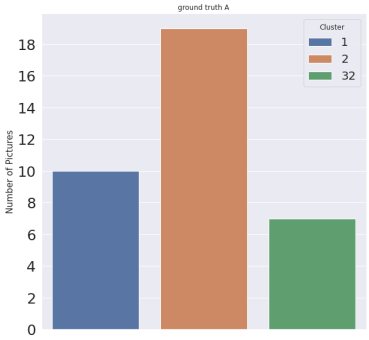
Aus Abbildung 31a wird jedoch deutlich, dass auch einzelne Münzen des Stempels A sehr weit entfernt von den Clustern 1 und 2 liegen. Dasselbe ist

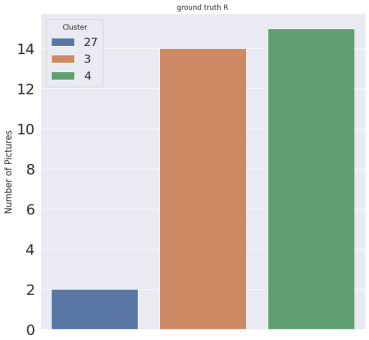
auch bei den Bildern der Klasse R zu beobachten (Abbildung 31b). Bei Stempel K ist die Streuung ein wenig größer, jedoch liegen hier die Cluster deutlich näher beisammen. Der Stempel E, welcher nicht Teil des Trainingsdatensatzes des CNN war, ist relativ weiträumig über das ganze Cluster verteilt.

Abbildung 31: Verteilung der Stempel A, R, K und E im Cluster

(a) Münzen von A

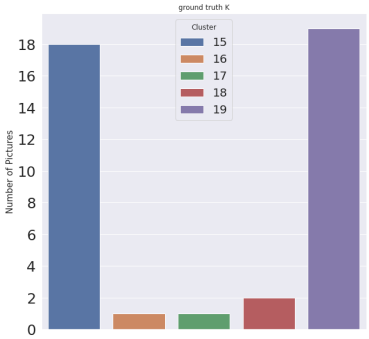
(b) Münzen von R

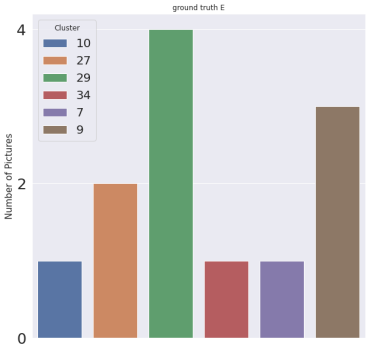




(c) Münzen von K

(d) Bilder von E





Beim hierarchischen Clustering fällt wie beim K-means Clustering letztlich auf, dass eine geringe Diversität an Stempeln nur in jenen Clustern vorkommt, die aus Stempeln des Trainings stammen. Daher wird das Dendrogramm auf 11 Stempel verkleinert, um heraus zu finden, wie die Münzen der Stempel, mit denen das CNN trainiert wurde, verteilt sind.

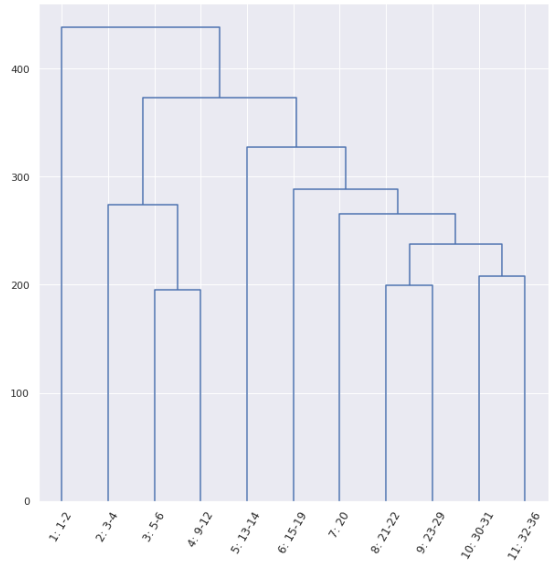


Abbildung 32: Überblick der 11 Cluster. x: a-b ist zu verstehen als: x neues Cluster, setzt sich aus den alten Clustern a-b zusammen

