CNNs - Single und Multiinput im Vergleich bei der Klassifikation keltischer Münzen

Bachelor-Thesis

geschrieben von:

von Albedyll, Axel

6XXXXXX

Betreuer:	Dr. Karsten Tolle
Lehrstuhl:	Big Data Lab

Frankfurt, 3. Oktober 2023

Zusammenfassung

Diese Arbeit beschäftigt sich mit der automatisierten Klassifizierung von keltischen Münzen mittels neuronaler Netze. Keltische Münzen sind historische Artefakte mit einzigartigen Merkmalen, deren Klassifikation Archäologen und Historikern tiefere Einsichten in ihre Herkunft und Bedeutung ermöglicht. Angesichts der Komplexität der manuellen Klassifizierung bieten neuronale Netze eine vielversprechende Alternative zur effizienten Verarbeitung großer Datenmengen. Im Kern der Studie stand der Vergleich verschiedener Netzwerkarchitekturen und Eingabedaten. Es wurden vier Ansätze untersucht: (1) Verwendung der Vorderseiten der Münzen als alleinige Dateneingabe, (2) Verwendung der Rückseiten der Münzen als alleinige Dateneingabe, (3) Verwendung der Vor- und Rückseite auf einem Bild als alleinige Dateneingabe, und (4) Verwendung sowohl der Vorder- als auch der Rückseiten der Münzen als separate Dateneingabe in parallelen Convolutional Neural Networks (CNNs), die über eine Concat-Schicht verbunden sind. Die Ergebnisse zeigten, dass das Modell, das ausschließlich auf den Rückseiten der Münzen trainiert wurde, die besten Vorhersagen erzielte. Das Multi-Input Modell, welches beide Seiten der Münzen als getrennte Eingänge nutzte, als eine vielversprechende Alternative heraus. Weiterhin wurde festgestellt, dass die Rückseiten der Münzen in der Regel einen höheren Informationsgehalt für die Klassifikation boten. Diese Arbeit hebt die Relevanz der sorgfältigen Auswahl von Netzwerkarchitekturen und Eingabedaten hervor. Die Ergebnisse legen nahe, dass bei der Klassifikation von keltischen Münzen die Verwendung getrennter Eingänge für Vorder- und Rückseiten potenziell vorteilhaft sein kann.

Inhaltsverzeichnis

Tabellenverzeichnis **vii**

Abbildungsverzeichnis **ix**

1 Einleitung **1**

2 Grundlagen **3**
2.1 Maschinelles Lernen . . . 3
2.1.1 Bestärkendes Lernen . . . 4
2.1.2 Unüberwachtes Lernen . . . 4
2.1.3 Überwachtes Lernen . . . 5
2.2 Klassifikationsprobleme . . . 7
2.3 Neuronale Netze . . . 8
2.3.1 Definition und Erklärung . . . 8
2.3.2 Convolutional Neural Network . . . 10

3 Modell und Daten **15**
3.1 ResNET-50 . . . 15
3.1.1 Verwendete Umsetzung . . . 16
3.2 Datensatz . . . 17

4 Versuch **21**
4.1 Versuchsaufbau . . . 21
4.1.1 Singleinput . . . 21
4.1.2 Multiinput . . . 22
4.2 Versuchsdurchführung . . . 23
4.2.1 Hardware . . . 23
4.2.2 Verwendete Metriken . . . 23
4.2.2.1 Accuracy . . . 23
4.2.2.2 Precision . . . 23

4.2.2.3 Recall 24
4.2.2.4 F1-Score 24
4.3 Ergebnisse 25
4.3.1 Versuch A: Originaler Datensatz 25
4.3.2 Versuch B: Angepasster Datensatz 29
4.3.3 Zusammenfassung 32

5 Fazit **33**

Literatur **35**

Appendices **39**

Anhang: Grafische Ergebnisse des Multiinput Modells für die Klassen IV und V **41**

Anhang: Ergebnisse des Versuches B **43**

Anhang: Ergebnisse des Versuches B(2) **45**

Anhang: Übersicht der Concat Modelle mit Preprocess Funktion **47**

Tabellenverzeichnis

2.1 Wahrheitstafel einer UND-Verknüpfung 9

3.1 Verteilung der Bilderanzahl je Kategorie und Klassen für Trainings- und Testdatensatz 17
3.2 Auflösung der Bilder je Kategorie und Klassen für Trainings- und Testdatensatz (in px) 18

4.1 Ergebnisse der Modelle in den Metriken *Accuracy*, *Precision*, *Recall*, *F1* für Bilder ohne Augmentation 25
4.2 Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz 28
4.3 Ergebnisse der Modelle in den Metriken *accuracy*, *precision*, *recall*, *f1* für Bilder mit Augmentation 31

1 Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz mit Augumentation 44

2 Ergebnisse der Modelle in den Metriken *accuracy*, *precision*, *recall*, *f1* für Bilder mit reduzierter Augumentation 45
3 Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz mit reduzierter Augumentation 45

4 Übersicht über die ermittelten Metriken zu den Concat Modellen unter Hinzunahme der Preprocess Funktion. 47

Abbildungsverzeichnis

2.1 Unterscheidungen maschinelles Lernen 4
2.2 Darstellung des überwachten Lernens 6
2.3 Beispielbild für Over- und Underfitting 7
2.4 Darstellung Ablauf einer Klassifikation 8
2.5 Darstellung eines einfachen neuronalen Netzes 9
2.6 Neuronales Netz für UND-Konjunktion mit Input $(v1, v2) \rightarrow (0, 1)$ und Output 0 . . 10
2.7 Neuronales Netz für UND-Konjunktion mit Input $(v1, v2) \rightarrow (1, 1)$ und Output 1 . . 10
2.8 Darstellung eines CNN 12
2.9 Verlauf der ReLu Aktivierungsfunktion 12

3.1 Beispielblock für Residual Learning 16
3.2 Darstellung zweier Münzen der Kategorie Obverse 18
3.3 Darstellung zweier Münzen der Kategorie Reverse 19
3.4 Darstellung zweier Münzen der Kategorie Merged 19

4.1 Struktur des CNNs für Single-Input Eingaben 22
4.2 Struktur des CNNs für Dual-Input Eingaben 22
4.3 Trainigsgraf und Verlauf des Loss und der Accuracy für die Single Input Modelle 26
4.4 Confusion Matrix für Single Input Modelle 27
4.5 Trainingsverlauf und Confusion Matrix für das Dual-Input Modell 28
4.6 GradCam Visualisierung einer keltischen Münze im Vergleich zwischen Single-Input (c) und Dual-Inut Modell (a, b) 29
4.7 Vergleich zwischen relevanten Informationsbereichen auf zwei Münzen mit Grad-Cam 30
4.8 GradCam Visualisierung einer keltischen Münze im Vergleich zwischen Singlein-put (c) und Multiinut Modell (a, b) 32

1 Trainingsverlaufskurven für die Accuracy und den Loss für das Dual-Input Modell auf den Klassen IV und V 41

2 Confusion Matrix für das Dual-Input Modell auf den Klassen IV und V 42

3 Confusion Matrix für Single- und Multi-Input Modelle auf Datensatz mit Augumentation 43

4 Confusion Matrix für Single- und Multi-Input Modelle auf Datensatz mit reduzierter Augumentation 46

5 Confusion Matrix für Mutli-Input Modelle mit Preprocess Funktion 48

Notation und Abkürzungsverzeichnis

ML	Machine Learning
CNN	Convolutional Neural Network
PKW	Personenkraftfahrzeug
ReLu	Rectified Linear Unit Aktivierungsfunktion
TP	True Positive
FP	False Positive
FN	False Negative
GradCam	Gradient-weighted Class Activation Mapping
ResNET	Residual Neural Networks
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ONEIROS	Open-ended Neuro-Electronic Intelligent Robot Operating System

Obverse	Single-Input Modell mit Münzvorderseiten als Dateninput
Reverse	Single-Input Modell mit Münzrückseiten als Dateninput
Merged	Single-Input Modell mit zusammengefügten Münzseiten als Dateninput
Concatenate	Multi-Input Modell mit Vor- und Rückseiten von Münzen als Input
Concat	Multi-Input Modell

L	Lossfunktion
Ω	Regularisierungsterm
ρ^{hlim}	Aktivierungsfunktion
o	Softmax Funktion

1 Einleitung

Die Einordnung von Kategorien für gesehene Objekte stellt für Menschen oft eine komplexe und zeitaufwändige Aufgabe dar. Die Möglichkeit, Objekte basierend auf ihren Merkmalen zu klassifizieren, ist jedoch von großer Bedeutung in vielen Anwendungsbereichen, darunter Bilderkennung, medizinische Diagnose, Spracherkennung und mehr [29][28]. In den letzten zwei Jahrzehnten hat sich eine vielversprechende und zunehmend ausgereifte Alternative herauskristallisiert. Der Einsatz von maschinellem Lernen.
Im Zeitalter der Digitalisierung und angesichts der rasanten Entwicklung der verfügbaren Rechenkapazitäten im Einklang mit dem Mooreschen Gesetz wird die Kategorisierung nicht mehr ausschließlich vom Menschen selbst durchgeführt [35]. Zunehmend wird durch den Einsatz von Computern als Maschinen dieser Prozess automatisiert [38]. Es hat sich die automatisierte Zuordnung zu Kategorien als vielversprechende Alternative erwiesen [7]. Durch den Einsatz von maschinellem Lernen können Computer große Datenmengen verarbeiten und Muster in den Daten erkennen, um eine automatische Klassifikation durchzuführen [12].
Bei der Klassifikation können sich die mathematischen Ansätze jedoch erheblich unterscheiden. Die Bandbreite der verwendeten Algorithmen reicht von Entscheidungsbäumen über Support Vektor Maschinen bis hin zu komplexen neuronalen Netzen (NN) [43][45]. Hierbei haben sich die neuronalen Netze in den letzten Jahren als äußerst leistungsfähige Werkzeuge für die Klassifikation herausgestellt [48]. Sie bestehen aus miteinander verbundenen künstlichen Neuronen, die in Schichten organisiert sind und in der Lage sind, komplexe Muster in den Daten zu erkennen und zu verarbeiten.
In dieser Arbeit wird an die bestehende Forschung zur Architektur von NN für Klassifikationen angeknüpft. Das spezifische Ziel dieser Arbeit ist es, die Leistung verschiedener Netzwerkarchitekturen für ein bestehendes Klassifikationsproblem von keltischen Münzen zu vergleichen. Keltische Münzen sind historische Artefakte, die in archäologischen Ausgrabungen entdeckt werden und oft mit einzigartigen Merkmalen versehen sind. Die Klassifikation von keltischen Münzen kann Archäologen und Historikern helfen, ihre Herkunft, Verwendung und Bedeutung besser zu verstehen.
Die Architekturen der neuronalen Netze, die verglichen werden, unterscheiden sich primär in den zugeführten Eingabedaten sowie der eingesetzten Netzstruktur. Im Rahmen dieser Arbeit

werden vier verschiedene Ansätze untersucht:

