Erklärbarkeit von Merkmalserkennung keltischer Münzen bei Anwendung von Clusteralgorithmen mit Convolutional Neural Networks

Bachelorarbeit im Fach Informatik

an der Goethe Universität Frankfurt am Main

Vorgelegt von: Luisa Maria Cardenas Vasco
Erstgutachter: Dr. Karsten Tolle
Abgegeben am: 28.02.2022

Erklärung zur Abschlussarbeit

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

Hiermit erkläre ich

Luisa Maria Cardenas Vasco

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

Zudem versichere ich, dass alle eingereichten schriftlichen gebundenen Versionen meiner vorliegenden Bachelorarbeit mit der digital eingereichten elektronischen Version meiner Bachelorarbeit übereinstimmen.

Frankfurt am Main, 28.02.2022

Unterschrift

Abkürzungsverzeichnis

CNN(s) ..... Convolutional Neural Network(s)
Grad-CAM ..... Gradient-weighted Class Activation Mapping
Guided Grad-CAM ..... Guided Gradient-weighted Class Activation Mapping

Zusammenfassung

Der Bereich Maschinelles Lernen (ML) ist ein wichtiger Bestandteil der Künstlichen Intelligenz (KI). Besonders gute Ergebnisse erzielen Modelle des maschinenellen Lernens im Bereich der Objekt- und Texterkennung als auch in der Klassifikation von Bildern. Jedoch ist die Wahl der Entscheidungen solcher ML-Modelle nicht transparent. Somit hat ein Anwender Schwierigkeiten, diese zu verstehen.

Durch die Schwierigkeit von Black Box Szenarien in diesem Feld, gewinnt die erklärbare künstliche Intelligenz immer mehr an Relevanz. Dieser Teilbereich hat sich zum Ziel genommen, Wissen für einen Anwender transparenter zu gestalten, sodass dann daraus weitere Beobachtungen und Analysen durchgeführt werden können.

Diese Bachelorarbeit befasst sich genau mit dieser Thematik, dass Daten in Form von Bildern in einem ML-Modell verarbeitet werden. Die prozessierten Informationen können mit Methoden aus dem Bereich der erklärbaren künstlichen Intelligenz angeschaut werden. Dadurch, dass die Ergebnisse transparent gestaltet werden, bildet sich ein Fazit über die Bilder.

Inhaltsverzeichnis

1 Einleitung

Täglich werden Bilder in unterschiedlichen Bereichen analysiert und ausgewertet. Im Bereich der Medizin sind Bilder unersetzlich, da sie relevante Informationen über den Gesundheitszustand von Patienten liefern. Durch die Unterstützung von Bilderkennung, werden Anomalien festgestellt, sodass Ärzte bessere Diagnosen erstellen [Chizari (2020)].

Ein weiterer Bereich, der sich für Anwendungen der *künstlichen Intelligenz* interessiert, ist die Numismatik. Für kollaboratives Arbeiten werden die Münzen fotografiert, da die Funde teilweise in Museen aufgestellt werden und somit nicht für Forschungsaufgaben direkt zur Verfügung stehen. Der Bestand der Bilder ermöglichen Anwendungen der *künstlichen Intelligenz* für verschiedene Forschungsansätze in der Numismatik auszutesten. Die Klassifikation von Münzen ist eine mögliche Aufgabenstellung eines Numismatikers [für Bildung und Forschung (2021)]. *Convolutional Neural Networks* werden in der Informatik speziell für die Klassifikation von Bildmaterial genutzt. Solche Netzwerke werden mit Algorithmen trainiert, um eine Klassifikation zu generieren.

Der komplexe Aufbau von *Convolutional Neural Networks* erfordert die Entwicklung von Methoden, die die Transparenz solcher Netzwerke ermöglichen.

Der Fokus dieser Arbeit liegt auf die Erklärbarkeit von Merkmalserkennung keltischer Münzen. Da durch unüberwachtes Lernen die Klassifikation der Münzen erfolgt, ist es umso wichtiger herauszufinden, welche Merkmale für die Klassifikation relevant sind. In diesem Zusammenhang werden Methoden der *Erklärbaren Künstlichen Intelligenz* eingesetzt. Ziel ist es zu untersuchen, welche Merkmale für die Entscheidung des Modells verantwortlich sind und inwiefern diese Entscheidungen mit Hilfe von Erklärbarkeitsmethoden nach menschlicher Wahrnehmung nachvollziehbar sind.

2 Grundlagen

Diese Arbeit beschäftigt sich mit Inhalten aus dem Deep Learning Bereich, um Lösungen für die definierte Aufgabenstellung zu finden. Dieses Kapitel beinhaltet Grundlagen aus diesem Bereich, die wichtige Bausteine für diese Arbeit bilden und im weiteren Verlauf Anwendung finden. Wir geben wieder, was Lernen im Kontext des maschinellen Lernens bedeutet. Die Methoden des überwachten und unüberwachten Lernens werden angeführt. Darauffolgend gehen wir auf den allgemeinen Aufbau von CNNs ein, die spezielle neuronale Netze sind und gehen auf die AlexNet Architektur ein. Anschließend geben wir Clusteralgorithmen an, die Anwendungen des unüberwachen Lernen sind. Für die Erklärbarkeit von intelligente Anwendungen, werden verschiedene visuelle Methoden vorgestellt.

2.1 Methoden des maschinellen Lernens

Maschinelles Lernen ist ein Teilgebiet der Künstlichen Intelligenz. Nach Tom Mitchell lernt ein Computerprogramm aus Erfahrung in Bezug auf eine Aufgabe und ein Leistungsmaß, wenn sich seine Leistung beim Lösen der Aufgabe, gemessen an den Leistungsmaß, mit zunehmender Erfahrung verbessert [Goodfellow et al. (2016), Kapitel 5]. In diesem Kontext werden Daten aus einem Datensatz als Erfahrungen verstanden [François (2017), Kapitel 1]. In der Regel wird der Datensatz in einem Trainingsdatensatz, Validierungsdatensatz und in einem Testdatensatz unterteilt. Ersteres wird für das Training des Computerprogramms verwendet. Daraus resultiert ein trainiertes Modell. Mit den Validierungsdaten wird das Modell evaluiert. Der Testdatensatz wird auf das trainierte Modell angewendet, um die Leistung zu messen. Im maschinellen Lernen gibt es verschiedene Lernmethoden, wie überwachtes und unüberwachtes Lernen [François (2017)].

2.1.1 Überwachtes Lernen

Die Methode des überwachten Lernens verfolgt den Ansatz jedem Datenexemplar des Trainingssatzes einen Label – Markierung - zu vergeben. Im Label wird die gewünschte Lösung zum Datenexemplar vom Anwender, also nicht vom System selbst, vergeben. Das Modell lernt für jedes Datenexemplar die vergebene Lösung, sodass es auf neue Daten die richtige Lösung für die definierte Aufgabe findet. Eine typische Aufgabe ist die Klassifikation von Daten [François (2017), Kapitel 4].

2.1.2 Unüberwachtes Lernen

Beim unüberwachten Lernen werden die Datenexemplare des Trainingssatzes nicht mit Labels versehen. Das Ziel bei der Anwendung dieser Methode ist die Erkennung von zugrundeliegende Strukturen und Muster. Diese Methode wird oft zur Datenanalyse herangezogen, um Datensätze besser zu verstehen und Erkenntnisse zu gewinnen. Eine bekannte Anwendung für das unüberwachte Lernen ist Clustering [François (2017), Kapitel 4].

2.2 Deep Learning

Im maschinellen Lernen gibt es eine Vielzahl an Algorithmen, weshalb sich daraus eine Teilmenge ergeben hat, die als Deep Learning bezeichnet wird. In diesem Bereich sind Algorithmen zusammengefasst, die Darstellungen von Daten in Schichten erlernen. Eine Schicht ist eine Einheit, die die Daten verarbeitet. In der Regel werden die Darstellungen der Daten von Modellen gelernt, die als neuronale Netze bezeichnet werden. Die Tiefe solcher Modelle wird durch die Anzahl der Schichten definiert. Die Schichten haben Tensoren - mehrdimensionale Objekte - als trainierbare Parameter [François (2017)]. Nach Verarbeitung der Eingabe durch alle Schichten, wird die ermittelte Vorhersage ausgegeben. Mit der Verlustfunktion wird der Verlustwert zwischen Vorhersage und erwarteten Ergebnis berechnet, um die Performanz zu bewerten. Dieser Wert wird für die Eingabe des Optimierers genutzt, der die Anpassung der Parameter in den Schichten übernimmt. Ziel ist es die Menge

an Parameter zu finden, die die Velustfunktion am Meisten minimieren. Ein kompletter Durchlauf des Trainingsatzes wird als Epoche bezeichnet [François (2017), Kapitel 2, 3]. Bevor wir zum nächsten Unterkapitel übergehen, halten wir Definitionen fest, die für den weiteren Verlauf dieser Arbeit bedeutend sind.

Forward Propagation

Als forward propagation wird das vorwärts Fließen der Information von der Eingabeschicht bis zur Ausgabeschicht des Netzes bezeichnet. Ein forward pass ist somit ein Vorwärtsschritt im Netz [Goodfellow et al. (2016)].

Gradienten

Alle Operationen, die in einem neuronalen Netz verwendet werden, sind differenzierbar, deshalb kann ihre Ableitung ermittelt werden. Ein Gradient ist die Ableitung von Funktionen, die als Eingabe Tensoren erhalten [François (2017), Kapitel 2].

Backpropagation

Der verfolgte Ansatz ist, dass sich in differenzierbaren Funktionen das Minimum analytisch ermitteln lässt. Das Minimum einer Funktion ist ein Punkt, an dem die Ableitung null ist. Alle Punkte, an denen die Ableitung gegen null geht, werden gesucht, um den Punkt auszuwählen, für den die Funktion den niedrigsten Wert hat. Da in einem neuronale Netz nach einer Menge von Parameter gesucht wird, die eine Minimierung der Verlustfunktion bewirken, ist dieser Ansatz anwendbar [François (2017), Kapitel 2].

2.3 Convolutional Neural Network

Feedforward Modelle werden als neuronale Netzwerke bezeichnet, da sie durch die Verkettung von verschiedenen Funktionen dargestellt werden. Die Funktionen werden von innen nach außen betrachtet, dabei ist die innerste Funktion, die erste

Schicht des Netzwerks und die äußerste Funktion wird als Ausgabeschicht bezeichnet. Eine Funktion kann trainiert werden, um einer Zielfunktion zu entsprechen. Der Trainingssatz liefert ungefähre Beispiele zur Darstellung der Zielfunktion. Jedes Datenexemplar wird mit einem Label versehen, sodass in der Ausgabeschicht, zu jedes Datenexemplar ein Wert erzeugt wird, der Nahe am Label liegt. Das Verhalten der versteckten Schichten, also die innere Funktionen, wird nicht von den Trainingsdaten bestimmt. Vielmehr liegt die Entscheidung beim Lernalgorithmus, wie sich die versteckten Schichten zur Eingabe verhalten, um die Zielfunktion zu erfüllen [Goodfellow et al. (2016)]. CNNs sind Feedforward Modelle deren Struktur und Bausteine wir nachfolgend im Zusammenhang mit der Bilderkennung betrachten, da sie besonders in diesem Bereich eingesetzt werden und gute Leistungen erbringen [François (2017), Kapitel 2].

2.3.1 Aufbau

In diesem Abschnitt geben wir die Bausteine und Operationen an mit denen ein CNN konstruiert wird. Da in dieser Arbeit Bilddaten verwendet werden, gehen wir auf die Darstellung von Bilder in diesem Rahmen ein.

Bilddaten

In Bilder werden Objekte auf Pixelmengen projiziert. Aufeinanderfolge Pixel stellen zusammen ein Objekt bzw. ein Merkmal des Bildes dar. Änderungen der Pixelwerte, kann darauf hindeuten, dass dieser Pixel und aufeinanderfolgende Pixel zusammen ein anderes Merkmal darstellt und somit sich von der vorigen Pixelmenge unterscheidet. Ein Bild lässt sich in Patches zerlegen, die verschiedene Objekte im Bild darstellen. Ein Patch ist ein kleiner Bereich eines Bildes. Bilder haben in der Regel drei Dimensionen, die Höhe, Breite und Farbtiefe. Mit einem Batch, kann eine Anzahl an Datenexemplare angegeben werden, die beim Training in einer Iteration verarbeitet werden. So kommt bei der Darstellung eine weitere Dimension hinzu. Ein Bild wird als ein vier dimensionaler Tensor – $[batch, farbtiefe, hoehe, breite]$ - aufgefasst. Farbbilder haben eine Farbtiefe von

drei [François (2017), Kapitel 2].