In Tabelle 1 kann man erkennen, dass sich jeder Stempel des Trainings einem Cluster zuordnen lässt. Cluster 1, 2, 3, 5, 7, 8 und 10 weisen eine sehr geringe Streuung auf, weshalb davon auszugehen ist, dass die nicht klassifizierten Bilder in den Clustern auch zu den Stempeln passen. Dies müsste allerdings durch einen Numismatiker verifiziert werden. Bei den Clustern 4, 5, 6, 9 und 11 wäre es sinnvoll, tiefere Abschnitte des Dendrogramms zu betrachten, indem man sich mehr als 36 Stempel ausgeben lässt und die Verteilung in den Clustern näher betrachtet. Hierdurch könnte man beispielsweise in Erfahrung bringen, wieso die Münzen des Stempels A dem Cluster 11 zugeordnet wurden.

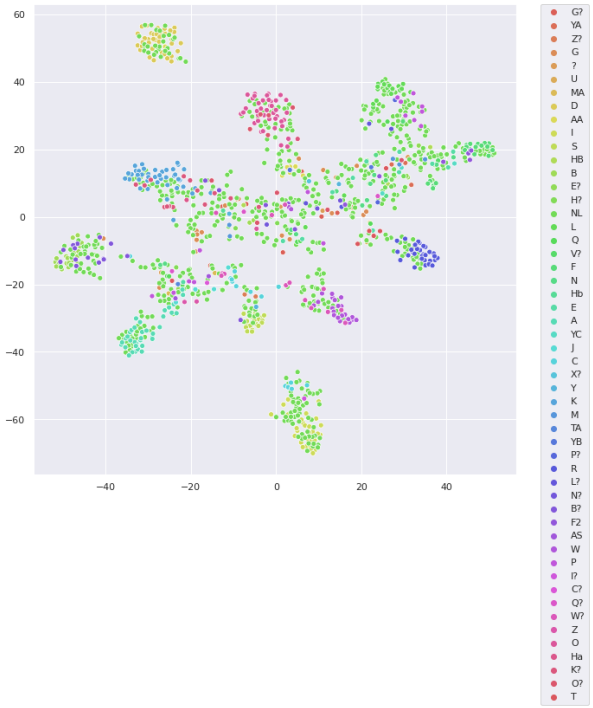
Tabelle 1: Übersicht der 11 Cluster

Cluster	1	2	3	4	5	6	7	8	9	10	11
Anzahl Bilder	53	49	35	182	98	162	47	59	304	64	162
Am häufigsten vorkommender Stempel	A	R	L	Unlabeled	Unlabeled	Unlabeled	D	Unlabeld	Unlabeled	Unlabeld	Unlabeld
zweithäufigster Stempel	Unlabeled	Unlabeled	Unlabeled	F	I	K	Unlabeled	W	O	B	S
Anzahl verschiedener Klassen	2	2	4	19	4	15	2	3	30	4	27

Des Weiteren wurde überprüft, ob die Münzen aus Cluster 1 (des hierarchischen Clusterings mit 11 Stempel aus Abbildung 32) auch in den entsprechenden Clustern des K-means Clustering vorkommen. Dabei finden sich alle Münzen des Stempels A vom Cluster 1 in den Clustern 1 und 8 von K-means. Lediglich eine Münze des Stempels A kommt zusätzlich in den Clustern von K-means vor, welche nicht in Cluster 1 des hierarchischen Clustering enthalten ist. Für die nicht klassifizierten Münzen aus Cluster 1 des hierarchischen Clustering gilt dasselbe,

es kommen also alle in den K-means Clustern 1 und 8 vor, jedoch enthalten diese 10 weitere nicht klassifizierte Münzen. Hieraus wird deutlich, dass die Ergebnisse der beiden Algorithmen sich stark ähneln.
Insgesamt lässt sich aus der Verteilung der Stempel erschließen, dass die extrahierten Merkmale einer Münze durch das CNN die Unterschiede zwischen den gelernten Stempeln im Clustering deutlich hervorheben können. Die Repräsentation der Münzen ist jedoch für Stempel, welche nicht im Training enthalten waren, nicht ausreichend. Diese Stempel kommen nämlich hauptsächlich in Clustern vor, die eine große Anzahl an verschiedenen Stempeln umfassen (siehe in Abbildung 36 z.B. Cluster 9 und 35 und Abbildung 37 z.B. Cluster 10 und 28). Dies trat sowohl beim hierarchischen Clustering als auch beim K-means Clustering auf.
Jene Cluster, welche eine große Streuung besitzen, bestehen größtenteils aus Münzen ohne bekannten Stempel. Es ist daher auch möglich, dass hier ein neuer Stempel gefunden wurde, jedoch lässt sich dies als Laie nicht ohne Weiteres beurteilen.
Die Visualisierung der extrahierten Merkmale aller Münzen durch TSNE zeigt, dass die Merkmale noch nicht differenziert genug sind. Die Struktur ähnelt dabei derjenigen in Abbildung 27a, wobei eine größere Streuung um die 11 Gruppierungen zu sehen ist.