1. Verwendung der Vorderseiten der Münzen als alleinige Dateneingabe (Singleinput)
2. Verwendung der Rückseiten der Münzen als alleinige Dateneingabe (Singleinput)
3. Verwendung der Vor- und Rückseite auf einem Bild als alleinige Dateneingabe (Singleinput)
4. Verwendung sowohl der Vorder- als auch der Rückseiten der Münzen als Dateneingabe (Multiinput)

Diese verschiedenen Ansätze ermöglichen es uns, die Auswirkungen der verwendeten Daten auf die Leistung des neuronalen Netzwerks zu untersuchen und zu vergleichen.
Die Arbeit gliedert sich in fünf Kapitel. Im ersten Kapitel wird eine kurze Einführung in das Thema gegeben und die wissenschaftliche Relevanz erläutert. Wir diskutieren die Herausforderungen der manuellen Klassifikation von Objekten und die Vorteile des maschinellen Lernens für automatisierte Klassifikationen.
Anschließend werden in Kapitel 2 die theoretischen und praktischen Grundlagen, basierend auf der Literatur, vorgestellt. Es werden die Konzepte der Klassifikation, die Struktur neuronaler Netze und die verschiedenen Algorithmen erläutert, welche in der Klassifikation eingesetzt werden.
Im dritten Kapitel präsentieren wir den verwendeten Datensatz von keltischen Münzen. Zudem stellen wir die Architektur des neuronalen Netzwerks vor, das wir für die Klassifikation implementiert haben. Des Weiteren wird erläutert, wie die Daten in das Netzwerk eingespeist werden.
Im vierten Kapitel führen wir die Experimente durch und werten die Ergebnisse aus. Es wird die Durchführung der Experimente und die verwendeten Metriken zur Bewertung der Leistung der verschiedenen Netzwerkarchitekturen beschrieben. Anschließend werden die Ergebnisse unserer Experimente präsentiert und im Hinblick auf die gestellten Forschungsfragen diskutiert.
Abschließend wird im Fazit ein Resümee gezogen und es wird der Erfolg des Ziels, die Modellleistung für das Klassifikationsproblem von keltischen Münzen unter Verwendung verschiedener Netzwerkarchitekturen zu vergleichen, bewertet. Es werden die wichtigsten Erkenntnisse unserer Arbeit hervorgehoben und auf offene Forschungsfragen hingewiesen, die sich aus dem Versuch ergeben haben.

2 Grundlagen

In diesem Kapitel werden die Grundlagen für das Verständnis der Arbeit näher gebracht. Hierbei untergliedern wir in vier Unterkapitel. Zunächst wird der Begriff des maschinellen Lernens und nachgelagert die hier verwendete Methode des überwachten Lernens (engl. *Supervised Learning*) erläutert. Im Folgenden wird die allgemeine Funktionsweise und Grundstrukturen von neuronalen Netzen, sowie die Besonderheiten der in dieser Arbeit verwendeten Modelle, den *Convolutional Neural Network* (CNN) vorgestellt und erklärt.

2.1 Maschinelles Lernen

„Machine learning (ML) is a field of computer science that studies algorithms and techniques for automating solutions to complex problems that are hard to program using conventional programing methods.“[32]

Aus der obigen Definition von Rebala et al. ist zu entnehmen, dass mit maschinellem Lernen das automatisierte Lösen von komplexen Problemen unter Verwendung von mathematischen Algorithmen und Methoden verstanden werden kann. Hierbei wird in der Literatur zwischen verschiedenen Arten des Lernens unterschieden (vgl. Abbildung 2.1). Die drei meistverbreiteten Kategorien sind die Folgenden:

1. *Reincorcement Learning*
2. *Unsupervised Learning*
3. *Supervised Leraning*

Als Nächstes werden die einzelnen Unterarten sowie deren Besonderheiten kurz dargestellt. Im Rahmen dieser Arbeit liegt der Fokus auf dem Bereich des Supervised Learning.

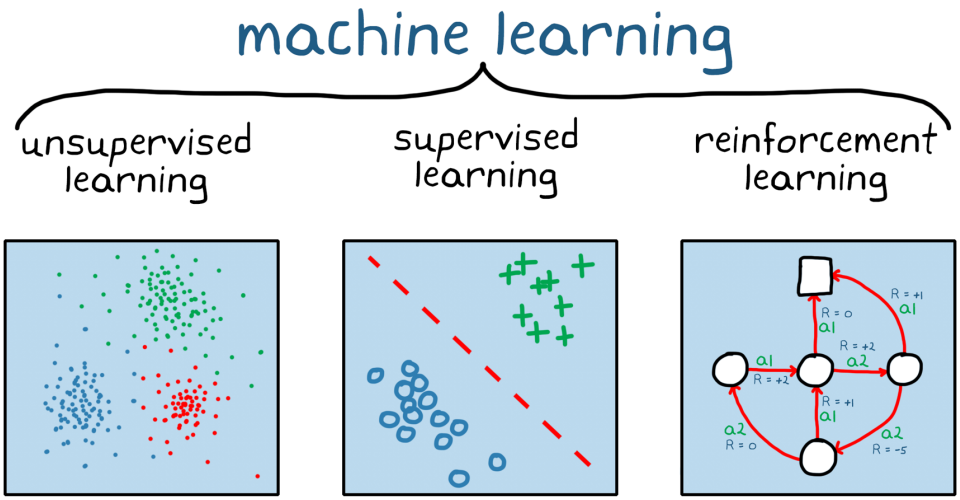


Abbildung 2.1: Bekanntesten Unterscheidungsformen des maschinellen Lernens in bestärkendes Lernen, unüberwachtes Lernen und überwachtes Lernen [22]

2.1.1 Bestärkendes Lernen

Unter Reinforcement Learning, versteht man das automatisierte Lernen eines Agenten innerhalb einer definierten Umgebung durch Trial-and-Error (Bestärkendes Lernen) [22][20]. Basis ist hierbei eine Belohnungsfunktion des Agenten, welche es zu optimieren gilt und dementsprechend verschiedene Verhalten evaluiert werden. Ein besonderer Punkt bei dieser Art des Lernens ist, dass der Verbesserungsprozess ohne menschliches Eingreifen erfolgt und somit sukzessive ein Verhaltensmuster erlernt wird. Bekannte Einsatzgebiete für Reinforcement Learning stammen unter anderem aus der Spieleindustrie, bei welchem trainierte Agentensysteme menschliche Persönlichkeiten in beispielsweise Schach oder anderen Brettspielen schlagen sollen[6]. In Abbildung 2.1 können wir im rechten Bild eine schematische Darstellung dafür sehen. Hier ist ein Zielzustand (eckiger Kasten) und mehrere Zwischenzustände abgebildet. Zwischen den Zuständen sind Pfeile dargestellt, mit jeweils einem Wert R für die Belohnungsfunktion. Dieser Wert R erhöht sich bei Annäherung zu dem Zielzustand und verringert sich gleichermaßen bei einer Erhöhung der Entfernung.

2.1.2 Unüberwachtes Lernen

Die zweite, vorgestellte Methode befasst sich mit der Art des unüberwachten Lernens (engl. Unsupervsied Learning). Unsupervised Learning bedeutet, dass ähnlich wie beim bestärkenden Lernen kein menschliches Eingreifen erfolgt [9]. Der Unterschied hierbei liegt jedoch darin, dass keine Nutzenfunktion eines Agenten optimiert wird, sondern vielmehr vorhande Strukturen und Gruppen innerhalb der Datenwolke gesucht werden [14]. Dies kann je nach betrachte-

tem Anwendungsgebiet beispielsweise eine mathematische Distanz-Metrik wie Manhatten oder Euklid sein. In Abbildung 2.1 kann man dies im linken Bild erkennen. Hier sehen wir mehrere Datenpunkte in Punktewolken, welche als Ergebnis eines unüberwachtes Lernen unterschiedlichen Farben und damit Gruppen oder Clustern zugeordnet wurden. Für Einsatzgebiete ergibt sich hier eines der zentralen Anwendungsgebiete in Form der Strukturierung und Clusterung von vorliegenden Daten ohne ein direktes menschliches Zutun [14][9]. Gängige Algorithmen sind hier hierarchische Clusterung, k-Means oder K Nearest Neighbors (KNN), auf welche im Rahmen dieser Arbeit aber nicht weiter eingegangen wird [16][9].