Bausteine

Für die Erkennung der Merkmale ist die Operation der Convolution, also die Faltungsoperation bedeutsam und macht einen CNN aus. Für die Klassifikation werden anschließend sogenannte vollständig verknüpfte Schichten verwendet. Diese Arbeit verwendet die Begriffe Convolutional-Schicht und Convolution. Mit „Convolution" ist die Faltungsoperation gemeint und eine „Convolutional-Schicht", ist eine Schicht, in der die Faltungsoperation durchgeführt wird [Goodfellow et al. (2016), Kapitel 9].

Die Convolutional-Schicht

Ein Filter ist eine Matrix , die von links nach rechts über das Bild verschoben wird, um die Convolution auf einem Patch der gleichen Größe auszuführen. Die Höhe und Breite des Filters ist dabei gleich. Die Werte des Filters wird elementweise mit den Werten des Patches multipliziert und addiert. Mathematisch wird der Skalarprodukt zweier Matritzen berechnet. Das Ergebnis ist eine Matrix, die im englischen Raum als *feature map* bezeichnet wird. In dieser Arbeit wird die Bezeichnung *Merkmalskarte* genutzt. Wichtig ist, dass Filter die gleiche Farbtiefe haben, wie die Eingabe, da sonst die Multiplikation nicht möglich ist. Beim Ablauf der Convolution werden zwei weitere Anweisungen gesetzt. Der *Stride* gibt an, in welchem Intervall der Filter über die Eingabe von links nach rechts gleitet. An den Ränder der Eingabe erfolgt durch die Angabe des *Paddings* eine Auffüllung rund um die Eingabe, um bei der Berechnung die Pixel am Rand nicht außer Acht zu lassen. Eine Convolutional-Schicht wird durch die genannten Komponenten definiert. Meistens werden verschiedene Filter verwendet, weshalb die Anzahl an Merkmalskarten durch die Anzahl der Filter bestimmt wird [Goodfellow et al. (2016), Kapitel 9].
Ein CNN besteht oft aus mehreren Schichten. Einer Eingabeschicht, mehreren Convolutional-Schichten, gefolgt von Aggregationsschichten und einer vollstän-

dig verknüpften Schicht. Aggregationsschichten werden eingefügt, um die Komplexität des Netzes zu reduzieren oder zu Regulieren. Zu diesen Schichten zählen Pooling und Aktivierungsfunktionen.

Pooling

Die Pooling Schicht verwendet eine Matix, die unter Angabe eines Strides von links nach rechts über die Merkmalskarte gleitet und eine Funktion anwendet. Beim Max-Pooling wird im Rahmen der Matrix das Maximum gewählt [Nikhil Ketkar (2021), Kapitel 6].

Aktivierungsfunktionen

Rectified Linear Units - kurz ReLU – ist eine nicht lineare Funktion, die durch ReLU$(x) = max(0, x)$ definiert wird. Die Funktion gibt somit für positive Zahlen, die Zahl selbst zurück, sonst wird null ausgegeben [François (2017), Kapitel 2].

Die Softmax Funktion wird auf die Ausgabe einer vollständig verknüpften Schicht angewendet, um die Wahrscheinlichkeitsverteilung der verschiedenen Klassen darzustellen. Nachdem ein Datenexemplar durch die Schichten der Merkmalsextraktion voran gegangen ist und bevor es die letzte Schicht erreicht, ist dieser ein Tensor der Form $[x_1, ..., x_k]$, bei k verschieden Klassen. Dieser Tensor ist die Eingabe für die Softmax Funktion. Für jeden Werteintrag des Tensor - jedes x_i - wird folgende Berechnung durchgeführt:

$$Softmax(x_i) = \frac{\exp x_i}{\exp x_1 + \cdots + \exp x_k} \tag{2.1}$$

[Contributors (2019)]

Auf diese Weise wird der Tensor gebildet, der Werte zwischen null und eins beinhaltet und summiert eins ergeben. Diese einzelne Werte geben die Wahrscheinlichkeit dafür an, dass das Datenexemplar zur Klasse i gehört.

Klassifizierer

Die vollständig verknüpfte Schicht verbindet die Neuronen in einer Schicht mit Neuronen in einer anderen Schicht. Diese Schicht wird verwendet, um die Datenexemplare in verschiedene Klassen zu gruppieren. Hier fängt die Phase des Klassifizieres an. Die Vorhersage, die für das Datenexemplar getroffen wird, ist die Klasse mit der dem höchsten Wahrscheinlichkeitswert [François (2017), Kapitel 5].

Die Verlustfunktion Negativ Log-Likelihood (NLL)

Diese Funktion wird oft auf das Ergebnis des Klassifizierers angewendet. Ein hoher Wahrscheinlichkeitswert ergibt nach der Anwendung von NLL niedrige Werte. Da die Wahrscheinlichkeit dafür, dass das Datenexemplar zur Klasse i gehört hoch ist, folgt, dass die Verlustfunktion in diesem Fall niedrig ist. Die Folgerung ist, dass das Modell in seiner Entscheidung der Vorhersage für das Datenexemplar sehr wahrscheinlich „richtig“ liegt. Niedrige Wahrscheinlichkeitswerte, ergeben nach NLL hohe Werte. Deshalb wird beim Training durch Backpropagation die Minimierung der Verlustfunktion gesucht [Miranda (2017)].

2.3.2 AlexNet Architektur

Die AlexNet Architektur ist ein Ergebnis des Wettbewerbs ILSVRC-2012. Der Wettbewerb „ImageNet Large Scale Visual Recognition Challenge (ILSVRC)“ wurde im Jahr 2010 gestartet, bei dem es sich um die Klassifikation von 1,2 Millionen Testbilder mit Labels in 1000 verschiedene Klassen handelte. Ziel war es die niedrigste Top-1- und Top-5-Fehlerrate unter den Wettbewerber zu erzielen. Die Top-5-Fehlerrate ist der Prozentsatz der Testbilder, bei denen die richtigen Labels nicht zu den fünf wahrscheinlichsten des Modells gehören. Im Jahr 2012 wurde das Paper „ImageNet Classification with Deep Convolutional Neural Networks“ von Alex Krizhevsky et al. veröffentlicht und gewann den Wettbewerb (ILSVRC-2012), mit der Anwendung eines CNN, das eine Top-5-Fehlerrate von 15,3% erreichte. Das Netzwerk ist als AlexNet bekannt [Alex Krizhevsky (2012)].

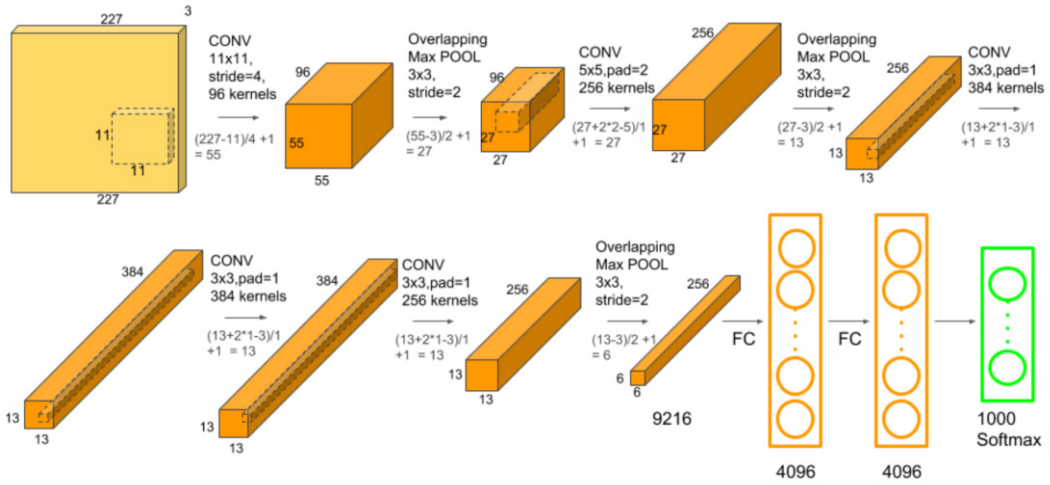


Abbildung 1: Die Abbildung stellt die AlexNet Architektur dar [*AlexNet Architecture* (o. J.)].

2.4 Clustering

Clustering ist eine der bekanntesten Anwendung des unüberwachten Lernens. Bei der Clusterbildung werden die Datenexemplare in verschiedenen Gruppen unterteilt. Dabei werden Datenexemplare mit ähnlichen Merkmalen oder Muster zusammen gruppiert, während Datenexemplare die sich von einander unterscheiden in verschiedenen Gruppen verteilt werden. In jeder Gruppe auch Cluster genannt, sollten wir somit Datenexemplare finden, die ein gemeinsames Merkmal aufweisen und sich von Merkmalen in anderen Cluster differenzieren. Es gibt verschiedene Clusteralgorithmen, wie zum Beispiel K-means [Patel (2019), Kapitel 5].

2.4.1 K-Means

Der K-means Algorithmus verteilt die Datenexemplare in k unterschiedliche Cluster. Die Zahl k wird vom Anwender vergeben und steht für die Anzahl der Cluster. Anfangs wird zu jedem Cluster ein Zentroid zufällig ausgewählt. Dieser Zentroid ist ein Datenexemplar aus dem Trainingssatz und unterscheidet sich von den anderen k-1 Zentroiden, damit jeder Zentroid eines Clusters eindeutig ist. Danach wird

der euklidische Abstand von jedem Zentroid zu allen anderen Datenexemplare aus dem Trainingssatz berechnet, und dann zum nächst näheren Zentroid gruppiert, d.h. bei dem der euklidische Abstand am geringsten ist. Auf diese Art und Weise werden in der ersten Iteration die k-Cluster gebildet. Nachfolgend wird in jedem Cluster der aktuelle Zentroid verlegt. Die neue Stelle des Zentroids ist der Mittelwert der Datenpunkte im Cluster. Der Abstand der Datenpunkte zu den Zentroiden wird neu ermittelt, und die Datenpunkte wechseln zu einem anderen Cluster, falls der Abstand des Datenpunktes zum Zentroid geringer ist, als zu seinem aktuellen. Danach werden erneut die Positionen der Zentroiden berechnet, den Mittelwert der Datenpunkte im Cluster. Dieser Ablauf wiederholt sich, bis die Datenpunkte nicht mehr zu einem anderen Cluster wechseln, denn dies bedeutet, dass kein anderer Zentroid näher gelegen ist, als sein aktueller. Das Ergebnis von K-means ist von der Startvereteilung der Zentroide abhängig, deswegen gibt es verschiedene Methoden zur Wahl der anfänglichen Zentroide. Wie oben beschrieben, können Datenexemplare des Trainingssatzes benutzt werden oder beispielsweise kann die Position der Zentroide durch die Berechnung von bestimmen Eigenschaften initiiert werden. Auch die Auswahl der Zahl k, ist für das Ergebnis von K-means ausschlaggebend . Die Problemstellungen, die durch K-means gelöst werden, variieren. Wenn der Anwender Wissen über die Datenmenge verfügt, ist die Auswahl von k eindeutig. Ist die Wahl der Anzahl der Cluster nicht offensichtlich, können verschiedene Methoden herangezogen werden, wie das Ellenbogenkriterium [Patel (2019), Kapitel 5].

2.4.2 DeepCluster

In der wissenschaftlichen Arbeit „Deep Clustering for Unsupervised Learning of Visual Features“ von Facebook AI Research wird die Methode DeepCluster vorgestellt. Dabei geht es um einen Ansatz im Bereich des unüberwachten Lernens, das für das Trainieren von CNNs eingesetzt wird. Wie im Namen enthalten, ist dieser Ansatz unter den Clusteralgorithmen einzuordnen . Trotz der Erfolge bei der Anwendung von Clustering in der Bildklassifikation, beschreiben die Autoren den Umstand, dass Clustering Methoden grundsätzlich für lineare Modelle entwickelt

werden und beim gemeinsamen Erlernen von Merkmalen nicht funktionieren. Der DeepCluster Ansatz wirkt diesem Umstand entgegen, indem es das Ende-zu-Ende Training von CNNs berücksichtigt, d.h. die Parameter werden gemeinsam trainiert. Beim DeepCluster werden iterativ die erstellten Merkmale geclustert und anschließend werden die Cluster Zuordnungen als Pseudolabels benutzt, um die Parameter des CNNs zu aktualisieren. Das Vergeben von Pseudolabels, ist eine selbstüberwachte Methode des unüberwachten Lernens, um die Labels, die beim überwachten Lernen direkt vergeben werden, zu imitieren. Dabei steht dem Anwender frei, welcher Clusteringalgorithmus eingesetzt wird. Die Autoren haben in der Anwendung den Schwerpunkt auf K-means gelegt [Caron et al. (2018)].