Abbildung 33: Visualisierung aller extrahierten Merkmale durch TSNE.



Für einen Numismatiker könnte diese Methode des Clustering durchaus hilfreich bei der Stempelanalyse sein. So können Bilder schon vorsortiert werden, was die Analyse ein wenig beschleunigen könnte. Allerdings ist diese Methode auch stark an die Grundwahrheit und die Anzahl der bereits klassifizierten Münzen gekoppelt. Zudem ist die genaue Anzahl an Stempeln unbekannt und schwer zu ermitteln, was die Festlegung der Größe für das Clustering erschwert. Daher ist anzunehmen, dass sich eine größere Anzahl an klassifizierten Münzen von einer größeren Anzahl an Stempeln, wodurch sich das CNN mit mehr Klassen trainieren lässt, positiv auf das Ergebnis des Clustering auswirken könnte. Auch eine stärkere Zusammenarbeit zwischen Informatiker und Numismatiker kann hier das Ergebnis des Clustering deutlich beeinflussen, da man als Informatiker ohne ein größeres Wissen über die Domäne viele falsche Entscheidungen treffen könnte, die sich wiederum negativ auf das Training sowie das Clustering auswirken können.

6 Vergleich zu bestehenden Methoden

Eine weitere computergestützte Methode für die Stempelanalyse stellten Dr. Peter van Alfen and Zachary Taylor 2020 vor [Die]. Der Vorteil der vorgestellten Methode besteht darin, dass keine bereits klassifizierten Münzen benötigt werden. Wie in dieser Arbeit wird das hierarchische Clustering benutzt. Jedoch unterscheidet sich die Methode zur Extrahierung der Merkmale.
Z.Taylor et al. extrahieren die Merkmale mit dem *ORB*(Oriented FAST and Rotated BRIEF) Algorithmus[RRKB11].*ORB* ist eine Kombination aus dem Algorithmus *FAST*, welcher für die Erkennung von Merkmalen in Bildern genutzt wird, sowie einer abgewandelten Version des *BRIEF* Algorithmus, welcher für die Beschreibung der Merkmale genutzt wird, beispielsweise bezüglich ihrer Form oder Orientierung im Bild.[9]

Abbildung 34: Beispiel der Merkmalerkennung für die Münze CATII-H-15560A durch *ORB*. *ORB* erkennt in diesem Beispiel mögliche relevante Merkmale der Münze nicht, wie etwa Nase oder Mund. Auch der Bruch erschwert eine Identifizierung der Stempel, da dieser nicht zu einem Stempelmerkmal gehört.

Die von *ORB* erkannten Merkmale der Münzen werden mithilfe des *Brute force descriptor matcher* miteinander verglichen. Hierbei wird jedes Merkmal einer Münze mit allen Merkmalen einer anderen Münze verglichen und dann ein Distanzwert für jeden dieser Vergleiche errechnet. Danach wird ein globaler Distanzwert zwischen den Münzen errechnet, indem über die 20 geringsten Distanzen der verglichenen Merkmale ein Durchschnittswert errechnet wird.
Die Clusterbildung erfolgt grundsätzlich wie in dieser Arbeit, nur werden diejenigen Münzen miteinander gepaart, die den geringsten globalen Distanzwert zueinander aufweisen. Die Methode zur Extrahierung der Merkmale durch *ORB* und dem Vergleich der Merkmale zwischen Münzen wurde als Vorbereitung auf diese Arbeit getestet. Allerdings schienen die Ergebnisse nicht sehr vielversprechend zu sein, was sich wie folgt darstellt.

[9] *ORB* ist dem Algortithmus *SIFT* sehr ähnlich. Dieser wurde bereits in einer vorangegangenen Bachelorarbeit von H. Loung [Luo19] zur Erkennung von Monogrammen genutzt. Diese Arbeit könnte zum besseren Verständnis von *ORB* dienen.