2.1.3 Überwachtes Lernen

In Rahmen dieser Arbeit befassen wir uns mit der dritten und am weitesten verbreiteten Kategorie des maschinellen Lernens: dem Supervised Learning. Während wir bei den zuvor vorgestellten Verfahren ungelabelte Daten vorliegend haben, liegen bei dem überwachten Lernen die Daten in beschrifteter Form vor [8][27]. Mathematisch wird mithilfe dieses Lösungsansatzes versucht, ein Zusammenhang zwischen einem gegebenen Input (X) und einem zu vorhersagenden Output ($\hat{y}$) herzustellen. In Abbildung 2.2 ist ein allgemeiner Ablauf für ein Training eines solchen Problems dargestellt. Der dargestellte Prozess besteht aus einem Modell (engl. *model*), der Schlussfolgerung (engl. *inference*), den Daten (engl. *data*) und einer Verlustfunktion (engl. *loss*). In der Abbildung ist zu sehen, dass mithilfe des Modells Vorhersagen ($\hat{y}$) für das Gesuchte y ermittelt und mit den realen Daten in einen mathematischen Kontext, dem Loss, gesetzt und anschließend wieder als Information für mögliche Modellanpassungen zurückgespielt werden. Mathematisch ergibt sich eine sogenannte *objective*-Funktion [10]:

$$obj(\theta) = L(\theta) + \Omega(\theta) \tag{2.1}$$

Die dargestellte Rechnung setzt sich aus zwei Summanden zusammen. Mit dem ersten Summanden wird der eben bereits erwähnte Loss dargestellt, mit welchem die Güte der Vorhersagen mit real existierenden Werten ermittelt wird. Die Wahl der Loss Funktion kann je nach betrachtetem Problem (Regression, Klassifikation, etc.) unterschiedlich sein.
Der zweite Summand aus 2.1 dem Omega ist der sogenannte *Regularisierungsterm* (engl. regularization term) [10][44]. Unter Verwendung diesen Kontrollterms, lässt sich die Komplexität des verwendeten Modells regulieren, um beispielsweise zu vermeiden, dass auf den zugrunde liegenden Daten ein reines auswendiges Lernen stattfindet, sondern vielmehr der algorithmische Zusammenhang gefunden werden kann. *Overfitting* steht hierbei für den Fall, dass das Modell genaue Vorhersagen auf den Trainingsdaten liefert und nicht auf den neuen ungesehenen Daten. [47][23]. Mathematisch wird hier kein Zusammenhang innerhalb der Struktur der Daten gefunden und lässt sich damit nicht auf neue ungesehene Daten adaptieren. Als Resul-

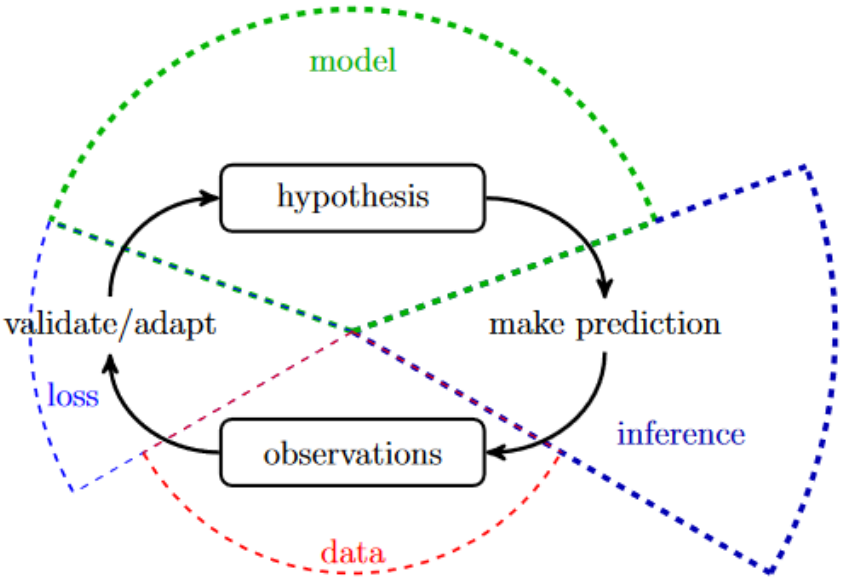


Abbildung 2.2: Darstellung des überwachten Lernens [20]

tat eines über angepassten Modells an die Trainingsdaten können neue, noch nicht gesehen, aber möglicherweise sehr ähnliche Datenpunkte nur schlecht vorhergesagt werden.
Gleichermaßen wird über diesen Term auch der gegenläufige Effekt, das *Underfitting* kontrolliert [23]. Hierbei ist das trainierte Modell nur begrenzt in der Lage korrekte Vorhersagen zu treffen, da zum Beispiel zu wenige oder die falschen Variablen für den Forecast gewählt wurden. Diese ungewünschten Effekte sind beispielhaft in Abbildung 2.3 dargestellt. In der Grafik sehen wir vier Graphen, bei denen jeweils auf der x-Achse die Zeit t und auf der y-Achse das Interesse eines Nutzers abgebildet ist. Ziel eines ML Ansatzes, wäre es beispielsweise eine visuelle Stufenform durch die Punkte zu legen, um sie zu beschreiben. In dem Graphen oben rechts können wir ein Beispiel von Overfitting erkennen. Hier wurden zu viele Teilstufen verwendet und damit eine erhöhte Modellkomplexität ermittelt. Anders hingegen ist die Situation im Bild links unten. Hier wurde lediglich eine Stufe eingesetzt, was zwar in einem geringen $\Omega(\theta)$ jedoch in einen hohen Loss-Wert, dem $L(\theta)$ resultiert. Hier haben wir den Fall des Underfittings vorliegen. In dem letzten Teilgraph, unten rechts, haben wir eine gute Positionierung und Anzahl der Stufen für das dargestellte Datenmuster gefunden. Da sich eine gute Balance zwischen Loss und Regularisierungsterm ergibt, konnte das Modell auf dem Problem gut generalisiert lernen.
Wenn wir dies nun auf unsere Abbildung 2.1 rückführen, sieht man in der Mitte ein mögliches Anwendungszenario für Supervised Learning. Hier wurden die Daten in Kreise oder Plus gelabelt, und Ziel soll es nun sein hier eine Trennung anhand dieser Daten mathematisch zu beschreiben. Das wird beispielhaft durch Aufstellung einer Geraden als rote Trennlinie dargestellt. Wie bereits zu Anfang erwähnt, stellt diese Art der Problemstellung die häufigste Erscheinungs-

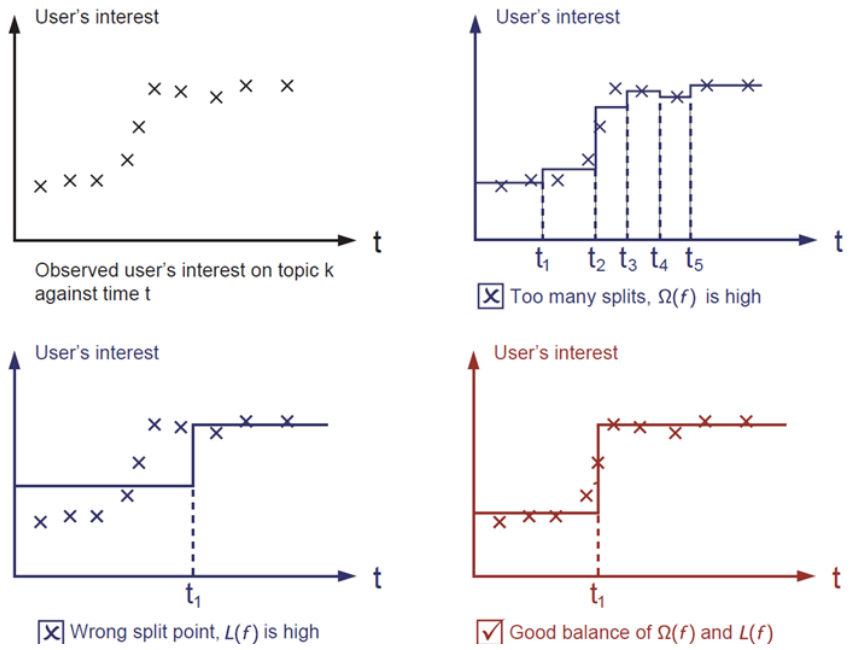


Abbildung 2.3: Beispielbild für Over- und Underfitting. Mit $\Omega(f)$ als Regularisierungsterm und $L(f)$ als Lossfunktion [44]

form im Bereich des ML dar. Während sich die Datengrundlage und das allgemeine Finden des optimalen Modells (vgl. Abbildung 2.2) vergleichen lässt, so unterschiedlich können die zu vorhersagenden Szenarien sein. Neben klassischen Regressions- oder Zeitreihenproblemen aus dem Bereich des Supervised Learnings mit stetigen Wertebereichen kann ebenfalls auch eine Zuweisung von Klassen für neue Daten Ziel der Vorhersage sein [8]. Diese Art möchten wir uns im nächsten Unterkapitel einmal näher anschauen, da sie für die Einordnung der Münzen angewendet wurde.