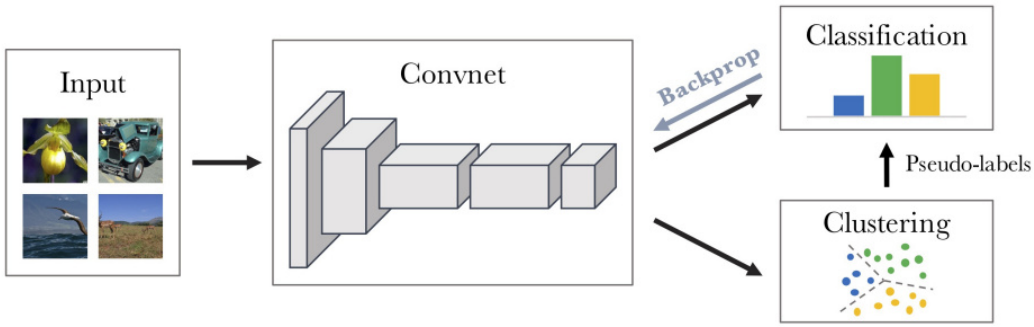


Abbildung 2: DeepCluster Ansatz [Caron et al. (2018)]

Die Eingabe von K-means ist die Menge der erzeugten Merkmale vom CNN und gruppiert diese Merkmale in k verschiedene Cluster. Anschließend lernt der Algorithmus gemeinsam eine Zentroidmatrix und die Clusterzuordnungen von jedem Datenexemplar. Das Lernen liefert eine Reihe von optimalen Zuordnungen und eine Zentroidmatrix. Die Zuordnungen werden im DeepCluster als Pseudolabels verwendet, allerdings wird von der Matrix kein Gebrauch gemacht. Zusammengefasst wechselt der DeepCluster zwischen den Cluster der Merkmale zur Erzeugung von Pseudolabels und der Aktualisierung der Parameter des CNNs durch die Vorhersage der erzeugten Pseudolabels [Caron et al. (2018)].

2.5 Erklärbare Künstliche Intelligenz

Die Erklärbarkeit von künstlicher Intelligenz soll die Transparenz von Modellen verbessern, damit in ihnen Vertrauen gewonnen werden kann . Vor allem im Deep Learning Bereich werden Modelle sehr schnell unübersichtlich, sodass nicht festgestellt werden kann, warum das Modell bestimmte Entscheidungen trifft. Auch wenn die Erfolge in der Bildklassifikation bei Anwendung von CNNs beachtlich sind, fehlen an Methoden für die Erklärbarkeit dieser Modelle. Aktuell gibt es verschiedene Methoden, die angewendet werden, um die Transparenz dieser Modelle zu fördern. Beispielsweise gibt es visuelle Lösungen bei der Erkennung von Merkmalen . Die erklärbare künstliche Intelligenz zielt dabei auf das Entwerfen und Entwickeln von Methoden, die dem Menschen helfen sollen das Verhalten, beziehungsweise die Entscheidungen eines Modells zu verstehen.
Die Arbeit von Ayyar et al. [Ayyar et al. (2021)] gibt einen Überblick von aktuellen White-Box Methoden, die bei der Bildklassifikation eingesetz werden, um die Entscheidungen von CNNs erklärbar zu machen. White-Box bedeutet, dass Methoden in dieser Kategorie, die Fähigkeit haben die interne Architektur und Parameter des Modells zu erkennen. Dabei unterteilen die Autoren bestehende White-Box Methoden in drei Taxonomien. Einmal in Methoden, die auf die Linearisierung des Deep Neural Networks basieren, die sich auf die Netzstruktur beziehen und Methoden, die auf den Adversarial-Ansatzes basieren. Die Erklärbarkeit in diesem Zusammenhang konzentriert sich auf die Zuordnung der Ausgabe zur Eingabe und beantwortet nicht die Frage, welche Komponenten des Modells zur Entscheidung beigetragen. Vielmehr wird die Beziehung zwischen den vorhergesagten Ausgangsklassen und Pixel des Bildes betrachtet [Ayyar et al. (2021)].
Diese Arbeit beschäftigt sich mit den Methoden Guided Backpropagation, Grad-CAM und die Kombination aus beidem Guided Grad-CAM, die in [Ayyar et al. (2021)] unteranderem in der Taxonomie der Methoden basierend auf Linearisierung des Modells aufgelistet werden. Die Analyse von Methoden in den anderen Taxonomien werden in dieser Arbeit nicht skizziert, da die Zeit im Rahmen der Bachelorarbeit bedingt ist.

2.5.1 Ausgangslage

Für die unterstehenden Methoden denken wir uns in die Ausgangslage, dass ein CNN jedes Bild aus einem Datensatz mit Anwendung von überwachtem Lernen in verschiedenen Klassen aufteilt. Wie im vorrigen Kapitel geschildert wird im Klassifizierer ein Tensor mit Wahrscheinlichkeitswerten generiert. Das Problem in der Erklärbarkeit liegt in der Zuweisung von jedem Pixel mit einem Relevanzwert in Bezug auf seinen Beitrag zur Ausgabe. Eine Relevanzkarte mit entsprechenden Relevanzwerten in Bezug auf die Ausgabe für Merkmale in den Convolutional-Schichten wird in den Visualisierungsmethoden ermittelt. Um dem Benutzer den Einfluss der Pixel darzustellen, wird in der Regel eine Visualisierung der Bewertung der Relevanzkarte vorgenommen, indem sogenannte „heat maps“ ermittelt werden und anschließend wird diese auf das Originalbild überlagert. Genauer ist eine Heat Map die Visualisierung der Relevanzkarte mit farbigen Look-up-Tables (LuTs), die eine Farbskala zwischen $[0, 1]$, von blau bis rot abbilden. Blau liegt bei null an und rot bei eins. Liegt ein Wert nah an der null, bekommt dieser eine bläuliche bis violete Farbe. Werte um 0.5 haben eher eine türkise Farbe und Werte nahe an der eins werden durch die Farben orange bis rot dargestellt. Die Visualisierung von Heat Maps ermöglicht somit dem Benutzer die Ergebnisse der Erklärungsmethoden zu verstehen, um Erkenntnisse und Schlüsse der Klassifikation zu gewinnen. Die Farbwerte geben dem Benutzer eine visuelle Antwort darauf, welche Pixelwerte relevant für die Einteilung in die Klasse sind [Ayyar et al. (2021)].

2.5.2 Methoden

Ausgehend der oben beschriebenen Ausgangslage (Klassifizierer und Problemstellung) wurden verschiedene Methoden entwickelt, die für die Visualisierung von Relevanzkarten für ein Bild verwendet werden können. Die folgenden Methoden basieren auf die Linearisierung von CNNs, da ein CNN als ein nicht linearer Klassifizierer aufgefasst werden kann. Diese Ansätze liefern Erklärungen, die die nichtlineare Abbildung approximieren. Die verschiedenen Methoden unterscheiden sich in der approximativen Berechnung der Parameter, weshalb unterschiedliche Erklä-

rungen geliefert werden. Guided Backprogation und Grad-CAM haben gemeinsam, dass sie auf die Berechnung der Gradienten im Modell zurückgreifen, um die lineare Approximation des Modells durchzuführen. Sie unterscheiden sich jedoch bei der Wahl der Schichten, in denen die Gradienten berechnet werden. Während Guided Backpropagation auf die Eingabeschicht zugreift, werden in der Grad-CAM Methode die Gradienten in der letzten Convolutional-Schicht berechnet [Ayyar et al. (2021)].

Gradient Backpropagation

Die Gradient Backpropagation Methode erklärt die Vorhersage eines Modells auf der Grundlage seines lokal ausgewerteten Gradienten. Ausgehend vom lokalen Gradient des Klassifikationsergebnis S_c bezogen auf die Eingabe x, bei einem Bild x_0 wird zur Berechnung des Gewichtungsparameters w aus der Approximationsgleichung verwendet. Die lineare Approximation des CNNs - der nicht linearen Abbildung $f(x)$ – wird als Taylor-Expansion erster Ordnung von $f(x)$ der Nähe eines bestimmten Bildes x_0 formuliert und $b = f(x)$. Die Gewichtungsparameter werden nach folgenden Schema ermittelt. Die partielle Ableitung des Klassifikationsergebnisses nach der Eingabe entspricht der Berechnung des Gradienten für einen einzigen Backpropagation Durchgang für das Eingabebild x. Diese partielle Ableitung übereinstimmt mit dem Backpropagation Schritt, der während des Trainings des CNNs vollzogen wird, das in den Meisten Fällen einem Batch von Bildern entspricht. Mit der Kennzeichnung von x_0 soll deshalb kenntlich gemacht werden, dass die Backpropagation nur für ein Bild aus dem Batch gilt. In dieser Methode wird die Backpropagation bis zur Eingabeschicht durchgeführt, um die Pixel zu ermitteln, die den größten Einfluss auf die Ausgabe haben. Die endgültige Heatmap der Relevanzwerte R_{ij} für ein Pixel x_{ij} werden im Falle eines RGB-Bildes, als das maximale Gewicht dieses Pixels aus den Gewichtungsmatrizen jedes der drei Channels berechnet [Ayyar et al. (2021)].

Guided Backpropagation

In der Arbeit von Springenberg et al. [Springenberg (2015)] wird eine neue Variante des DeconvNet Ansatzes vorgeschlagen, das auf eine größere Menge von Netzwerken angewendet werden kann, als zu dieser Zeit (2015) bereits bestehenede Arbeiten. Die Autoren gehen bei den Experimenten von einem Allgemeinen Aufbau von CNNs aus, die gleichen Prinzipien folgen, wie im Kapitel *Convolutional Neural Networks* beschrieben. CNNs unterscheiden sich demnach häufig in der Anzahl der der verwendeten Convolution-Schichten und der Verwendung von den Aggregationsschichten, wie beispielsweise Maxpooling und Dropout. Die neue Variante des DeconvNet Ansatzes - Guided Backpropagation - ist auf eine Größere Anzahl von CNNs anwendbar.

Im Bezug auf die DeconvNet-Methode stellen die Autoren dar, dass die Verwendung der Switches aus einem Forward Pass, das DeconvNet und seine zugehörige Wiederherstellung auf ein Bild bedingt und nicht direkt die gelernten Merkmale visualisiert. Da die Architektur des einfachen CNNs keine Maxpooling Schichten integriert, bedeutet es, dass ohne Switches nicht „deconvolve“ werden kann. Die DeconvNet Methode liefert für höhere Schichten des einfachen CNN keine gute Visualisierung, da die Strukturen im Bild nicht klar erkennbar werden. Der Schluss daraus ist die Bestätigung, dass niedrigere Schichten allgemeinere Strukturen erlernen, weshalb die Wiederherstellung eines Musters möglich wird, das sie aktiviert. Die Merkmale in höheren Schichten werden genauer und es gibt kein Bild, das diese Neuronen maximal aktiviert. Die Autoren schlagen eine Modifikation des DeconvNet vor, um eine genauere Wiederherstellung in allen und insbesondere in höheren Schichten zu ermöglichen. Eine Methode, die den Teil eines Bildes visualisiert, der ein bestimmtes Neuron am stärksten aktiviert, liegt in der Berechnnung des Gradienten der Aktivierung in Bezug auf das Bild. Als „Deconvolution“ wird ein Backward Pass durch das Netzwerk durchgeführt, mit der Ausnahme, dass beim Propagieren durch eine Nichtlinearität, der Gradient ausschließlich auf der Grundlage des oberen Gradientensignals berechnet wird und der untere Eingang ignoriert wird. Ist die Nichtlinearität durch die ReLU Funktion gegeben,

dann werden bestimmte Einträge auf der Grundlage der oberen steigung auf Null gesetzt. Anstatt Werte auszublenden, die negativen Einträge des oberen Gradienten („Deconvnet“) oder der unteren Daten (Backpropagation) entsprechen, werden die Werte ausgeblendet, bei denen mindestens einer dieser Werte negativ ist. Diese Kombination wird als Guided Backpropagation bezeichnet, da zur normalen Backpropagation ein zusätzliches Führungssignal aus den höheren Schichten hizugefügt wird. Dadurch wird der Rückfluss von negativen Gradienten verhindert, die den Neuronen entsprechen, die die Aktivierung der höheren Schicht – dessen Visualisierung gesucht wird – verrigern. Guided Backpropagation funktioniert im Vergleich zu DeconvNet gut und die Verwendung von Switches ist nicht mehr notwendig. Die Visualisierung von höheren Schichten wird mit dieser Methode ermöglicht. Bei der Berechnung von Heatmaps auf der Grundlage von Gradienten verschafft die Antwort auf die Frage, welche Pixel zur Vorhersage des Netzes beitragen. Die von Springenberg et al. vorgeschlagenen Lösung verhindert die Rückregression negativer Gradienten, die beim „DeconvNet“ auftreten, da sie die Aktivierung der höheren Schicht – die visualisiert werden soll – verringern. Durch die Kombination der vom DeconvNet durchgeführten Rectification und der Gradient Backpropagation wird dies ermöglicht. Guided Backpropagation schlägt somit vor, den Fluss negativer Gradienten zu beschränken, die während der Backpropagation einen negativen Wert haben, und auch die Werte, die während des Forward Pass (Vorwärtsdurchgangs) einen negativen Wert hatten. Die Aufhebung negativer Gradientenwerte wird als „guidance“ bezeichnet.