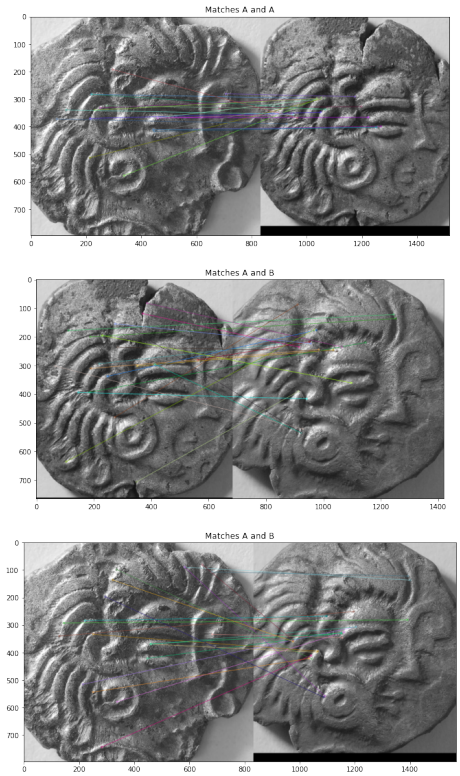


Abbildung 35: Test der Merkmalextrahierung durch *ORB* mit anschließendem Matching durch den *Brute force descriptor matcher*. Es wurden jeweils die 20 besten Treffer ausgegeben. Stempel der Münzen in der Überschrift enthalten.

Wie in Abbildung 35 zu sehen ist, sind die gepaarten Merkmale zweier Münzen häufig unzutreffend. DEs ist daher nicht davon auszugehen, dass mit dieser Methode bessere Ergebnisse für das Clustering der keltischen Münzen erzielt werden können. Der bedeutendste Vorteil der Methode von Z.Taylor et al. gegenüber der Methode dieser Arbeit ist allerdings, dass keine vorherige Klassifizierung der Münzen notwendig ist. Sie kann so für jeden möglichen Datensatz an Münzen eingesetzt werden. Hingegen ist die Methode dieser Arbeit stark an die Anzahl der klassifizierten Münzen gebunden, wie in Kapitel 5.3.2 erörtert wurde.

Da es sich bei der Stempelanalyse um ein sehr fachspezifisches Aufgabenfeld handelt, sind die vorzufindenden Methoden hierfür stark begrenzt.
Es gibt jedoch viele weitere Ansätze für die Bildklassifizierung oder für die Clusterbildung anderer Datensätze, wobei diese mit deutlich größeren Datenmengen arbeiten; so etwa der Ansatz von Mathilde Caron, Piotr Bojanowski, Armand Joulin, und Matthijs Douze[CBJD18]. Caron et al. trainieren ein CNN zur Klassifikation von Bildern. Beim Training des CNN werden *pseudo-label* erstellt, indem Cluster über die extrahierten Merkmale der Bilder erstellt werden.

Diese Cluster bilden somit die Grundwahrheit für jede Epoche des Trainings. Der Vorteil dieser Methode ist, dass für das Training des CNN keine klassifizierten Daten zur Verfügung stehen müssen. Allerdings eignet sich der Ansatz nicht für die Zwecke diese Arbeit, da hierfür die genaue Anzahl an Stempeln bekannt sein oder man sich zumindest auf eine bestimmte Anzahl an Stempeln beschränken müsste. Da diese Arbeit durch die Anzahl der zur Verfügung stehenden Daten relativ beschränkt ist, ist ein Vergleich zu diesen Ansätzen wohl weniger repräsentativ.
Die wahrscheinlich präziseste und beste Methode für die Stempelanalyse ist hingegen die Analyse per Hand durch einen Numismatiker. Die Unterschiede zwischen den Stempeln sind zum Teil so gering, dass sich eine exakte Identifizierung durch den Computer ohne die Hilfestellung in Form von klassifizierten Daten als schwierig erweist. Auch für einen Laien ist das Erkennen der Stempel keine triviale Aufgabe. Sowohl die Methode von Z. Taylor et al. als auch die Methode dieser Arbeit erfordern in einem letzten Schritt immer die Verifizierung durch einen Numismatiker. Dieser kann zumindest durch die beiden vorgestellten Methoden nicht ersetzt werden, was die Schwierigkeit dieser Aufgabe verdeutlicht.