2.2 Klassifikationsprobleme

„[...] is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown“[25]

Im Rahmen diesen Kapitels sollen dem Leser Klassifikationsprobleme und damit auch der Forschungsgegenstand dieser Arbeit näher gebracht werden. Aus der obigen Definition kann man bereits erkennen, dass es sich bei Klassifikationen um Probleme aus dem Bereich des Supervised Learning handelt. Hierbei haben wir wieder Daten vorliegend, welche gelabelt und in unterschiedliche Kategorien oder auch Klassen eingeordnet wurden. Ziel ist es jetzt, wie aus

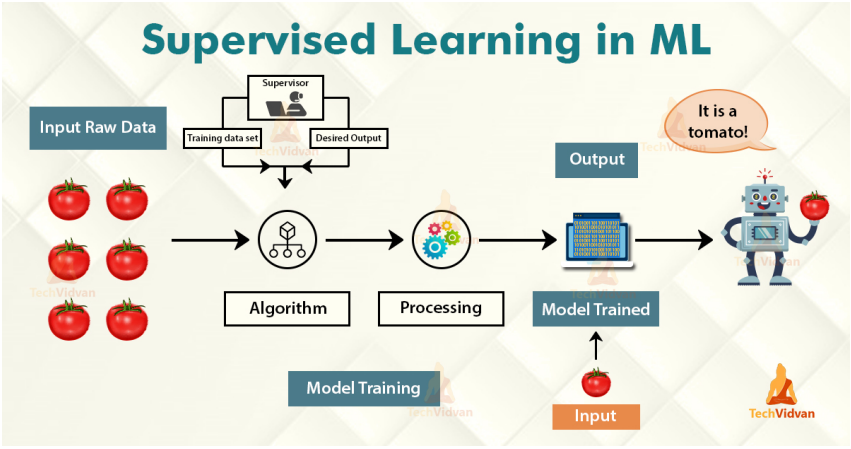


Abbildung 2.4: Darstellung Ablauf einer Klassifikation anhand einer Objektlassifikation in Form von Gemüse (Tomaten) [40]

Kapitel 2.1.3 bekannt, ein Modell auf Basis dieser Daten zu trainieren und somit auf neuen Daten eine entsprechende Einordnung durchzuführen. In Abbildung 2.4 ist dies schematisch dargestellt. Hier sehen wir auf der linken Seite unsere Datengrundlage, in diesem Fall Tomaten, welche im Bereich *Model Training* in ein Klassifikationsproblem übertragen werden. In diesem Schritt können wir oberhalb den Domänenexperten Mensch sehen, welcher auf die Daten eine Zuweisung von Objekt und Klasse durchführt. Im Anschluss erhalten wir ein trainiertes Modell, welches einen neuen Input verarbeiten und idealerweise korrekt klassifizieren kann. In diesem Schaubild wurde vereinfacht nur eine Klasse dargestellt. In der Realität würde es an dieser Stelle eine Vielzahl möglicher Kategorien geben.

2.3 Neuronale Netze

2.3.1 Definition und Erklärung

Mit dem Vorbild des menschlichen Gehirns bilden neuronale Netze einen Teilbereich der Neuroinformatik und orientieren sich an ihrem menschlichen Vorbild in vereinfachter Form hinsichtlich der Wissensverarbeitung [19]. Ähnlich wie beim Menschen befinden sich in einem künstlichen neuronalen Netz Neuronen, welche miteinander verknüpft sind und in verschiedenen, hintereinander gelagerten Schichten (engl. *Layer*) vorliegen [46]. Ein Beispiel für ein solches Netz ist in Abbildung 2.5 gegeben. Hier können wir die eben erwähnten Schichten erkennen. Die erste Schicht eines allgemeinen Netzes ist der sogenannte *Input Layer*. Über diesen werden die Daten enkodiert und in das Netz eingespielt. Nachfolgend sind hier exemplarisch zwei *Hidden Layer* abgebildet, welche auf die Daten angepasst werden und damit dann der letzten Schicht, dem *Output Layer* ein Signal und das Ergebnis präsentieren. Es gilt hier zu erwähnen,

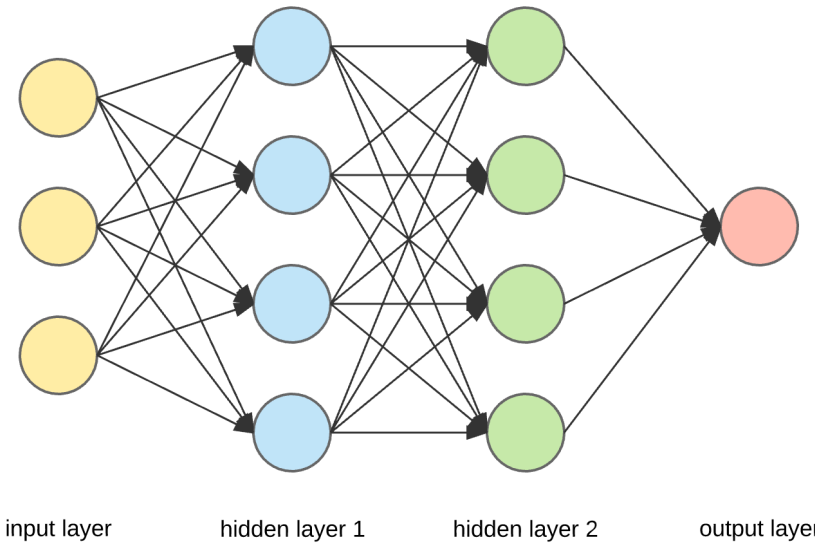


Abbildung 2.5: Darstellung eines einfachen neuronalen Netzes [15]

dass im Bereich der Hidden Layer, durchaus auch weniger oder mehrere Schichten vorhanden sein können [2]. Im Folgenden möchten wir die Art und Weise der Daten- beziehungsweise Wissensverarbeitung an einem einfachen Beispiel verdeutlichen. Hierfür betrachten wir ein logisches UND aus der booleschen Aussagenlogik. Der zu betrachtende Term stellt sich folgendermaßen dar:

$$f(v1, v2) = v1 \wedge v2 \tag{2.2}$$

Um nun das korrekte Verhalten für einen Input von $v1$, $v2$ bewerten zu können, betrachten wir die Wahrheitstafel für den Term 2.2. Für das Netz gilt, dass die Kanten zwischen den Neuro-

v2	v1	f(v1,v2)
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 2.1: Wahrheitstafel einer UND-Verknüpfung

nen mit einer Übergangsfunktion versehen sind und den gegebenen Dateninput entsprechend verarbeiten. Gleichermaßen gibt es für jedes der gegebenen Neuronen im Hidden und Output Layer einen Schwellenwert, welcher überwunden werden muss, damit das Neuron angeregt ist und eine entsprechende Aussage für das Ergebnis ermöglicht [50]. In Abbildung 2.6 können wir eine beispielhafte Darstellung sehen. Hierbei stellen die gelben Neuronen den Input, das blaue Neuron den Hidden und das rote Neuron den Output Layer dar. Gleichermaßen können wir sehen, dass die beiden Kanten w_1 und w_2 eine Gewichtung 1 tragen und das Neuron einen

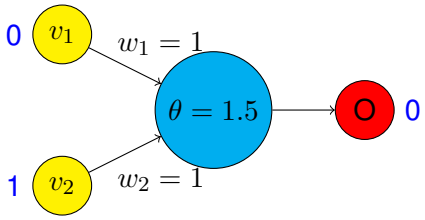


Abbildung 2.6: Neuronales Netz für UND-Konjunktion mit Input $(v1, v2) \rightarrow (0, 1)$ und Output 0

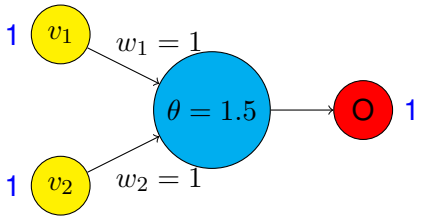


Abbildung 2.7: Neuronales Netz für UND-Konjunktion mit Input $(v1, v2) \rightarrow (1, 1)$ und Output 1

Schwellenwert von $\theta = 1,5$ besitzt. Damit das Neuron angeregt ist und ein korrektes Verhalten am Output abbildet, müssen wir eine passende Aktivierungsfunktion wählen [49][50].

$$\rho^{hlim}(v) = \begin{cases} 1 & v > 0 \\ 0 & v \leq 0 \end{cases} \tag{2.3}$$

Anhand der Gleichung 2.3 können wir sehen, dass genau dann das Neuron angeregt ist, wenn v_i größer 0 ist. Für unser Neuron im Hidden Layer gilt für den gegebenen Input 2.7 folgendes für die Aktivierungsfunktion:

$$\rho^{hlim}((w_1 \cdot v_1 + w_2 \cdot v_2) - \theta) = \rho^{hlim}((1 \cdot 1 + 1 \cdot 1) - 1.5) = \rho^{hlim}(0.5) = 1 \tag{2.4}$$

Mit der Gleichung 2.4 können wir den Zusammenhang für unser gesuchtes Ergebnis der Konjunktion 2.1 ablesen. Hierbei darf nur dann eine 1 und damit ein aktiviertes Neuron im Output vorhanden sein, wenn sowohl v_1 als auch v_2 gleich 1 sind. Dies wird mit denen hier gewählten Kantengewichten, Schwellenwerten sowie Aktivierungsfunktionen erreicht. Bei komplexeren realen sowie wissenschaftlichen Problemen, ist eine entsprechende Lösung nicht immer so trivial ersichtlich, sondern besteht aus einem iterativen Prozess, in welchem über mehrere Schritte (Epochen) eine fortlaufende Anpassung der Kantengewichte, Verknüpfungen zwischen der Neuronen sowie der eingesetzten Schwellenwerte besteht, bis eine akzeptable Datenrepräsentation in der Netzstruktur abgebildet ist [41][50].