Mit dieser Anwendung haben die Autoren anhand von den Experimenten die Erkenntnis, dass die Visualisierungen schärfer sind für die deskriptiven Regionen im Eingangsbild. Es ist zu erkennen, dass die Guidance-Schritte zu einer Verringerung der Anzahl von Pixeln führen, die einen höheren Wichtigkeitswert haben, und somit eine etwas schärfere Visualisierung erzeugen [Springenberg (2015)].

Grad-CAM

Die Autoren von Grad-CAM [Selvaraju & et al. (2017)] schlagen Ende 2019 eine Methode vor, die Gradienten einer Zielkategorie verwendet, die in die letzte Convolution-Schicht einfließen, um eine grobe Lokalisierungskarte zu erstellen. Die Lokalisierungskarte führt zu Hervorhebungen der Pixel, die zur Vorhersage geführt haben. Bei der Anwendung von Grad-CAM ist kein erneutes Traning nötig und kann auf eine Vielzahl von CNN Architekturen durchgeführt werden, dabei werden keine Änderungen an der Architektur vorgenommen [Selvaraju & et al. (2017)]. Die Grad-CAM Methode ist eine Verallgemeinerung der CAM Methode von Zhou et al. [Zhou & et al. (2015)]. Eine gute visuelle Erklärung des Modell zur Rechtfertigung einer beliebigen Zielkategorie sollte klassendiskriminierend – die Kategorie im Bild lokalisieren – und hochauflösend – feinkörnige Details erfassen – sein. Die autoren stellen fest, dass die Methoden Guided Backpropagation und DeconvNet die Eigenschaft haben hochauflösend zu sein, jedoch nicht klassendiskriminierend. Ansätze, die mit Lokalisierungskarten arbeiten, wie CAM oder Grad-CAM erfüllen diese Eigenschaft. Die CAM Methode von Zhou et al. nimmt Änderungen an der CNN Architektur, indem vollständig verbundene Schichten durch Convolution-Schichten und globales Avarage-Pooling ersetzt werden. So entstehen klassenspezifische Informationen des Bildes. Die Grad-CAM Methode greift auf die Informationen der Gradienten, die in der letzten Convolution-Schicht einfließen, da bekannt ist, dass die Merkmalskarten aus tieferen Convolution-Schichten genauere Muster erfassen und somit klassenspezifischer sind.
Zur Ermittlung der Lokalisierungkarte in der Grad-CAM Methode wird zuerst die Berechnung des Gradienten der Punktzahl für die Zielkategorie in Bezug auf Aktivierungen von Merkmalskarten einer Convolution-Schicht durchgeführt.
Berechnung des Gradients der Punktzahl y^c für die Klasse c in Bezug auf Aktivierungen von Merkmalskarten A^k einer Convolution-Schicht:
Diese zurückfließenden Gradienten werden global über die Breiten-und Höhendimensionen - indiziert durch i und j – gepoolt, um die Gewichte der Neuronen zu erhalten:

Bei der Berechnung von α_k^c, während der Backpropagation von Gradienten, basiert die Berechnung auf aufeinanderfolende Matrixprodukte der Gewichtsmatrizen und des Gradienten in Bezug auf die Aktivierungsfunktionen bis zur letzten Convolution-Schicht, auf die die Gradienten propagiert werden. Das Gewicht α_k^c stellt somit eine Linearisierung des tiefen Netzes nach A dar und gibt die Wichtigkeit der Merkmalskarte k für eine Klasse c. Anschließend wird eine gewichtete Kombination von forward activation maps und ReLU ausgeführt. ReLU wird angewendet, da nur die Merkmale – die einen positiven Einfluss auf die gewähle Klasse haben - von Interesse sind. Die positiven Pixel, die y^c maximieren, sollen hervorgehoben werden. Im Gegensatz haben negative Pixel eher einen niedriegen Einfluss auf die Klasse c, diese Pixel sind eher irrelevant für die Klasse c und gehören wahrscheinlich zu anderen Klassen.

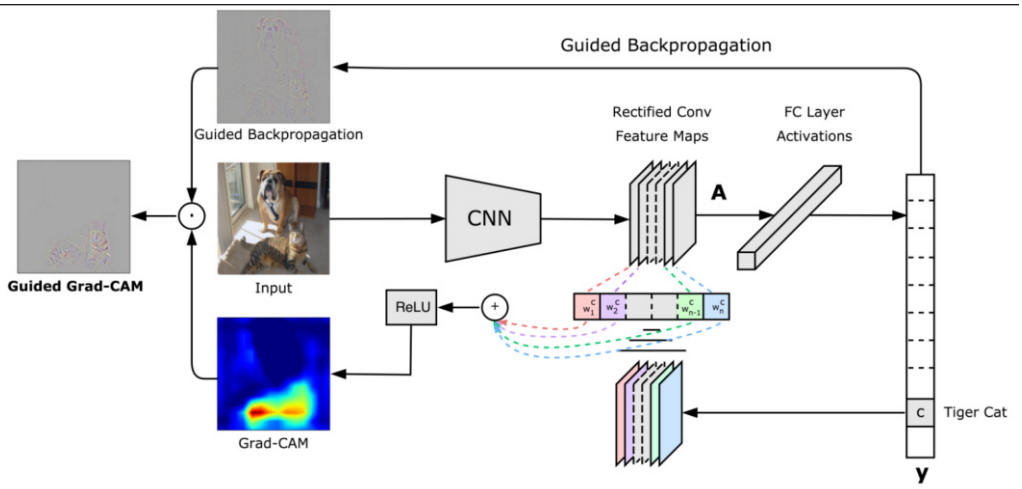


Abbildung 3: Überlick der Grad-CAM Methode [*Grad-CAM* (o. J.)]

Als Eingabe ist ein Bild und die zum Bild gewüschte Zielklasse gegeben, dabei betrachten wir nur die Aufgabe zur Bildklassifikation. Das Bild wird von einem CNN verarbeitet bis es zur Klassifikation im CNN propagiert wird, aber noch vor Anwendung der Softmax Funktion, um eine Bewertung für die Zielklasse zu erhalten. Bis auf die gegebene Zielklasse, werden alle Gradienten der anderen Klassen auf null gesetzt, die der Zielklasse auf eins. Nur die Gradienten der Zielklasse werden auf eins gesetzt, die der anderen Klassen auf null . Dieses Signal wird auf die Featuremaps von Interesse – nach dem ReLU auf die Featuremaps angewendet wurde –

„backpropagated“, um die grobe Grad-CAM Lokalisierung zu erhalten. Die erstellte Heatmap stellt die relevanten Pixel, die für die Zugehörigkeit der Klasse sprechen. Die rötlichen Stellen in der Heatmap zeigen die relevanten Pixel des Bildes für die Zielklasse während die bläulichen Stellen, darauf hinweisen, dass diese Pixel irrelevant sind. Wird die erstellte Heatmap punktweise mit Guided Backpropagation multipliziert, ergibt sich Guided Grad-CAM, das kurz im nächsten Abschnitt zusammengefasst wird.

Guided Grad-CAM

Die Grad-CAM Methode ist klassendiskriminierend, jedoch ist die erstellte Heatmap grob. Die Featuremap der letzten Convolution-Schicht hat im Vergleich zum Eingabebild eine geringere Auflösung , weshalb die Featuremaps keine feinkörnige Details auffassen, wie die Methode Guided Backpropagation. Deshalb haben die Autoren von [Selvaraju & et al. (2017)] beide Methoden kombiniert, indem die Heatmap punktweise mit Guided Backpropagation multipliziert wird. Diese Anwendung wird Guided Grad-CAM genannt und hat gezeigt, dass wichtige Regionen des Eingabebildes, die relevant für die Entscheidung sind, mit einer höheren Auflösung visualisiert werden. Beispielsweise haben die Autoren festgestellt, das Guided Grad-CAM für die Zielklasse „Tiegerkatze“ nicht nur die Regionen der Katze hervorhebt, sondern auch die Streifen der Katze. Diese Stellen sind für die Zielklasse „Tiegerkatze“ relevant [Selvaraju & et al. (2017)].

3 Konzept

Das Konzept baut auf den vorgestellten theoretischen Grundlagen. In dieser Arbeit wird mit dem DeepCluster ein AlexNet trainiert, um eine Clusterbildung aus den Eingabedaten zu generieren. Auf das Ergebnis werden die Methoden Grad-CAM und Guided Grad-CAM angewendet, damit eine manuelle Untersuchung auf die Erklärbarkeit des unüberwachten Trainings auf den Datensatz erfolgt.

3.1 Wahl des Datensatzes und Hintergrund

Der zu analysierende Datensatz hat seinen Ursprung aus dem Projekt „Classification and Representation for Networks“ – ClaReNet. Das Projekt kommt zustande durch die Kooperation der Römisch-Germanischen Kommission mit dem Big Data Lab Team der Goethe Universität Frankfurt am Main. Das Projekt stellt für die Analyse drei unterschiedliche keltische Münzserien bereit. Jede dieser Münzserien weist unterschiedliche Problemstellungen in deren Klassifikation vor. Dadurch entstehen Forschungsfragen zu Klassifikationsverfahren. Es werden Analysen durchgeführt und Möglichkeiten getestet über die Klassifikation oder Darstellung der Münzserien.
Basierend aus diesem Kontext, werden Klassifikations-und Darstellungsmethoden im Deep Learning Umfeld verwendet [für Bildung und Forschung (2021)]. Diese Arbeit beschäftigt sich mit der zweiten Münzserie - „Monnaies á la croix“ - des ClaReNet Projekts. Die meisten Münzen der zweiten Münzserie mit fast 2000 verschiedenen Münzen, stammen aus großen Geldschätzen, die im 19. Jahrhundert entdeckt wurden [Hiriart (2017), Seite 13-14]. Das Besondere an diesem Münzsatz ist, dass auf der Rückseite ein Kreuz abgebildet wird, sodass dadurch automatisch eine Unterteilung mit vier Feldern entsteht. In Diesen Feldern werden wiederum Symbole abgebildet oder sind leer. Da die Münzen eine große typologische Vielfalt