7 Fazit und Ausblick

Die hier genutzte Methode für die Stempelanalyse hat gezeigt, dass die Extrahierung von Merkmalen durch ein CNN durchaus auf Münzdaten angewendet werden kann. Allerdings bleibt sie stark abhängig von der Art des Datensatzes, auf dem das CNN trainiert wurde. Die Merkmale für die Stempelanalyse sind so detailliert und spezifisch, dass ein weitreichendes Vortraining nötig ist.
Das Training der CNNs in Kapitel 4 mithilfe der Datenaugmentation erwies sich als überraschend gut. Dieses Ergebnis könnte, wie bereits angesprochen, von dem geringen Testdatensatz verfälscht sein. Daher wäre eine erneute Überprüfung der Performance durchaus sinnvoll, sobald mehr klassifizierte Daten zur Verfügung stehen.
Auch ist zu berücksichtigen, dass die Arbeit nur einen geringen Teil der Möglichkeiten abdeckt, die zur Verfügung stehen. Es gibt viele kleine Stellschrauben, die man drehen kann, welche das Ergebnis des Clustering stark beeinflussen könnten. So könnte die Wahl der ResNet Architektur oder die Wahl des Clusteringalgorithmus nicht optimal gewesen sein. Hier gibt es noch eine Vielzahl an Algorithmen und Architekturen, die getestet werden könnten.
Interessant wäre es auch, die Merkmale mit dem trainierten CNN aus der Arbeit von A.Loyal[Loy18] zu extrahieren, welches auf einen Datensatz mit römischen Münzen trainiert wurde. Hier stellt sich die Frage, ob dieses einen Vorteil für die Merkmalextrahierung bringt.
Eine weitere Möglichkeit bestünde darin, die extrahierten Merkmale vor dem Clustering durch einen Algorithmus, wie beispielsweise TSNE, zu verändern. Abbildung 27a zeigt nämlich auf, dass dieser die Daten sinnvoll gruppieren kann und reduziert dabei die Dimensionen, wodurch sogar die Berechnungszeit für das Clustering verkürzt werden kann. Des weiteren könnte man auch einen Autoencoder für die Merkmalextrahierung nutzen. Ein Autoencoder ist ein neuronales Netz, welcher aus einem Endcoder und Decoder besteht. Der Encoder komprimiert hierbei die Daten, welche von dem Decoder wieder dekompriminert werden. Dabei könnte man die Ausgabe des Encoders nehmen und darüber ein Clustering bilden.
Auch eine Kombination der Clusterbildung des Ober- und Unterstempel könnte getestet werden, indem von beiden Seiten Merkmale extrahiert werden und die Cluster über beide Merkmale vereint gebildet werden. Es stehen somit viele Möglichkeiten zur Verfügung, diese Methode zu verändern. Leider war eine Durchführung und Auswertung infolge der kurzen Bearbeitungszeit nicht möglich. Es ist jedoch davon auszugehen, dass eine engere Zusammenarbeit Informatiker und Numismatiker die besten Verbesserungschancen bietet.

8 Literaturverzeichnis

Literatur

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, January 2006.

[CBJD18] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[Cho18] Francois Chollet. *Deep learning with Python*. Manning Publications, January 2018.

[Clu] Scikit learn clustering performance evaluation. `https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation`. Accessed: 1-11-2020.

[Coi] Introduction to numismatic terms and methods. `http://numismatics.org/Seminar/TermsMethods/`. Accessed: 24-11-2020.

[Die] Coins and computation: New developments in the computer-aided die study. `https://www.academia.edu/43328887/Coins_and_Computation_New_Developments_in_the_Computer_Aided_Die_Study`. Accessed: 24-9-2020.

[For] Forschungsprojekt: Ancient coins clustering. `https://github.com/maccuryj/AncientCoins`. Accessed: 24-10-2020.

[Gam19] S. Gampe. Kombination maschineller lernmethoden der bild- und-texterkennung auf antiken münzdaten, 2019.

[GGTN18] Joris Guérin, Olivier Gibaru, Stéphane Thiery, and Eric Nyiri. Cnn features are also great at unsupervised classification, 2018.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

[Imaa] Image net. `http://image-net.org/`. Accessed: 24-11-2020.

[Imab] Imagedatagenerator. `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`. Accessed: 24-10-2020.

[JB07] Christof Schuster Jürgen Bortz. *Statistik für Human- und Sozialwissenschaftler*. Springer Verlag, 2007.