2.3.2 Convolutional Neural Network

In diesem Unterkapitel soll die Funktionsweise eines Convolutional Neural Network (CNN) näher dargestellt werden. Diese Art der Netze findet in unserem Versuch in Kapitel 4 Verwendung. Ein CNN stellt einen speziellen Typ eines künstlichen neuronalen Netzwerks dar, welches hauptsächlich im Bereich der Verarbeitung von Bildern und anderen Gitterdaten wie Sprach- und Audiodaten verwendet wird. Das besondere eines CNN ist das Zusammenspiel von drei elementaren Schichten [26]:

1. Faltungsschichten (engl. *Convolutional Layer*)
2. Pooling-Schichten
3. Vollständig verbundene Schichten (Fully Connected Layer)

Zu Anfang eines jeden CNNs steht wieder ein Input Layer, in welchem die gegebenen Daten, zum Beispiel Bilder, empfangen werden [26][30]. Bei einem Bild besteht die Eingabeschicht aus einer mehrdimensionalen Matrix von Pixelwerten. Jedes Pixel repräsentiert die Intensität oder Farbinformation an dieser Position im Bild.
Die nächste Schicht stellt das Herzstück eines CNNs dar, die *Pooling Schicht*. In dieser Ebene werden aus den Bildern markante Merkmale identifiziert, welche für die Problemlösung relevant sind. Dies erfolgt mithilfe eines Filters, dem sogenannten *Kernel*, welcher über das Eingabebild geschoben wird (Faltung) [26]. Während der Faltung wird jeder Filter beispielsweise mit einem Teil des Eingangsbildes elementweise multipliziert und die Ergebnisse summiert. Zum tieferen Verständnis betrachten wir an dieser Stelle ein Beispiel für die Funktionsweise eines Kernels. Angenommen wir haben einen 3x3 Kernel (links) und ein Bild (rechts), welche folgendermaßen definiert sind:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2 & 4 & 6 & 1 \\ 3 & 5 & 2 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

Für dieses Bild und den gegeben Kernel ergeben sich die folgenden Ergebnisse für die Berechnungen.

$$\begin{pmatrix} 2 & 4 & 6 \\ 3 & 5 & 2 \\ 1 & 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 & +4 \cdot 0 & +6 \cdot 1 \\ 3 \cdot 0 & +5 \cdot 1 & +2 \cdot 0 \\ 1 \cdot 1 & +2 \cdot 0 & +3 \cdot 1 \end{pmatrix} = \begin{pmatrix} 8 \\ 5 \\ 4 \end{pmatrix}$$

Hierbei sehen wir, dass der Kernel einen Teil des Bildes überlappt und an diesen Stellen multipliziert und summiert wird. Diese Berechnungen werden für jede Position im Eingangsbild wiederholt, bis der Kernel einmal über das komplette Bild gelaufen ist. Wir können in Abbildung 2.8 sehen, dass zu Anfang ein Bild eines PKW gegeben ist. Darauf folgt das Scannen mittels des Filters auf markante Merkmale, welche eine Aktivierungskarte von relevanten Teilbereichen erzeugt. Die Gewichtungen im Kernel können variieren und durch das Training des CNNs optimiert werden, um spezifische Merkmale im Eingangsbild zu erkennen. Durch die Verwendung verschiedener Kernel mit unterschiedlichen Gewichtungen kann das CNN komplexe Merkmale wie Kanten, Texturen oder Formen in Bildern erfassen. Gleichermaßen können mehrere Kernel in einem Convolutional Layer kombiniert werden, indem die Aktivierungskarten kombiniert werden[39].
Der nächste Punkt der Abbildung 2.8 ist hier mit ReLu gekennzeichnet. Allgemein steckt hier

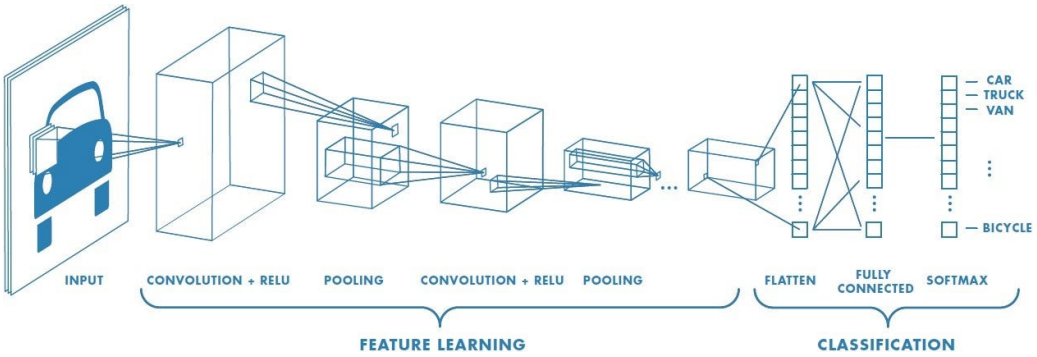


Abbildung 2.8: Prozessskizze einer Bildklassifikation mittels eines CNN [3]

eine Aktivierungsfunktion, welche wir bereits in Kapitel 2.3 kennengelernt haben. Die ReLu Funktion unterscheidet sich im Gegensatz zu der vereinfachten Funktion aus unserem Beispiel lediglich darin, dass ihr Wert linear wie in Gleichung 2.5 ersichtlich fortgetragen wird [3] (vgl. Abbildung 2.9). Im nächsten Schritt kommt eine Dimensionsreduktion der Aktivierungskarten

$$\rho^{hlim}(v) = \begin{cases} v & v \geq 0 \\ 0 & v < 0 \end{cases} \qquad (2.5)$$

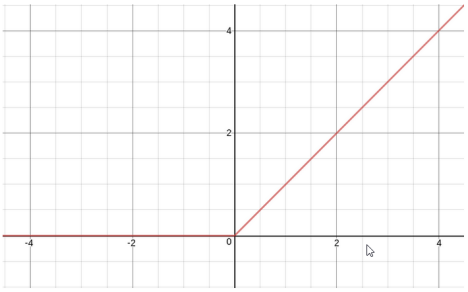


Abbildung 2.9: Verlauf der ReLu Aktivierungsfunktion [3]

mithilfe der *Pooling Layer*. Eine der häufigsten Pooling Methoden ist das sogenannte *Max-Pooling* [4]. Hierbei wird innerhalb eines bestimmten Bereichs der Aktivierungskarte der größte Wert ausgewählt und die anderen Werte verworfen. Durch diese Dezimierung der Informationen werden die erfassten Merkmale der Bilder auf die wichtigsten Informationen reduziert und die Rechenlast verringert. Gleichermaßen wird durch die Reduktion der Modellkomplexität die gefundenen Merkmale robuster gegenüber kleinen Verschiebungen und Variationen [4]. Neben dem Max-Pooling gibt es in der Literatur auch andere Verfahren, wie etwa Mittelwertbildung und Summenbildung. Jedoch liefert die nach Albawi et al. das Max-Pooling mitunter die besten Ergebnisse unter Erreichung einer Reduktion der Eingangsgrößen von etwa 75%. Aus Abbildung 2.8 können wir sehen, dass sich diese Abfolge von Convolutional Layer, Aktivierungsfunktion und Pooling Layer durchaus wiederholen kann und oft ein mehrstufiges Modell darstellt. Innerhalb dieser Schichten wird das eigentliche Muster für die Problemstellung gefunden und im *Feature Learning* festgehalten.
Im nächsten Schritt eines CNN kommt der Bereich der eigentlichen Klassifikation. Nach dem

letzten Pooling Layer kommt ein sogenannter *Flatten Layer*. Wie der Name schon vermuten lässt, wird hierbei aus der resultierenden Matrix der vorangegangenen Schicht ein Spaltenvektor für die weitere Verarbeitung erzeugt. Nach dieser Strukturumformung folgen ein oder mehrere *Fully Connected Layer*. In diesen Layern sind alle Neuronen mit allen Neuronen der vorhergehenden Schicht verknüpft. Die vollständig verbundenen Schichten dienen dazu, die extrahierten Merkmale in eine finale Klassifizierung oder Vorhersage für die initiale Problembetrachtung zu bringen. Die letzte Schicht des CNNs stellt die Ausgabeschicht dar, in welcher die endgültige Vorhersage des Netzwerks entnommen werden kann. Ähnlich wie bei den Pooling Layern kommt auch hier ein besonderer Algorithmus zum Tragen, meist die *Softmax* Funktion (vgl. 2.6)[4].

$$o_i = \frac{e^{z_i}}{\sum_{i=1}^{M} e^{z_i}} \tag{2.6}$$

Hier ist o_i der Softmax Output, z_i ist der output i vor der Softmax Funktion und M ist die Anzahl der verfügbaren Ausgabeneuronen ensprechend die Anzahl der Klassen [4]. Nach dieser Formel wird eine Wahrscheinlichkeitsverteilung über die Klassen gegeben.

Abschließend lässt sich sagen, dass zu Beginn die Gewichte der Schichten zufällig initialisiert werden. Während der Trainingsphase werden diese dann an den gegebenen Input via Backpropagation und einem Optimierungsalgorithmus wie dem *Stochastic Gradient Descent* angepasst. Ziel diesen iterativen Prozesses ist es, die Vorhersagen des Netzes an die tatsächlichen Daten anzupassen, indem ein Verlustfunktionswert (engl. *loss*) minimiert wird.
Aufgrund der hohen Trainingszeiten während der Optimierung eines solchen Netzes hat sich in der Literatur neben der zufälligen Initialisierung der Gewichte die Nutzung von *Pretrained Models* aufgezeigt. Bei dieser Art von Modellen wurde bereits ein Training auf einem oder mehreren Datensätzen für eine bestimmte Problemstellung ausgeführt, weshalb die Gewichte hier bereits in der Lage sind, aussagekräftige Merkmale auszuwählen. Der Vorteil dieser Modelle liegt neben der verkürzten Rechenzeit meist auch in einer besseren Vorhersagequalität im Vergleich zum randomisierten Ansatz [24][31].

3 Modell und Daten

In diesem Kapitel möchten wir die im Versuch in Kapitel 4 verwendeten Daten sowie das eingesetzte Modell vorstellen. Bei den Daten handelt es sich um keltische Münzen, welche in fünf Klassen eingeordnet wurden. Für die Modellierung wurde ein pretrained CNN, das sogenannte *ResNET-50*, verwendet (vgl. Kapitel 2.3.2).