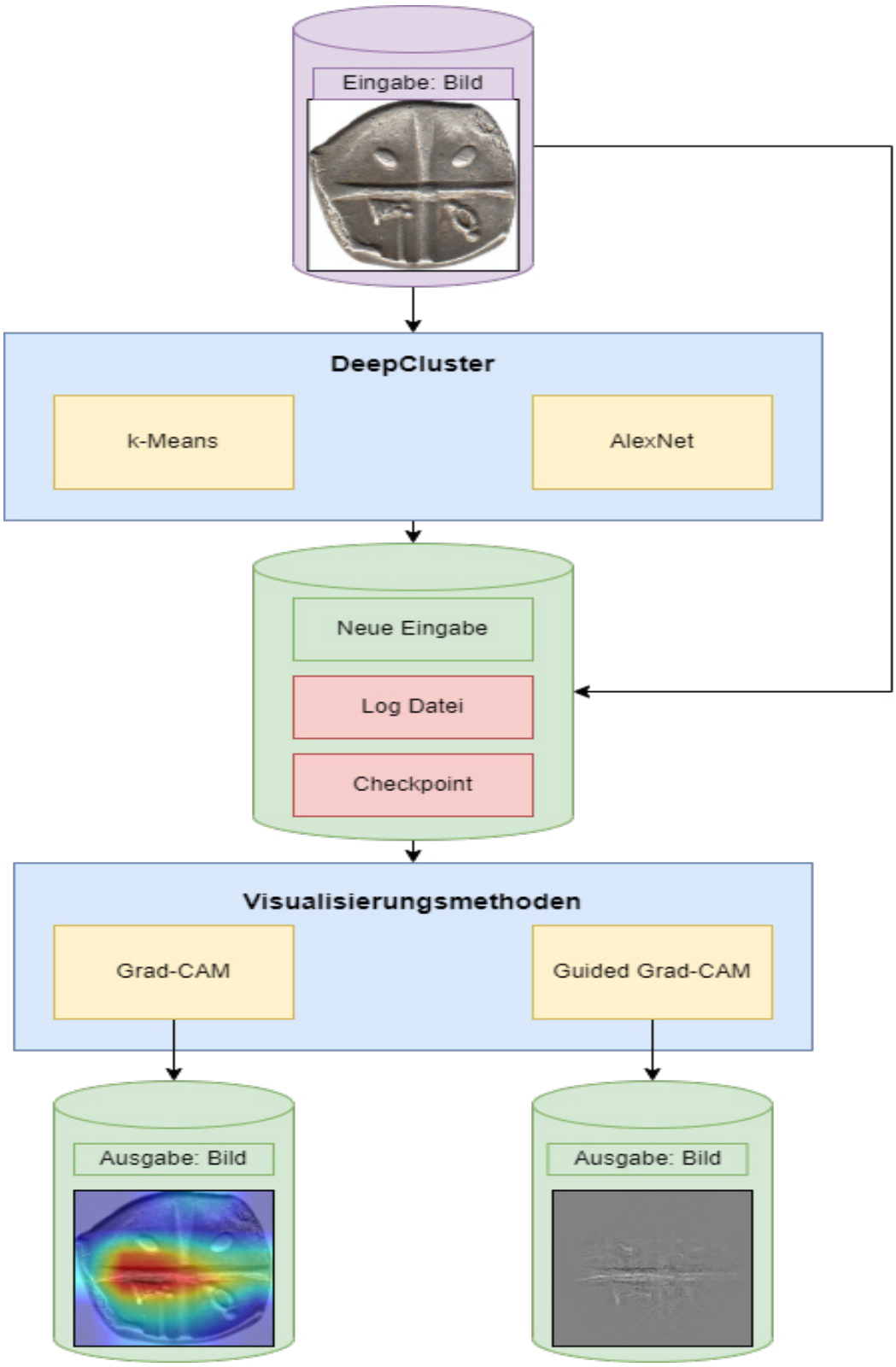


Abbildung 4: Ablauf des selbst entwickeltes Konzepts

aufweisen, entstehen Schwierigkeiten und Herausforderungen in der Numismatik bei ihrer Analyse und Zuordnung [Hiriart (2017), Preface]. Zu den Münzen gibt es keine *richtige* Labels. Die erwarteten Ergebnisse sind aus diesem Grund unbekannt.

(a) MNRB-206 (b) Ninv21

Abbildung 5: Links: positives Beispiel einer Kreuzmünze. Alle Merkmale sind erkennbar; Rechts: negatives Beispiel, da der Zustand der Münze im Vergleich schlecht ist.

3.2 AlexNet als CNN

Die Wahl der AlexNet Architektur liegt an zwei Argumenten. Zum einen wird in den Grundlagen erwähnt, dass im Bereich der Bildklassifikation gute Ergebnisse erzielt werden. Zum anderen wird in der DeepCluster Implementierung AlexNet bereitgestellt.

3.3 DeepCluster mit K-means als Clusteralgorithmus

Der Datensatz, die CNN-Architektur und die Parametereingabe für k-means erlauben das Ausführen des DeepClusters. Ein Augenmerk muss auf die Wahl von k gelegt werden. In den Versuchen liegt diese zwischen 50 und 100. Ein zu großes k führt dazu, dass nur eine kleine Menge von Bildern in einem Cluster zugeteilt werden. Wiederum wird bei einem zu kleinem k erwartet, dass die Dichte an Bildern in einem Cluster zu hoch ist. Eine nicht zutreffende Wahl von k führt zu einer erschwerten Analyse der Merkmalserkennung, da diese manuell erfolgt. Da die Anzahl der Labels nicht bekannt ist, kann die Wahl von k nicht daran ausgemacht werden. Caron et al. haben bei Experimenten festgestellt, dass ein Maß an Übersegmentierung vorteilhaft ist [Caron et al. (2018)]. In diesem Fall kann nicht ermittelt werden, ob

ein k zwischen 50 und 100 eine Übersegmentierung im Bezug auf den Datensatz bildet. Da die Münzen verschiedene Merkmale aufweisen, ist davon auszugehen.
In den Versuchs Durchläufen wird eine Epochen Zahl zwischen 300 und 400 gewählt. Zum einen wurde die Wahl der Obergrenze aus Laufzeit Gründen ausgewählt. Die Entscheidung ist bestärkt durch die Wahl der Autoren des DeepClusters. Sie entschieden sich für eine maximale Grenze von 500. Eine zu kleine Wahl der Zahl kann dazu führen, dass das Modell möglicherweise nicht gut trainiert wird. Die Auswahl der Parameter spielt somit eine wichtige Rolle bei der Lösung der Klassifikationsaufgabe. Die Epochen Anzahl ist bei einigen Versuchen gleich, damit die Wahl von k in den Versuchen besser eingeschätzt und verglichen werden kann.

3.4 Grad-CAM und Guided Grad-CAM als Visualisierungsmethoden

Der DeepCluster erschafft uns eine Ausgangslage, um eine strukturierte Analyse durchführen zu können. Die Visualisierungsmethoden Grad-CAM und Guided Grad-CAM benötigen die Trainingsinformationen aus dem Training des AlexNet. Mit dem Grad-CAM Verfahren erhalten wir die Information, welche Pixel eines Bildes im Fokus liegen, um genau diese Clusterbildung zu erzeugen. Mit Guided Grad-CAM prüfen wir ob, relevante Regionen in einem Bild eine höhere Auflösung vorweist. Diese bildliche Darstellung soll uns dabei helfen, Muster zu erkennen, die möglicherweise unsere Forschungsfragen beantwortet.

3.5 Versuchsplanung

Dieses Kapitel soll einen Überblick beschaffen, welche Parameter bei den sieben Versuchen eingesetzt werden.

Reverse Modell

Für das Reverse Modell liegen 1864 Bilder der Rückseite vor. Der Datensatz wird nicht vorverarbeitet und es werden keine Bilder aussortiert.
Die vorliegende Tabelle zeigt die durchgeführten Versuche, mit verschiedenen Pa-

Versuchs ID	Anzahl der Cluster k	Epochen
R1	55	350
R2	65	350
R3	75	325
R4	75	400
R5	85	300
R6	85	350
R7	95	350

Abbildung 6: Geplante Versuchsdurchführung für die Behauptungen

rametereingaben für die Cluster-und-Epochenanzahl. Alle anderen Parameter wurden für die finalen Testläufe nicht modifiziert.

4 Umsetzung

Anhand des Konzepts ist es möglich eine Umsetzung zu realisieren. Wie die Realisierung stattfand, wird in diesem Kapitel beschrieben.
Dafür wird zunächst eine Einführung vorgestellt, welche die notwendigen Voraussetzungen für eine Implementierung darstellen soll, weil solch ein Konzept nicht ohne weiteres realisierbar ist. Darauffolgend wird auf die DeepCluster Implementation eingegangen, welches aus unserem rohen Datensatz dann neue Eingaben zur Verfügung stellt. Zuletzt wird die Implementation der Visualisierungsmethoden vorgestellt, welche nur genutzt werden, kann, durch die neuen Eingaben aus dem vorherigen Kapitel. Mithilfe der Visualisierungsmethoden ist es dann möglich, die gezielten Fragestellungen aus Kapitel 3 zu beantworten.

4.1 Voraussetzungen für die Durchführung

Das Aufbauen einer passenden Umgebung mit den Werkzeugen ist in der Softwareentwicklung eine Herausforderung. Durch den schnellen technologischen Wandel in der IT-Branche, sind heutzutage viele Methoden möglich, um eine gesamte Umgebung/Instanz aufzusetzen. Realisierbar ist es beispielsweise durch:

1. Lokaler Rechner
2. Externer Server via SSH oder Remoteverbindung
3. Cloud Umgebung

In diesem Unterkapitel wird die Herangehensweise von dieser Problemstellung skizziert. Des Weiteren kann es dazu dienen, anderen Lesern zu helfen, weitere ähnliche Konzepte zu implementieren, da das Aufsetzen einer Server- bzw. Entwicklungsumgebung eine Herausforderung für manche sein kann.

4.1.1 Umgebung

Anfangs startete die Implementierung auf meinem privaten Windows Rechner. Nach einer gewissen Zeit kristallisierte sich heraus, dass die vorhandenen Ressourcen meines Rechners nicht ausreichend sind, um das vorgestellte Konzept umzusetzen. Für eine erfolgreiche Umsetzung des Codes sind folgende Mindestanforderungen notwendig:

GPU

Eine GPU ist für diese Bachelorarbeit sinnvoll und angebracht, da normalerweise Sourcecode auf einer CPU kompiliert wird. Da die CPU jedoch zum einen von vielen Applikationen auf einem Rechner verwendet wird und zum anderen auch eine begrenzte Kapazität besitzt, ist es durchaus sinnvoll, den gesamten Arbeitsprozess auf eine GPU auszulagern. Des Weiteren erlaubt eine GPU, mathematische Berechnungen schneller durchzuführen.

Unix Betriebssystem

Jedes Betriebssystem ist anders aufgebaut und hat seine Vor-und-Nachteile. In der reinen Softwareentwicklung wird beispielsweise sehr häufig eine Unix Distribution ausgewählt, da hier mehr Spielraum ermöglicht wird beim generellen Aufsetzen der Serverinstanz. Auch im Rahmen dieser Bachelorarbeit ist es wichtig, da manche Bibliotheken nur auf Linux genutzt werden können.

Basierend aus verschiedenen Quellen, entschieden wir uns dann für Google, da diese stark auf dem Markt etabliert sind. Spezifisch für die Bachelorarbeit wurden zwei ausgewählte Anwendungen benutzt, welche im Folgenden genannt und erklärt werden:

Google Colab

Google Colab stellt für einen Benutzer eine Jupyter Notebook (web-basierte interaktive) Umgebung/Instanz, mit minimalen Installationen, bereit. Somit muss

der Benutzer sich nicht um grundlegende Installationen kümmern und kann sich direkt mit seinen notwendigen Installationen beschäftigen.
Der Anwender kann ohne weitere Installationen nativen Python Code oder Bash Befehle ausführen. Des Weiteren, ist ein Jupyter Notebook dafür bekannt, durch Markdown Funktionalität, Texte zu formatieren, sodass das gesamte Notebook für einen Leser sehr strukturiert dargestellt werden kann. Somit wurde von Google ein Produkt bereitgestellt, welches ohne großen Aufwand eine Cloud Umgebung zur Verfügung stellt. Außerdem läuft das Notebook auf einem Unix Betriebssystem, sodass die Bibliotheken, die für die Implementierung in dieser Arbeit gebraucht werden, installiert und verwendet werden können.

Google Drive

Die Google Drive ist ein Speicherort in der Cloud Umgebung. Somit ist ein Nutzer in der Lage Dateien zu speichern oder zu erstellen. Daraus folgt auch, dass der physische Speicher nicht beansprucht wird, da es von Google selbst gespeichert wird. Im Rahmen der Bachelorarbeit reserviert unser Datensatz viel Speicher, da es eine große Menge an Bildern sind.