[JP18] Adam Gibson Josh Patterson. *Getting started with deep learning*. O'Reily, 2018.

[Kah05] Helmut Kahnt. *Das Grosse Münzlexikon*. Gietl Verlag, January 2005.

[Kme] Scikit learn k-means clustering. `https://sklearn.org/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans`. Accessed: 19.11.2020.

[Loy18] A. Loyal. Maschinelles lernen angewendet auf bilder antiker münzen. masterarbeit, 2018.

[Luo19] H. Luong. Automatisierte bilderkennung vonmonogrammen mit opencv, 2019.

[MR10] Oded Maimon and Lior Rokach, editors. *Data Mining and Knowledge Discovery Handbook, 2nd ed*. Springer, 2010.

[Mue] Jersey's celtic coin hoard. `https://www.jerseyheritage.org/jersey-s-celtic-coin-hoard`. Accessed: 12.11.2020.

[PW17] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[RH07] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[RRKB11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. ORB: an efficient alternative to SIFT or SURF. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc Van Gool, editors, *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pages 2564–2571. IEEE Computer Society, 2011.

[SCD+19] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, Oct 2019.

[Sci] Scipy hierarchical clustering. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html#id2`. Accessed: 18-11-2020.

[T-t] Scipy t-test, howpublished = https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html, note = Accessed: 24-10-2020.

[TfE] Earlystopping. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping. Accessed: 24-10-2020.

[Tsn] t-distributedstochastic neighbor embedding. https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html. Accessed: 28-11-2020.

9 Abbildungsverzeichnis

Abbildungsverzeichnis

10 Tabellenverzeichnis

Tabellenverzeichnis

11 Anhang

11.1 GitLab Repository

Die verwendeten Python Skripte und Beispiele sind alle auf GitLab hochgeladen. Unter `https://gitlab.com/Rob_Kra/celtic-coin-clustering`

11.2 Tabellen

Tabelle 2: Parameter aller Experimente. Die Felder mit roter Schrift sind die Unterschiede zu Experiment 1. Horizontale Spiegelung ist bei allen Experimenten auf True gesetzt. Es wurden in jedem Experiment die gleiche Anzahl an Bildern generiert(500).

Experiment	rotations range	height & width shift range	Zoom range	fill mode	brightness range	Durschnittlicher Top 1 Fehler in %	standard Abweichung
1	45	0.1	0.1	nearest	None	10,43	3,29
2	45	0.1	0.1	wrap	None	12.86	3.89
3	45	0.1	0.1	constant	None	10	3.06
4	45	0.1	0.1	reflect	None	12.14	2.54
5	45	0.1	None	nearest	None	11.29	3.78
6	45	None	0.1	nearest	None	11.13	3.7
7	45	None	None	nearest	None	10.29	3.98
8	45	0.1	0.1	nearest	(0.5,1.5)	10.71	3.6
9	45	0.05	0.05	nearest	None	11.14	3.37
10	90	0.1	0.1	nearest	None	13.86	2.9

Tabelle 3: Übersicht aller Stempel. Die Ordner ohne Bilder kommen wahrscheinlich dadurch zustande, dass beim automatischen Zuschneiden der Bilder kein Portrait erkannt wurde.

Stempel	?	A	AA	AS	B	B?	C	C?	D	E	E?	F	F2	G	G?	H	H?	Ha	Hb
Anzahl Bilder	4	36	7	2	21	19	14	1	39	12	4	22	7	20	8	0	4	1	1
Stempel	HB	I	I?	J	K	K?	L	L?	M	MA	N	N?	Unlabeld	O	O?	P?	P?	Q	Q?
Anzahl Bilder	3	32	1	4	41	15	21	3	5	3	13	3	657	42	10	15	2	7	3
Stempel	R	R?	S	R	TA	U	V?	W	W?	X?	Y	YA	YB	YC	Z	Z?			
Anzahl Bilder	31	0	19	6	1	1	1	24	14	1	3	3	2	3	2	2			

Abbildung 36: Überblick aller 36 Cluster des K-means Clusterings.
Größere Übersicht abrufbar im Ordner 'Experimente' des GitLab Repository

Abbildung 37: Überblick aller 36 Cluster des Hierarchischen Clusterings.
Größere Übersicht abrufbar im Ordner 'Experimente' des GitLab Repository