3.1 ResNET-50

Das ResNET-50 Modell wurde im Jahr 2016 durch He et al. im Rahmen eines Papers des *IEEE Conference on Computer Vision and Pattern Recognition* vorgestellt [17]. Das vorgestellte Modell stellt eine Sonderform der aus Kapitel 2.3.2 bekannten CNNs dar. Besondere Aufmerksamkeit bekam das Modell durch den großen Erfolg bei dem ILSVRC *classification tasks* von ImageNet im Jahr 2015, wo es mitunter am besten Abschnitt und so im direkten Vergleich mit anderen *state-of-the-art* Modellen mithalten konnte [21].
In herkömmlichen CNNs wird versucht, jede Schicht dazu zu bringen, die optimale Darstellung auf dem gegebenen Input innerhalb des Convolutional Layers zu finden. Dies ist gerade in tiefen Netzwerken mit vielen Schichten mit viel Aufwand verbunden [17]. An dieser Problematik setzt das ResNET Modell durch das sogenannte *Residual Learning* an. Die Idee des Residual Learnings beruht auf einer Verbindung von nicht direkt hintereinanderliegenden Schichten, den sogenannten Shortcut Verbindungen [17]. Durch diese Verbindung wird ein Übermitteln von Informationen zwischen entfernten Schichten ermöglicht. In Abbildung 3.1 ist dies an einem Beispiel aufgezeigt. In der Grafik können wir zwei hintereinanderliegende Schichten erkennen, zwischen denen jeweils eine Aktivierungsfunktion liegt. Ganz oben ist als Ausgabe der Aktivierungsfunktion ein Input x, in der Mitte ein $F(x)$ gegeben. Gleichermaßen ist dort ein Beispiel für einen Shortcut zwischen dem oberen und der unteren Aktivierungsfunktion abgebildet. Durch diese Abkürzung ergibt sich am Ende eine Aktivierungsfunktion von $F(x) + x$. Die Abkürzungen dienen dazu, die Residuen (Differenzen) zwischen den Schichten zu erfassen und mit den Aktivierungen aus vorherigen Schichten zu kombinieren, um daraus wiederum bessere

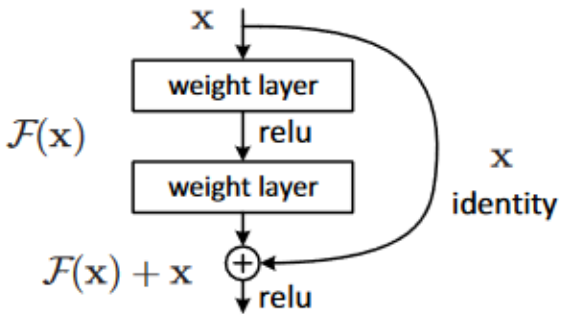


Abbildung 3.1: Beispielblock für Residual Learning [17]

Aktivierungen zu erzeugen [17][21]. Durch diese besondere Struktur können tiefere Netzwerke effizienter trainiert werden.
In dem Paper von He et al. wurden verschiedenen Architekturen und Anzahl von Schichten innerhalb miteinander im Hinblick auf deren Modellperformance verglichen. Hierbei wurden neben 18, 34, 101 und 152 Schichten auch das von uns im Versuch verwendete Modell mit 50 Layern verglichen. Es hat sich gezeigt, dass durch den Einsatz der Shortcut Verbindungen die Performance mit Steigerung der Schichten verbessern lässt, da durch den Einsatz der genannten Shortcuts ein Gradienten verschwinden sowie das Explodieren vermieden werden kann. Zusammenfassend wurde mit den Residualnetzwerken das in der Literatur bekannte Problem angegangen, dass tiefere Netzwerke nicht zwingend auch zu besseren Prognosen geführt haben (Degradierungsproblem).
Trainiert und validiert wurde das ResNET-50 Modell auf dem *ImageNet* Datensatz. Hierbei handelt es sich um eine Datenbank an Bildern und Klassen für die Bildklassifikation [33]. Insgesamt gibt es innerhalb der Datenbank 1000 Klassen mit 1,2 Millionen *training images* und 50 Tausend *test images*. Durch dieses vorangegangene Training wurden die internen Modellgewichte bereits auf das Erkennen und Erfassen von relevanten Eigenschaften in Bildern vortrainiert und liefert dadurch für unseren Versuch in Kapitel 4 eine schnellere Lösungsfindung im Finetuning auf dem Datensatz (vgl. Kapitel 2.3.2).

3.1.1 Verwendete Umsetzung

In unserer Arbeit verwenden wir das ResNET-50 Model, welches auf dem ImageNet Datensatz vortrainiert ist und können es via *keras* als *Tensorflow*-Modell laden. Tensorflow ist ein von Google im Jahr 2015 entwickeltes Machine Learning Framework für die Verwendung neuronaler Netze [1]. Angesprochen wird das Framework durch die Bibliothek *keras*, welche eine benutzerfreundliche Schnittstelle für Tensorflow bereitstellt. Entwickelt wurde dieses im Rah-

men des ONEIROS (open-ended Neuro-Electronic Intelligent Robot Operating System) durch den Google Entwickler François Chollet. Das Framework, sowie das verwendete Modell kann folgendermaßen innerhalb einer python Umgebung geladen werden:

```
1  import tensorflow as tf
2  from tensorflow import keras
3
4  resnet50 = tf.keras.applications.ResNet50(include_top = False, pooling = "avg",
   weights ='imagenet')
```

In Zeile 4 Können wir den Ladevorgang des Modells sehen. Hierbei wird mit *include_top = False* der Fully Connected Layer zu Beginn des Netzes abgeschnitten und das Modell mit den Gewichtungen auf Basis des ImageNet geladen.

3.2 Datensatz

Wie bereits in Kapitel 1 beschrieben, handelt es sich bei dem untersuchten Datensatz um Bilder von keltischen Münzen, welche in fünf Klassen eingeordnet sind. Die gefundenen Münzen stammen von Jersery Heritage und wurden bereits im Rahmen des Clarenet Projektes untersucht [18][11]. Hierbei liegen die Bilder der Münzen entweder als Vorderseiten, Rückseiten oder als Merged Bild, also Abbildung von Vorder- und Rückseite auf einem vor.

In Tabelle 3.1 können wir die Verteilung und Anzahl der Bilder über die erwähnten Kategorien sowie in den fünf Klassen sehen. Insgesamt umfasst der Datensatz 12.055 Bilder von

Category	Obverse	Reverse	Merged
Train			
I	615	611	611
II	615	603	603
III	615	609	609
IV	615	613	613
V	615	614	614
Total	3075	3050	3050
Test			
I	179	179	179
II	184	184	184
III	170	170	170
IV	238	238	250
V	185	185	185
Total	956	956	968

Tabelle 3.1: Verteilung der Bilderanzahl je Kategorie und Klassen für Trainings- und Testdatensatz

Kategorie	Obverse	Reverse	Merged
Train	224x224	256x256	256x256
Test	224x224	256x256	256x256

Tabelle 3.2: Auflösung der Bilder je Kategorie und Klassen für Trainings- und Testdatensatz (in px)

denen etwa 76% ($\widehat{=}9175$ Bildern) für das Training und entsprechend 25% ($\widehat{=}2880$ Bildern) für das Testen verwendet werden. Gleichermaßen können wir anhand der Tabelle sehen, dass wir gut balancierte Klassen vorliegen haben. Darunter versteht man, dass die Anzahl der Bilder gleichmäßig über alle fünf Klassen verteilt sind. Die vorliegenden Bilder sind im *.jpg* Format gespeichert und haben wie in Tabelle 3.2 für die Kategorien *Reverse* und *Merged* 256x256 Pixel und für *Obverse* 224x224 Pixel als Auflösung. Daraus ergibt sich für den in Kapitel 4 vorgestellten Versuch, eine Harmonisierung der Auflösung in der Datenaufbereitung. Da das im Versuch verwendete ResNET50 Modell die Auflösung 224x224 erwartet, skalieren wir entsprechend unsere beiden Kategorien herunter. In Abbildung 3.2 sind zwei Beispielbilder für die

Abbildung 3.2: Darstellung zweier Münzen der Kategorie Obverse. Links Münze der Klasse II Rechts Münze der Klasse III

Vorderseiten gegeben. Auf der linken Seite haben wir eine Münze der Klasse II und rechts eine Münze der Klasse III. An diesem Beispiel können wir sehen, dass die Objekte jeweils auf einem weißen Hintergrund fotografiert sind. Gleichermaßen sind für diese beiden Klassen bereits visuelle Unterschiede mit dem menschlichen Auge ersichtlich. Auf beiden Bildern sind Gesichter im Profil abgebildet. Auf der linken Seite ist eine Nase zu erkennen, während auf der rechten Seite ein ausgeprägterer Haarschopf zu sehen ist. In Abbildung 3.3 ist ebenfalls ein Beispiel für Bilder der Kategorie II und III gegeben. Hier können wir ebenfalls bereits mit dem menschlichen Auge Unterschiede erkennen. Die letzte Kategorie ist im Beispiel 3.4 gegeben. Hierbei sehen wir bereits, dass aufgrund der nebeneinanderliegenden Bilder, eine verhältnismäßig geringere Auflösung der einzelnen Münzseiten und damit der Sichtbarkeit der

einzelnen Merkmale vorhanden ist.

Abbildung 3.3: Darstellung zweier Münzen der Kategorie Reverse. Links Münze der Klasse II Rechts Münze der Klasse III

Abbildung 3.4: Darstellung zweier Münzen der Kategorie Merged. Links Münze der Klasse II Rechts Münze der Klasse III

4 Versuch

In diesem Kapitel stellen wir den Kern dieser Arbeit vor. Hierbei handelt es sich um ein mehrstufiges Experiment, bei welchem verschiedene Netzarchitekturen von CNNs und unterschiedlichem Input miteinander hinsichtlich ihrer Prognosegüte verglichen werden. Neben den bereits bekannten verschiedenen Datengrundlagen aus Kapitel 3.2 werden sowohl Netze mit Single- als auch mit Multiinput untersucht. Für den Versuch ergeben sich die folgenden Szenarien:

1. **Datengrundlage**: Obverse | **Netzstruktur**: Singleinput
2. **Datengrundlage**: Reverse | **Netzstruktur**: Singleinput
3. **Datengrundlage**: Merged | **Netzstruktur**: Singleinput
4. **Datengrundlage**: Obverse und Reverse | **Netzstruktur**: Multiinput

4.1 Versuchsaufbau

Der Aufbau des Versuchs unterscheidet sich in den beiden Netzstrukturen: dem Single- und Multiinput. Im Folgenden möchten wir die sich ergebenden Netzstrukturen vorstellen.