Um das Konzept umsetzen zu können wird eine Verknüpfung beider Google Anwendungen aufgebaut. Für unseren Vorgang ist es notwendig, dass die Google Colab Instanz auf die Dateien in der Google Drive zugreifen kann. Des Weiteren muss sie auch in der Lage sein, die Ergebnisse, die wir erzeugen, in der Google Drive abzuspeichern. Das erlaubt uns das Wiederverwenden der erzeugten Ergebnisse und vereinfacht uns den Automatisierungsprozess.
Anfangs wurde die Standardversion von Google Colab genutzt, die für jeden Benutzer, welcher einen Google Zugang hat, nutzbar ist. Jedoch kristallisierte sich bei der Entwicklung heraus, dass die DeepCluster Implementierung sehr viele Ressourcen in Anspruch nimmt und für die Berechnung teilweise zu lang braucht. Nach einer Recherche wurde uns klar, dass Google Colab weitere Lizenzen anbietet, welche leistungsfähigere GPUs bereitstellt, jedoch nur nach dem Erwerb einer Lizenz.

Wir entschieden uns für den Erwerb dieser Lizenz und beobachteten eine deutlich schnellere Laufzeit.

4.1.2 Programmiersprache und Abhängigkeiten

Das Konzept beruht auf zwei Implementierungen, welche beide auf der Programmiersprache Python aufgebaut sind. Python unterstützt für den Data Science Bereich eine Vielzahl von bereitgestellten Bibliotheken/Module. Somit ist es für einen Nutzer nicht mehr notwendig, diese selbst zu implementieren, sondern kann sie auf einfacher Weise wiederverwenden, indem es durch den Python Paket Manager installiert wird.
Wie im Unterkapitel davor skizziert, wird normalerweise ein Source Code auf einer CPU ausgeführt. Um es auf eine GPU umlenken zu können, muss hierfür explizit eine Python Bibliothek installiert werden (CUDA). Somit werden alle mathematischen Berechnungen anstatt auf der CPU, nun auf der GPU laufen.
Spezifisch muss ein Augenmerk auch auf die Faiss Bibliothek gelegt werden, welche von der Facebook AI Research Gruppe entwickelt wurde. Diese ist eine wichtige Abhängigkeit für den DeepCluster Code, ansonsten können die gesamten Berechnungen nicht durchgeführt werden. In der Dokumentation von diesem Modul wurde explizit genannt, dass es nicht auf Windows installierbar ist, sondern nur auf Unix Systemen.

4.2 DeepCluster Implementation

Zunächst wird sich die Umsetzung des DeepClusters angeschaut. Hierfür wurde das öffentlich zugängliche Repository von Facebook Research genommen, welches für die Bachelorarbeit als ein Teil der Codebasis dient.
Da die Codebasis auf Python Version 2.7 beruht, musste zunächst der Quellcode für neuere Python Versionen angepasst werden. Das garantiert eine Einheitlichkeit zu den Visualisierungsmethoden. Somit müssen die erzeugten Eingabeparameter für die Visualisierungsmethoden nicht verändert werden auf Konventionen bezüglich der Programmiersprache. So wurde beispielsweise in der Hauptdatei *main.py*

```
target = target.cuda(async=True)
```

geändert zu:

```
target = target.cuda(non_blocking=True)
```

Der Grund hierfür ist, dass der Parameter *async* bei bei der CUDA Bibliothek veraltet ist, da höhere Python Versionen *async* in ihrer Implementierung reserviert haben. Eine andere Änderung war zum Beispiel der Zugriff auf den Checkpoint. So wurde die originale Implementierung von:

```
for key in checkpoint['state_dict']:
    if 'top_layer' in key:
        del checkpoint['state_dict'][key]
model.load_state_dict(checkpoint['state_dict'])
```

zu

```
state_dict = checkpoint['state_dict'].copy()
to_delete = []
for key in state_dict:
    if 'top_layer' in key:
        to_delete.append(key)
for key in to_delete:
    del state_dict[key]
model.load_state_dict(state_dict)
```

geändert.

Ziel des DeepClusters ist es, die notwendigen Trainingsdaten mit den Clusterbildungen zu erhalten. So wurde die Ausgabe der Log Datei erweitert. Sie sieht nun folgendermaßen aus:

```
cluster_log.log((deepcluster.images_lists, train_dataset.
   imgs, args.epochs))
```

Davor wurden nur die Clusterbildungen ausgegeben. Nun folgen auch die Bilder und die eingegebene End Epoche (aus Automatisierungs Gründen). Diese Informationen werden zusätzliche für die Visualisierungsmethoden benötigt, neben dem Checkpoint. In der Google Drive wird automatisiert ein *exp* Ordner erstellt mit den Checkpoint und der Cluster Log Datei.

Für das Testen und Ausführen der Implementation, wird ein Notebook bereitgestellt, welches sequenziell ausführbar ist. So wird am Anfang zum einen die notwendigen Umgebungsvariablen definiert und zum anderen eine Python Umgebung mit allen verpflichtenden Bibliotheken aufgesetzt. Danach ist es möglich den DeepCluster ausführen, indem die Ausführungsdatei *main.sh* gestartet wird, welche konfigurierbar (beispielweise Clusteranzahl, Epochen) ist. Nach dem Ausführen werden die gewünschten Trainingsinformationen zurückgegeben, sodass die Visualisierungsmethoden initialisiert werden können.

4.3 Implementation der Visualisierungsmethoden

Die Basis der Visualisierungsmethoden stellte der Github Nutzer Kazuto1011 bereit. Im Interesse sind hier die Implementationen der Bildbearbeitung Methoden [kazuto1011 (2020)].
Ein Workflow wurde in der *main.py* implementiert, mit den Visualisierungsmethoden. Der Ablauf wurde ähnlich wie ein ETL Prozess (extract-transform-load) aufgebaut. Zunächst müssen die gewonnenen Trainingsdaten aus dem DeepCluster extrahiert werden. Anhand diesen kann der Transformierung Prozess gestartet werden, welches das trainierte CNN Modell für die Bildverarbeitung benötigt mit den Münzen Bildern. Durch die Transformation in Tensoren, werden die mathematischen Berechnungen durchgeführt. Zum Schluss erhalten wir im Ladungs Prozess die Bilder, mit den verschiedenen Visualisierungsmethoden draufgespielt.
Auch hier entstand die Herausforderung der Automatisierung des ETL Prozess, da durch die hohen Berechnungen mit der Vielzahl von Eingabedaten eine hohe Auslastung folgte. Umgangen wurde dies, indem im Notebook spezifisch bestimmte Schritte sequentiell ausgeführt wurden.
Um Also unsere Testergebnisse erhalten zu können, musste zunächst aus der Log Datei, die im DeepCluster generiert wurde, die Zuweisung der Cluster zu jedem Bild entnommen werden. Mit diesen Informationen führt dann die Implementierung folgendes aus:

1. Lesen einer CSV Datei, welche den Pfad zum Bild mit zugehörigem Clusterziel

beinhaltet

2. Durchführen des Visualisierungsmethoden Workflows (ETL Prozess)

Nach der Durchführung solch eines Durchlaufs werden die Resultate automatisiert in einem Ordner abgespeichert. Die Ordner Struktur ist in Form von *results/<Epochenzahl>/<Zielklasse>/* aufgebaut. In genau diesem Ordner werden dann alle Bilddateien generiert, welche mit dem Namen, der angewandten Visualisierungsmethoden, versehen werden.
Diese Beschreibung eines Durchlaufs galt für eine einzelne Zielklasse. Da jedoch k-Cluster existieren, muss dies k-Mal durchlaufen werden, welches ebenfalls automatisiert ist. Ansonsten müssen hier viele manuellen Prozesse durchgeführt werden.

5 Analyse der Ergebnisse

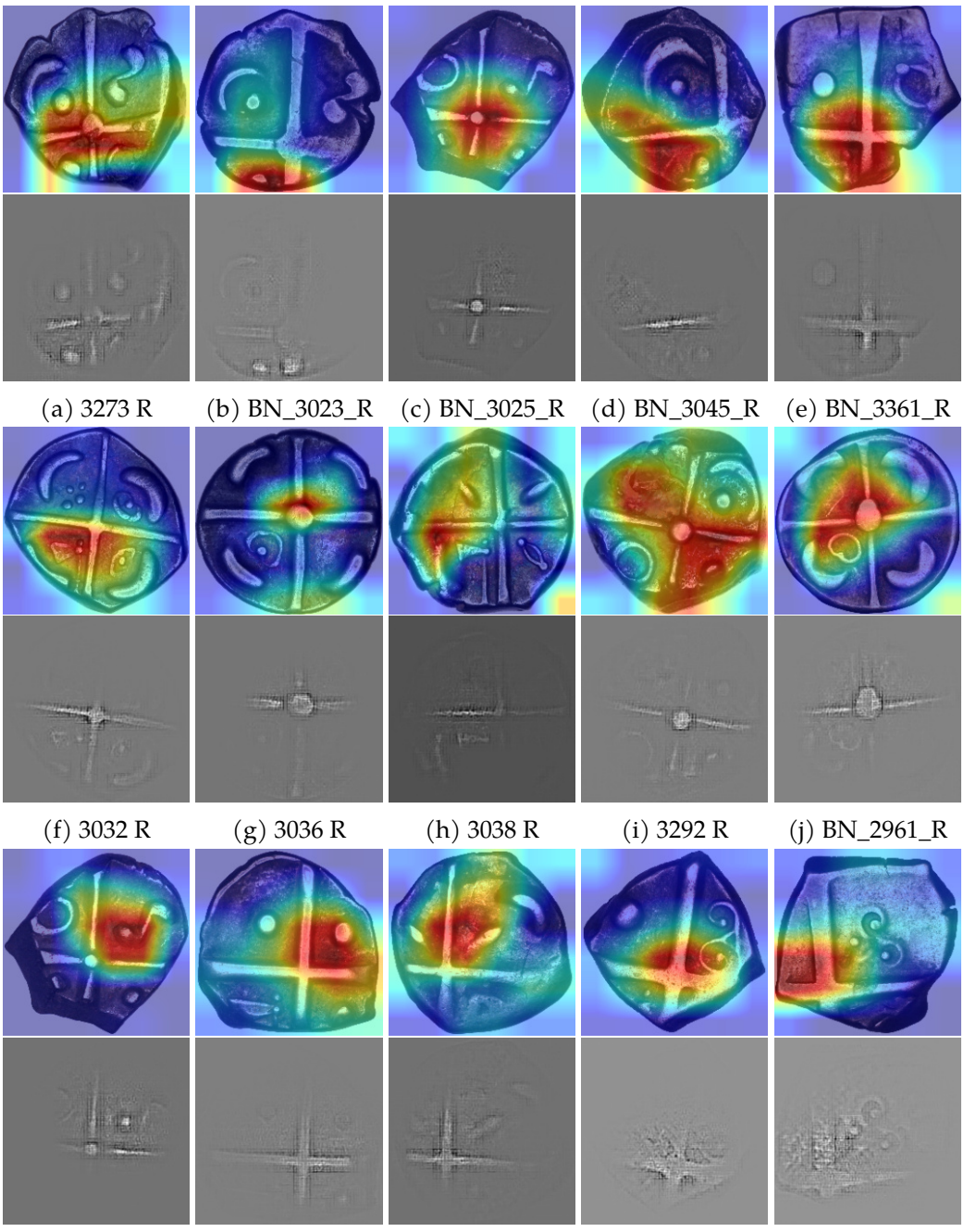
Dieses Kapitel widmet sich der Analyse und Evaluation der Versuche. Zur Analyse werden Beobachtungen in den Clustern der Versuche - durch die Versuchs ID angegeben - aufgestellt. Diese Beobachtungen beruhen auf die Ergebnisse der Visualisierungsmethoden. In jedem Cluster wird nach Merkmalen geschaut, die das Entstehen des Clusters rechtfertigen. Beim Durchsuchen aller Cluster eines Versuches werden Beobachtungen notiert und exemplarische Beispiele zur Darlegung ausgesucht, da in dieser Arbeit aus Zeitgründen nicht alle Cluster skizziert werden können. Aus den Beobachtungen werden Behauptungen erstellt, die Ansätze für die Erklärbarkeit der Merkmalserkennung liefern.

Folgende Behauptungen haben sich in den Versuchen ergeben:

1. Das Kreuz ist ein *globales* Merkmal.
2. Die horizontale Linie des Kreuzes ist ein sehr markantes Merkmal.
3. Die Qualität der Bilder in Bezug auf die Farbe und den Hintergrund ist relevant.
4. Die Abnutzung der abgebildeten Münzen ist nicht vorteilhaft.
5. Münzen, die viele Merkmale aufweisen, werden zusammen gruppiert.
6. Das Merkmal *Axt* ist markant.
7. Ähnliche Merkmale kommen nicht gemeinsam in Cluster vor.