4.1.1 Singleinput

Aus der Einführung zu diesem Kapitel, sehen wir, dass die meisten Netze unter die Struktur des Singleinputs fallen. Dazu gehören die neuronalen Netze, welche die Vorderseiten, Rückseiten oder die zusammengeführten Bilder als alleinige Eingabe in das Netz erhalten. In Abbildung 4.1 ist die Struktur schematisch dargestellt. Hier sehen wir auf der linken Seite unsere drei Datengrundlagen, welche Bild für Bild an das CNN übermittelt werden. Wie in Kapitel 3.1 beschrieben, verwenden wir für unseren Versuch das vortrainierte ResNET50 Modell. Darauf folgt ein Softmax Layer, in welchem das zugeführte Bild in eine der fünf Münzkategorien eingeteilt wird.

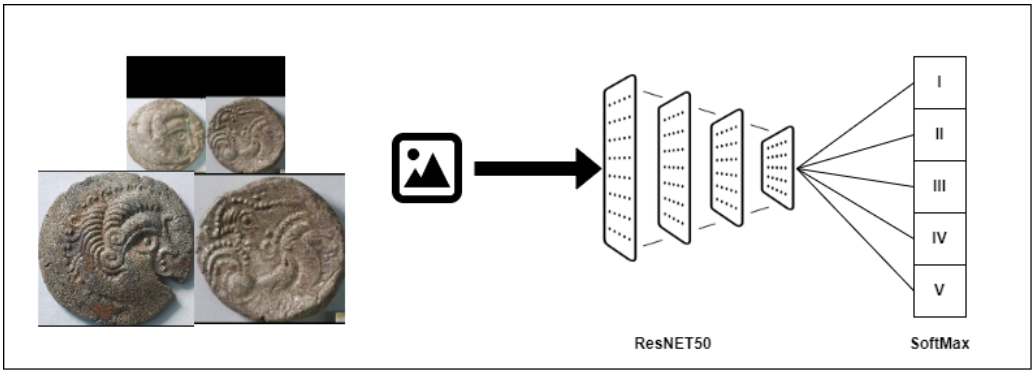


Abbildung 4.1: Architekturskizze des CNNs für Singleinput Eingaben

4.1.2 Multiinput

Neben dem Single Input betrachten wir aber auch ein mehrdimensionales Netz, was zwei Eingaben, der Vorder- und der Rückseite einer Münze, als Dual-Input erhält. In Abbildung 4.2 ist die Struktur schematisch dargestellt. Hierbei können wir ähnlich zu dem Netz aus Kapitel 4.1.1 sehen, dass auch hier wieder die Bilder einzeln an das ResNET50 Modell geliefert werden. Der Unterschied zwischen den beiden Strukturen ist, dass wir hier zwei parallelgeschaltete Modelle vorliegend haben, welche jeweils Bilder der Vorder- oder der Rückseite verarbeiten. Anschließend werden diese beiden Netze über eine Verbindungsschicht, dem sogenannten Concat Layer, miteinander verbunden. Bei dieser besonderen Art der Schicht eines neuronalen Netzes werden mehrere Eingabematrizen (Tensoren) miteinander verbunden. Als Vorteil erreichen wir eine Art der Informationsbündelung, indem wir beide Eingaben zusammenführen, ohne eine Verringerung der Münzauflösung wie beim Merged Bild zu erhalten. Durch diese Bindung der Informationen kann sich die Modellleistung verbessern.

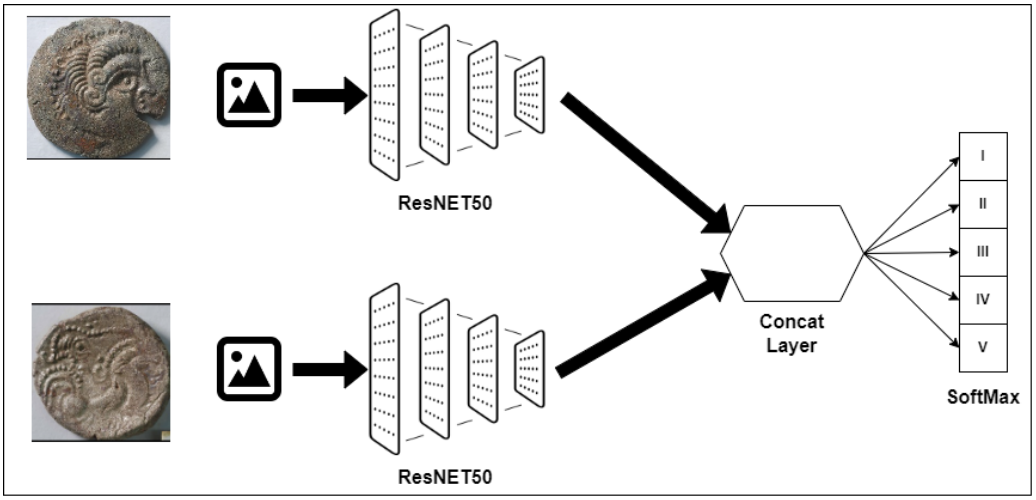


Abbildung 4.2: Architekturskizze des CNNs für Multiinput Eingaben

4.2 Versuchsdurchführung

4.2.1 Hardware

Durchgeführt wurden alle Experimente auf einem Windows 10 Pro 64Bit System. Dieser PC verfügte über den Ryzen 9 5900x Prozessor und hatte als Grafikkarte die Nvidia 1080ti mit 11 Gigabyte GDDRX5 Speicher zur Verfügung. Als Arbeitsspeicher standen dem System 32GB DDR4 Speicher (@3200 Mhz) zur Verfügung. Entwickelt wurde innerhalb einer Anaconda Umgebung in der Programmiersprache *python* (v3.10.9), unter Verwendung der Bibliotheken von tensorflow (v2.8.3).

4.2.2 Verwendete Metriken

Im Rahmen dieser Arbeit beschränken wir uns auf die gängigen Evaluierungsmetriken für Klassifikationsprobleme mit mehreren Klassen. Hierbei wird zwischen der *Accuracy*, der *Precision*, dem *Recall* und dem *F1-Score* unterschieden. Diese Metriken möchten wir dem Leser nun einmal kurz vorstellen.

4.2.2.1 Accuracy

Die erste Metrik ist die Accuracy, zu Deutsch Genauigkeit. Hierbei wird untersucht, wie viele Vorhersagen korrekt waren.

$$accuracy = \frac{\#\ richtige\ Vorhersagen}{\#\ gesamte\ Vorhersagen} \tag{4.1}$$

Der Accuracy Score wird berechnet, indem man die Anzahl der Treffer durch die Gesamtzahl der Dateninstanzen teilt. Eine höherer Accuracy Score zeigt an, dass das Modell eine höhere Genauigkeit bei der Klassifikation der Daten aufweist. Es ist jedoch wichtig zu beachten, dass der Accuracy Score in bestimmten Szenarien nicht immer die beste Metrik ist. Insbesondere wenn die Klassen im Datensatz nicht gleichmäßig verteilt sind (ungleiches Klassenverhältnis) oder wenn die Kosten für FP und FN Vorhersagen unterschiedlich sind, können andere Metriken wie Precision, Recall oder F1-Score informativer sein [42].

4.2.2.2 Precision

In einem Klassifikationsproblem mit mehreren Klassen misst die Precision-Metrik die Genauigkeit des Modells bei der Vorhersage der korrekten Klasse für jede einzelne Klasse. Die Preci-

sion ist eine der gängigen Evaluierungsmetriken, um die Leistung eines Klassifikationsmodells zu bewerten. Die Metrik wird berechnet, indem die Anzahl der korrekt vorhergesagten positiven Fälle (True Positives, *TP*) durch die Summe der korrekt vorhergesagten positiven Fälle und der falsch vorhergesagten positiven Fälle (False Positives, *FP*) geteilt wird. Mit anderen Worten misst die Precision das Verhältnis der korrekt positiven Vorhersagen zur Gesamtzahl der positiven Vorhersagen. Für eine Klasse c ergibt sich der folgende Zusammenhang:

$$Precision(c) = \frac{TP(c)}{TP(c) + FP(c)} \tag{4.2}$$

Die Precision kann einen Wert zwischen 0 und 1 haben, wobei ein Wert von eins eine perfekte Precision (100%) bedeutet, während ein Wert von 0 eine sehr schlechte Vorhersagegüte (0%) darstellt. Um einen Gesamtwert zu erhalten, werden für die Precision Scores der einzelnen Klassen ermittelt und gemittelt, um eine aggregierte Metrik zu erhalten, die die durchschnittliche Genauigkeit über alle Klassen hinweg darstellt [42]. Diese aggregierte Form wird oft als *Marco-Average Precision* bezeichnet.

4.2.2.3 Recall

Ähnlich wie die Precision verwendet auch der Recall die Anzahl der *TP*, um gebildet zu werden. Die Unterscheidung liegt hierbei im Nenner des Bruches. Hier wird kein FP betrachtet, sondern die *False Negatives FN*. Es ergibt sich der folgende Bruch für eine Klasse c.

$$Recall(c) = \frac{TP(c)}{TP(c) + FN(c)} \tag{4.3}$$

Der Wertebereich, liegt auch hier wieder zwischen 0 und 1, wo eine 1 (100%) auf einen perfekten Recall und eine 0 darauf hinweist, dass alle relevanten positiven Fälle nicht erkannt wurden. Ähnlich wie bei der Precision wird der Recall für jede einzelne Klasse in einem Multiclass-Problem berechnet. Für jede Klasse c werden die $TP(c)$ und $FN(c)$ separat gezählt und der Recall(c) entsprechend berechnet. Anschließend können die Recall-Werte für alle Klassen gemittelt werden, um einen aggregierten Recall zu erhalten, der die durchschnittliche Fähigkeit des Modells, relevante positive Fälle über alle Klassen hinweg zu erkennen, darstellt [42]. Diese aggregierte Recall-Metrik wird oft als *Macro-Average Recall* bezeichnet.