Behauptung 1

Eine wichtige Beobachtung in den Clustern ist, dass immer Bilder vorkommen, bei denen die Stellen des Kreuzes hervorgehoben werden. In der Abbildung 7 sind (a) bis (e) aus dem Cluster 9 des Versuches R1. Aus dem selben Versuch sind (f) bis (j), jedoch aus dem Cluster 6. Die letzten 5 sind auch dem sechsten Cluster von R7



(a) 3273 R (b) BN_3023_R (c) BN_3025_R (d) BN_3045_R (e) BN_3361_R

(f) 3032 R (g) 3036 R (h) 3038 R (i) 3292 R (j) BN_2961_R

(k) BN_3025_R (l) BN_3056_R (m) BN_3251_R (n) BN_3303_R (o) BN_3304_R

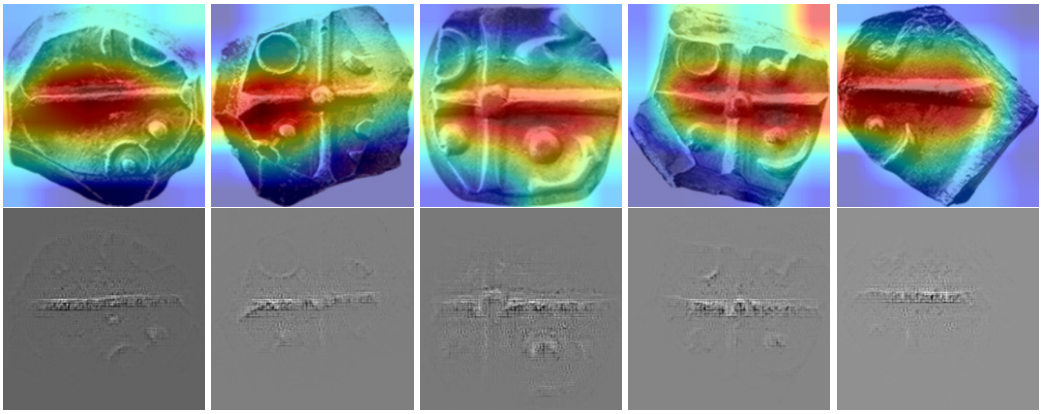
Abbildung 7: Kreuz als globales Merkmal

entnommen. Aus der Abbildung 7 stechen verschiedene Kreuz Varianten heraus. Eine dieser Varianten hat einen Punkt im Kreuz als ein besonderes Merkmal. Wie in den Ergebnissen der Guided Grad-CAM zusehen, sticht dieses Merkmal hervor. Die andere Variante besitzt kein Punkt. Obwohl unterschiedliche Merkmale gegeben sind, verteilt dass CNN Modell sie in gleiche Cluster.

In allen Versuchen und in jedem Cluster ist das Hauptmerkmal das Kreuz, da die *heat map* diese Region hervorhebt. Aus dieser Beobachtung folgt, dass die Kreuze das *globale* Merkmal in diesem Datensatz ist.
Obwohl das Modell nicht vorher *weiß*, dass in den Münzen ein Kreuz abgebildet ist, erkennt das CNN Modell die Haupt Eigenschaft des Datensatzes. Demzufolge wird durch die Anwendung der Visualisierungsmethoden deutlich, dass das CNN, sich selbst Wissen angeeignet hat.

Behauptung 2

Wie in der ersten Behauptung dargelegt, ist das Kreuz ein *globales* Merkmal. In Abbildung 8 ist das Cluster 15 aus R7 abgebildet, welche 5 der 20 Exemplare des Clusters darstellt. In diesem Cluster ist besonders auffällig, dass die Pixel, die durch die Grad-CAM hervorgehoben werden, auf der horizontalen Linie liegen. Die Visualisierung von Guided Grad-CAM hebt besonders die Breite der Linie hervor und verdeutlicht, dass dies das Merkmal des Clusters ist. In anderen Clustern wird nicht nur die horizontale Linie berücksichtigt, weshalb sich dieses Cluster, von denen unterscheidet. Interessant in diesem Beispiel ist, dass die Bilder (b) bis (e) neben der horizontalen Linie noch andere gemeinsame Merkmale vorweist.
Schlussfolgerung aus dieser Beobachtung ist, dass die horizontale Linie ein dominantes Merkmal ist. Eine mögliche Erklärung hierfür ist, dass diese Linie ein Bestandteil des globalen Merkmals ist.

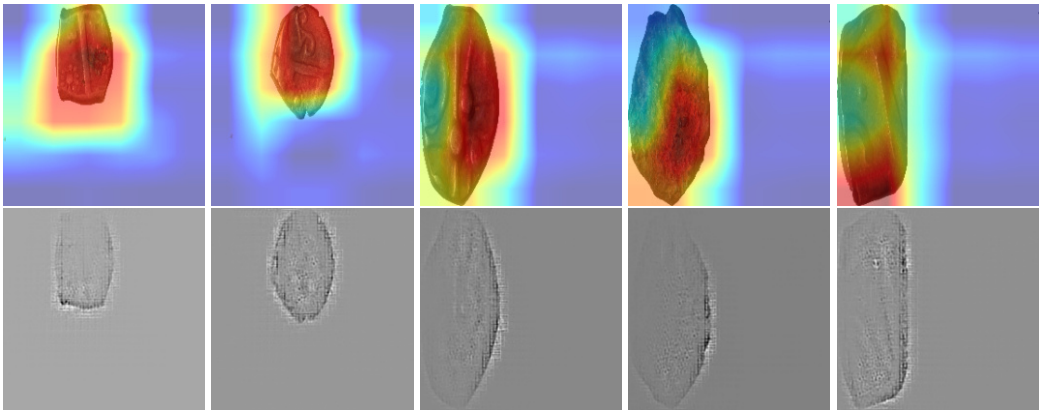


(a) 11_R copie (b) 14_R copie (c) 174_R copie (d) 28_R copie (e) 43_R copie

Abbildung 8: Horizontale Linie als dominantes Merkmal

Behauptung 3

Die abgebildeten Münzen aus der Abbildung 9 zeigen einen Ausschnitt aus Cluster 15, Versuch R6. Die Beobachtung ist, dass die Guided Grad-CAM den Fokus auf den Rändern der Münze setzt. Somit wird die äußere Form der Münze erfasst und nicht die Merkmale innerhalb. Außerdem wurde die Münze so fotografiert, dass der Hintergrund mehr als 50% des Bildes ausmacht.
Die Schlussfolgerung ist, dass die Muster auf der Münze nicht vom Modell erfasst werden können, weil die Bildqualität für diesen Anwendungsfall nicht angebracht ist. Für die Merkmalserkennung ist die Qualität des Bildes entscheidend.

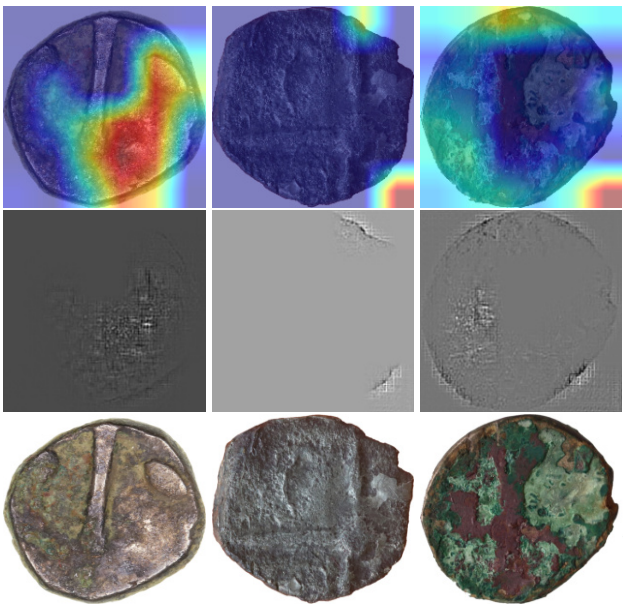


(a) L325aR (b) L325R (c) M243_33R (d) M243_42R (e) M243_46R

Abbildung 9: Münzen mit schlechter Bildqualität

Behauptung 4

In Abbildung 10 sind neben den Ergebnissen der Visualisierungsmethoden auch jeweils das original Bild zu sehen. Anhand der original Bilder ist nach menschlicher Wahrnehmung ein Abnutzen der Münzen zu erkennen. Die Methoden zur Visualisierung bestätigen, dass es Schwierigkeiten bei der Mustererkennung solcher Exemplare gibt.
Die gewonnene Erkenntnis ist, dass die Qualität der Münzen mit zu berücksichtigen ist.



(a) 4076 R (b) EYM.01.43_R (c) Ninv21R

Abbildung 10: Beispiel der Visualisierung von abgenutzten Münzen

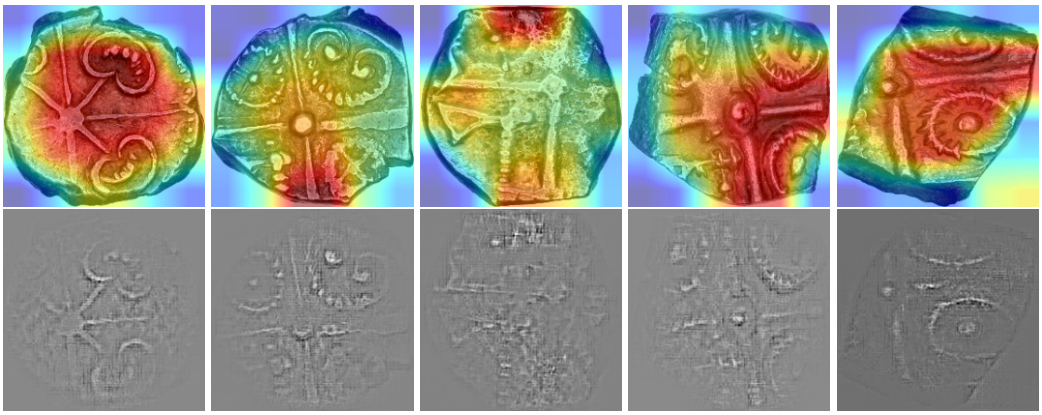
Behauptung 5

Einige Münzen weisen viele Merkmale auf. In der Abbildung 11 werden fünf Exemplare vom Cluster 1 aus dem Versuch R1 dargestellt. In diesem Cluster sind Münzen zusammen gruppiert, die verschiedene Merkmale aufweisen. Die Ergebnisse zeigen, dass das Modell alle Merkmale auffassen kann. Die Abbildung 11(b) bis 11(e) sind ähnlich, da folgende Merkmale repräsentativ für das Cluster sind:

1. das globale Kreuz Merkmal
2. eine *Brezel* ähnliche Form
3. eine Axtform
4. ein Kreis mit Zacken am Umfang und mit einem Punkt in der Mitte
5. eine *Rad* ähnliche Form

Durch die Grad-CAM werden Klassen relevante Merkmale gezeigt und in der Guided Grad-CAM werden feinkörnige Details angezeigt.
Aus dieser Beobachtung folgt, dass die Auffassung vieler Merkmale zu einer Gruppierung geführt hat. Außerdem unterstützt die Guided Grad-CAM, das Herausfinden von markanten Stellen, um besser beurteilen zu können, welches der Merkmale

die größte Bedeutung besitzt.



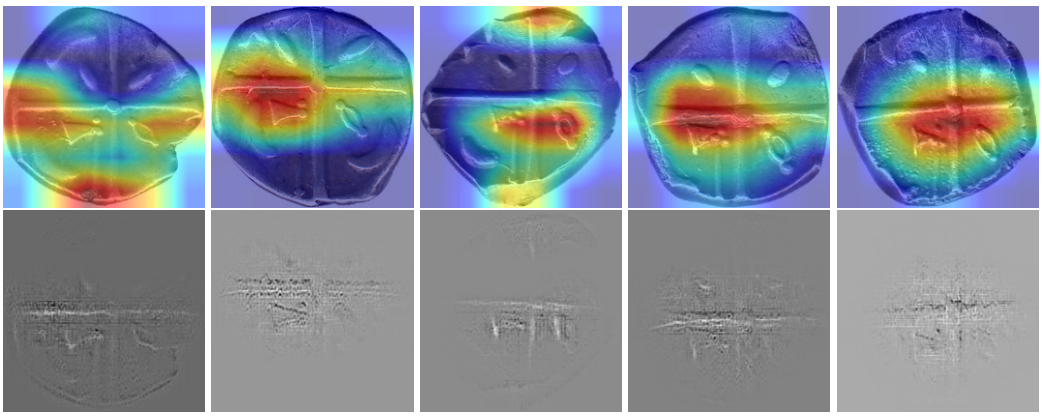
(a) BN_3460_R (b) BN_3488_R (c) BN_3491_R (d) BN_3499_R (e) BN_3500_R

Abbildung 11: Münzen mit einer Vielzahl von Merkmalen

Behauptung 6

Auf einige Münzexemplare wird eine *Axt* abgebildet. Die Darstellungen der Äxte im gesamten Münzsatz variieren und unterscheiden sich voneinander (Beispiel Abb.11(c), Abb.12(a)). Die vorliegenden Exemplare aus Abbildung 12 weisen eine ähnliche Axt vor. Die Axt ist bei allen vorliegenden Münzen, gleich positioniert (unteres, linkes Feld). Der Stiel ist dünn und verläuft senkrecht zum Rand. Dieses Merkmal wird in 12(a) durch die Visualisierung der Guided Grad-CAM genauer aufgefasst. Der dreieckförmige Kopf der Axt wird feiner von der Guided Grad-CAM wiedergegeben. Die Grad-CAM zeigt die Relevanz der Klasse. Des Weiteren wird Behauptung 2 wieder belegt, da die horizontale Linie von den Methoden für die Erklärbarkeit, bei allen Münzen vertreten ist.

Somit wird hier gezeigt, dass die Axt ein bedeutendes Merkmal ist. Nichtsdestotrotz ist die horizontale Linie im Vergleich ein bedeutenderes Merkmal als die Axt.



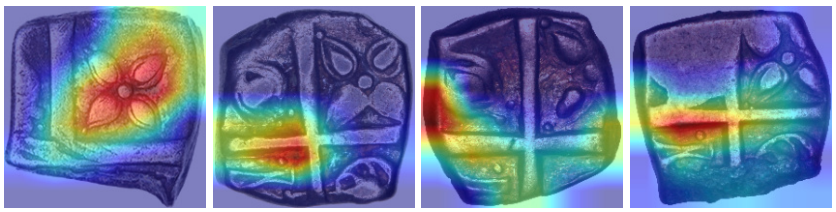
(a) MNRB-205 R (b) MNRB-206 R (c) MNRB-210 R (d) MNRB-212 R (e) MNRB-214 R

Abbildung 12: Merkmal - Axt

Behauptung 7

Im folgendem Beispiel (Abbildung 13) sticht das Muster einer *Blume* auf den ersten Blick heraus. Die Exemplare (b), (c) und (d) kommen im Cluster 43 des Versuchs R1 gemeinsam vor. In diesem Cluster ist das Hauptmerkmal das Kreuz. Die Grad-CAM hebt in diesen drei Fällen die linke Seite der horizontalen Linie am Kreuz hervor. Ebenfalls befindet sich oben rechts das Muster einer *Blume*, welches jedoch nicht als Hauptmerkmal gesehen wird. Das *blumenförmige* Muster von (b) und (d) weist im Vergleich zu (c) eine höhere Ähnlichkeit auf. Das Kreuz auf den drei Münzen ist fast auf dergleichen Höhe platziert.

Das Bild (a), welches aus einem anderem Cluster ist, hat ebenfalls eine Blume an der selben Stelle, wie die anderen vorher genannten Münzen. Jedoch zeigt die Grad-CAM für dieses Bild, dass die Klasse Blume relevant ist und nicht das Kreuz. Durch die Visualisierung wird deutlich, warum (a) nicht gemeinsam mit den anderen gruppiert wurde. Während das Modell in (a) hauptsächlich das Muster der Blume erkennt, ist es in den drei anderen das Kreuz. Diese Merkmale unterscheiden sich, weshalb die Zuordnung nachvollziehbar wird. Mit dem Wissen, dass alle Münzen das Kreuz als gemeinsames Merkmal haben, würde der Mensch andere Merkmale betrachten, um eine Gruppierung zu erzeugen. In diesem Fall würden alle vier Münzen zusammen gruppiert werden.



(a) MAN_sn17 (b) 3316 R (c) BN_3206_R (d) BN_3207_R

Abbildung 13: Vorkommen von Merkmal Blume in verschiedene Cluster

6 Fazit

Das realisierte Konzept dieser Arbeit zeigt, dass die Anwendung von bestehenden Visualisierungsmethoden hilfreich sind, um die Entscheidung eines CNNs, bei unüberwachten Lernen, zu verstehen. In dieser Bachelor Arbeit wurde bestätigt, wie wichtig die Qualität des Datensatzes ist. Eine unzureichende Qualität des Datensatzes (hier spezifisch Bilder), ist nicht für die Merkmalserkennung vorteilhaft. Da Bilder mit schlechter Qualität zusammen gruppiert werden, kann eine Datenbereinigung folgen. Dies ist eine Information, die unüberwachtes Lernen liefert.
Die Umsetzung hat gezeigt, dass Wissen über die Münzserie generiert wird, sodass das globale Merkmal herausgefunden wurde, wofür genau diese Münzserie bekannt ist (Monnaies á la croix), nämlich das Kreuz. Diese Aussage konnte nur wegen den Ergebnissen der Methoden in der erklärbaren künstlichen Intelligenz ermittelt werden. Es wird auch deutlich, dass das Modell verschiedene Merkmale berücksichtigt, um die Aufgabe der Klassifikation zu erfüllen. Anhand der gebildeten Clustern wird die Münzserie *vorsortiert*, sodass Numismatiker darauf aufbauen können.
Zusammenfassend kann gesagt werden, dass Grad-CAM und Guided Grad-CAM gute Methoden zur Erklärbarkeit von Merkmalen sind. Außerdem eignet sich die Anwendung von Clusteralgorithmen gut für die Vorsortierung von sowohl kleinen als auch großen Datensätzen.

6.1 Ausblick

Da im Rahmen dieser Arbeit die Zeit begrenzt ist, konnte das Konzept nicht ausgiebig getestet werden in Form von Versuchsdurchführungen, bei denen die CNN Architektur des Clusteralgorithmus und die Parameter im DeepCluster verändert werden, um eine breitere Anzahl von Ergebnissen zu generieren und untereinander zu vergleichen.
Des Weiteren muss überprüft werden, ob Grad-CAM und Guided Grad-CAM die aktuell besten Ansätze sind, um Merkmalserkennung mit Deep Neural Networks zu visualisieren. In der Arbeit von Ayya et al. werden andere Erklärbarkeitsmethoden zusammengefasst, die auf diesen Datensatz angewendet werden können [Ayyar et al. (2021)].

6.2 Herausforderungen

In diesem Kapitel werden die Herausforderungen der Arbeit skizziert.

6.2.1 Google Colab

Die Ressourcen und Kapazitäten der Google Colab Instanz war eine Herausforderung. Selbst mit einer lizensierten Version wurde keine optimale Leistung garantiert. So mussten Durchläufe aktiv am Rechner betreut werden, weil die Instanz ansonsten in den nicht interaktiven Modus wechselt. An anderen Tagen liefen Durchläufe teilweise langsamer, weil wahrscheinlich eine hohe Serverauslastung existierte. Doch der Höhepunkt war, dass an manchen Tagen die GPU geblockt wurde, mit der Begründung, dass die GPU zu sehr benutzt wurde.

6.2.2 DeepCluster

Die DeepCluster Implementierung erschwerte die Umsetzung in vielen Hinsichten. Zum einen hat die Anpassung der Python Version 2.7 zu 3.7 viel Zeit in Anspruch genommen. Idealerweise darf die Funktionalität des Codes nicht verändert werden, ansonsten funktioniert der Code nicht oder es liefert falsche Ergebnisse, was wie-

derrum fatal für die weitere Analyse mit den Visualisierungsmethoden wäre. Die Angabe von Tests minimieren Schwierigkeiten bei der Implementierung. Da diese nicht vorhanden waren, mussten regelmäßig Zwischenprüfungen durchgeführt werden, was viel Zeit in Anspruch nahm.
Ein generelles Problem war, dass Debbugen nicht möglich war. Zum einen wegen der Kompatibilität mit dem Betriebssystem und zum anderen, wegen Google Colab. Für Arbeiten mit dieser Thematik ist ein Linux Rechner mit viel Leistung und eine Entwicklungsumgebung von Vorteil. Das Fehlen eines Debuggers erschwert die Arbeit auf Korrektheit. Dadurch kann nicht zur Laufzeit analysiert werden, welche Daten, Datentypen und Datenstrukturen im Code erzeugt und weitergegeben werden. Beispielsweise hat die Benutzung der Klasse Dataparallel, welche auf das Modul PyTorch aufbaut, Schwierigkeiten bei der Umsetzung der Visualisierungsmethoden verursacht.

6.2.3 Visualisierungsmethoden

Bevor das Problem mit der Klasse Dataparallel behoben wurde, lag die Fehlersuche in der Implementierung der Visualisierungsmethoden.
Ein weiteres Problem war, dass mehrere Implementierungsansätze bezüglich Grad-CAM versucht wurden, doch keine dieser Ansätze hatte nach längerem Testen funktioniert, da die meisten Implementierungen nicht generisch implementiert wurden, sodass die mathematischen Berechnungen fehlschlugen.
Auch die Visualisierungsmethoden erschwerten die gesamte Umsetzung, da die mathematischen Berechnungen, mit der Verwendung von PyTorch, Anwendung auf einer GPU und Google Colab als Vereinigung betrachtet, ein Ressourcen Problem verursachte. Die Implementierung für die Visualisierungsmethoden musste so erstellt werden, dass der Code immer stückweise und automatisiert die Eingabe verarbeitet.

Literaturverzeichnis

Alex Krizhevsky, G. E. H., Ilya Sutskever. (2012). *Imagenet classification with deep convolutional neural networks.* Zugriff am 2022-01-03 auf https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Alexnet architecture. (o. J.). Zugriff am 2022-01-07 auf https://www.learnopencv.com/wp-content/uploads/2018/05/AlexNet-1.png

Ayyar, M., Benois-Pineaua, J. & Zemmaria, A. (2021). *White box methods for explanations of convolutional neural networks in image classification tasks.* Zugriff am 2022-02-26 auf https://arxiv.org/pdf/2104.02548.pdf

Caron, M., Bojanowski, P., Joulin, A. & Douze, M. (2018). Deep clustering for unsupervised learning of visual features. In *European conference on computer vision.*

Chizari, N. (2020). *Deep-learning pipeline zur erkennung von anomalien in thoraxröntgenbildern.* Zugriff am 2022-02-27 auf https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2020-proj/chizari.pdf

Contributors, T. (2019). *Softmax funktion.* Zugriff am 2022-02-22 auf https://pytorch.org/docs/stable/generated/torch.nn.Softmax.html

François, C. (2017). *Deep learning with python*. Manning Publications.

für Bildung und Forschung, B. (2021). *Classifications and representations for networks. from types and characteristics to linked open data for celtic coinages.* Zugriff am 2022-02-25 auf http://www.bigdata.uni-frankfurt.de/clarenet/

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT Press. (http://www.deeplearningbook.org)

Grad-cam. (o. J.). Zugriff am 2022-01-07 auf https://github.com/ramprs/grad-cam

Hiriart, E. (2017). *Catalogue des monnaies celtiques*. BnF Éditions.

kazuto1011. (2020). *Grad-cam with pytorch.* Zugriff am 2022-02-10 auf https://github.com/kazuto1011/grad-cam-pytorch

Miranda, L. J. (2017). Understanding softmax and the negative log-likelihood". *ljvmiranda921.github.io*. Zugriff auf https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

Nikhil Ketkar, J. M. (2021). *Deep learning with python*. Apress.

Patel, A. A. (2019). *Hands-on unsupervised learning using python*. O'Reilly Media.

Selvaraju, R. R. & et al. (2017). *Visual explanations from deep networks via gradient-based localization.* Zugriff am 2022-02-25 auf https://openaccess.thecvf.com/content_ICCV_2017/papers/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.pdf

Springenberg, B. R., Dosovitskiy. (2015). *Striving for simplicity: The all convolutional net.* Zugriff am 2022-01-11 auf JostTobiasSpringenberg*,AlexeyDosovitskiy*,ThomasBrox,MartinRiedmiller

Zhou, B. & et al. (2015). *Learning deep features for discriminative localization.* Zugriff am 2022-02-20 auf https://arxiv.org/pdf/1512.04150.pdf