4.2.2.4 F1-Score

Die letzte Metrik bildet der F1-Score. Hierbei wird eine Metrik ermittelt, welche das ausgewogene Verhältnis zwischen Precision und Recall einer Klassifikation darstellt und zumeist gerade

bei unbalancierten Klassen Anwendung findet. Der F1-Score ist folgendermaßen definiert:

$$f1 = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall} \tag{4.4}$$

Der Wertebereich liegt zwischen -1 und 1 und berücksichtigt sowohl die FP als auch die FN. Ein hoher F1-Score zeigt an, dass das Modell sowohl eine geringe Anzahl von False Positives als auch eine geringe Anzahl von False Negatives hat und somit eine gute Balance zwischen Precision und Recall erreicht [42].

4.3 Ergebnisse

Im folgenden Kapitel möchten wir die gesammelten Ergebnisse des Versuchs darstellen. Hierfür unterscheiden wir zwischen zwei Versuchsdurchläufen. Bei dem ersten Durchlauf wurden die Daten in unveränderter Form verwendet und beim zweiten Durchlauf wurde einige Anpassung (engl. *Data Augmentation*) durchgeführt. Hierbei beginnen wir zunächst mit einer Übersicht und Erläuterung der Metriken für die einzelnen Netzstrukturen, um diese dann im zweiten Teil zusammenzuführen.

4.3.1 Versuch A: Originaler Datensatz

Metric	Obverse	Reverse	Merged	Concatenate
Accuracy	0.7762	0.8672	0.7821	0.8159
Precision	0.7876	0.8773	0.8003	0.8265
Recall	0.7878	0.8668	0.8007	0.8246
F1	0.7869	0.8652	0.7991	0.8248

Tabelle 4.1: Ergebnisse der Modelle in den Metriken *Accuracy*, *Precision*, *Recall*, *F1* für Bilder ohne Augmentation

In Tabelle 4.1 können wir die Ergebnisse der Metriken für die einzelnen Versuchsaufbauten erkennen. Hierbei wurde jeweils, die Metrik Accuracy, Precision, Recall sowie der F1-Score für die einzelnen Modelle ermittelt (vgl. Kapitel 4.2.2). Für unsere Single Input Modelle haben wir Werte im Bereich von 0.7869 - 0.8652 für den F1-Score erzielen können. Hierbei wurde das beste Ergebnis mit dem Modell erreicht, welches auf die Rückseiten (Reverse) trainiert wurde. Das schlechteste Ergebnis hingegen erreichten wir bei dem Modell der Vorderseiten, welches jedoch mit 0.7869 nur verschwindend geringer war als mit den zusammengeführten Bildern (0.7991). In Abbildung 4.3 sind die Verläufe der Loss Funktion sowie der Accuracy gegeben. Hierbei können wir erkennen, dass wir in fast allen Fällen eine schnelle asymptotische Annäherung besitzen. Lediglich für das Bild in der Mitte, dem Modell der Rückseiten, ist ein weniger

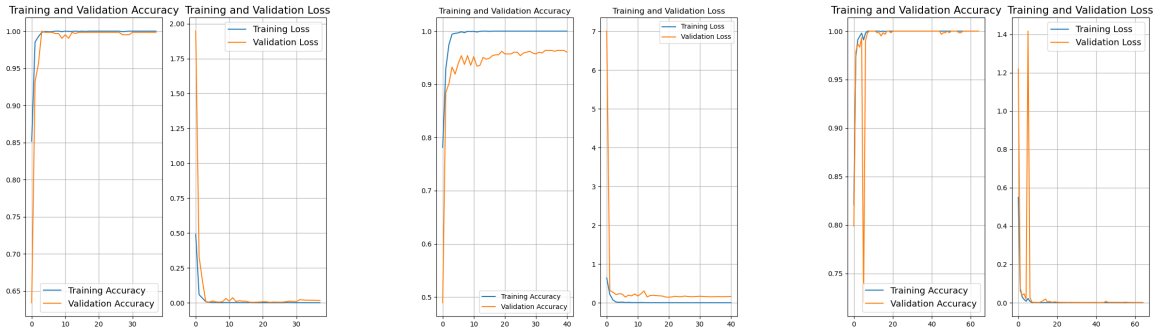


Abbildung 4.3: Trainigsgraf und Verlauf des Loss und der Accuracy für die Single Input Modelle. Orange - Training, Blau - Validierung. Links - Vorderseite | Mitte - Rückseite | Rechts - Merged

schnelles und distanzierteres Wachsen zu beobachten. Dies gilt auf den Trainingsdaten (blau) als auch für die Validierungsdaten (orange). Ebenso, haben wir uns die Vorhersagequalität der Modelle in einer Verwirrungsmatrix (engl. *confusion matrix*) betrachtet. Anhand der Darstellung im Anhang 3 können wir sehen, dass alle drei Modelle ähnlich gute Vorhersagen geliefert haben. Im linken oberen Bild, dem Modell der Vorderseiten, ist zu erkennen, dass wir für die Klassen I, II und III gute Vorhersage Ergebnisse liefern können. Lediglich für die Klassen IV und V gibt es einige Abweichungen. So wurden beispielsweise 84 Münzen als Klasse IV eingeordnet. Diese waren jedoch Elemente der Klasse V. Gleiches wurde bei 91 Münzen der Klasse IV beobachtet, obwohl diese Elemente der Klasse V waren. Für die beiden anderen Modelle des Singleinputs lässt sich ein ähnliches Verhalten ablesen. Jedoch ergibt sich für das Modell oben rechts, dem Modell der Rückseiten, für die Klassen IV und V eine deutlich geringere Abweichung von 66 für die Vorhersage von IV zu V und ein Wert von 21 für die Vorhersage für den Fall der Klasse V zu IV.

Für unser Multiinput Modell können wir der Tabelle 4.1 einen Wert von 0.8248 für den F1-Score und damit das zweitbeste Ergebnis für unseren Versuchsaufbau entnehmen. Dies gibt uns eine Antwort auf eine der Forschungsfragen, und zwar, dass das Zusammenführen der beiden CNNs im Multiinput einen deutlichen Mehrwert mit einer Steigerung von 3% - 4% gebracht hat im Hinblick zu dem Vorderseiten- beziehungsweise dem Merged Modell. In Abbildung 4.5 können wir auf der linken Seite den Verlauf des Trainings für das Dual-Input Modell sehen. Auch hier können wir ein schnelles, asymptotisches Annähern beobachten. In der rechten Hälfte sehen wir, dass verglichen zu den Single-Input Modellen, mit Ausnahme des Rückseitenmodells, die falsch vorhergesagten Münzen in allen Klassen weniger geworden sind. Jedoch hat sich auch hier ein Problem in den Klassen IV und V erfassen lassen. Für 83 Münzen wurde die Klasse IV vorhergesagt, obwohl es sich um V gehandelt hat. Gleichermaßen wurden 65 Mün-

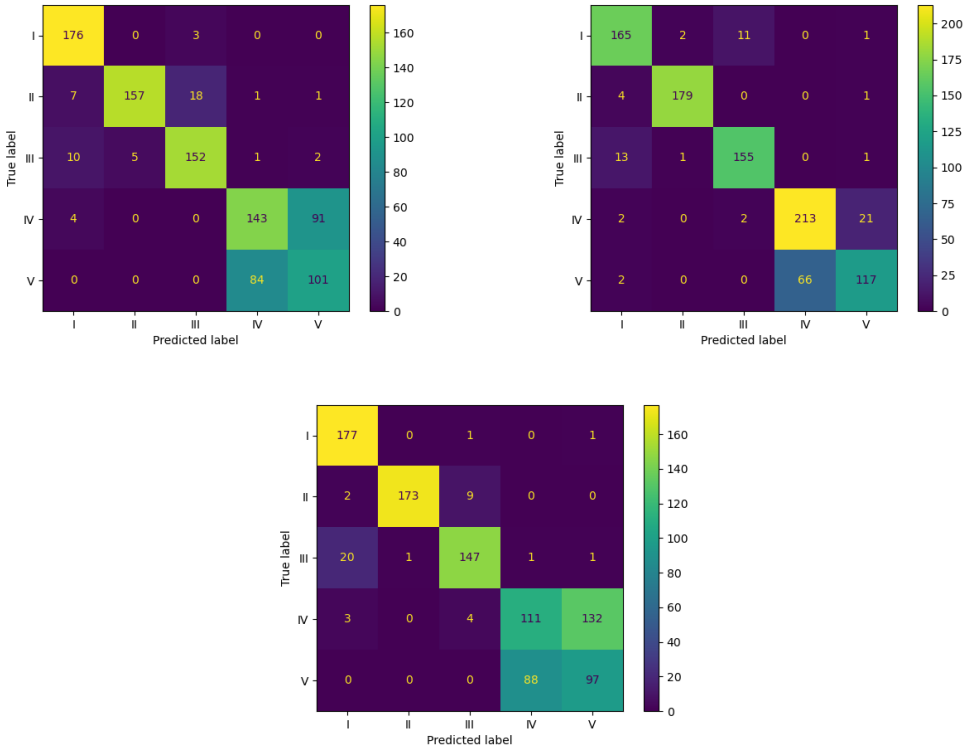


Abbildung 4.4: Confusion Matrix für Single Input Modelle.
Oben Links - Vorderseite | Oben Rechts - Rückseite | Unten - Merged

zen der Kategorie V zugewiesen, obwohl diese IV zugehörige sind. Aus diesem Grund wurde ein separates Dual-Input Modell trainiert, welches lediglich Bilder der Kategorien IV und V erhalten hat. Leider konnte hier keine Verbesserung in der Vorhersagegenauigkeit erzielt werden, sondern es war eine Verschlechterung der Werte zu beobachten. So haben sich die Werte von 83 auf 91 und von 65 auf 71 erhöht. Der interessierte Leser kann die entsprechenden Abbildungen im Anhang 5 unter Figur 1 und 2 finden.

In dem Experiment wurde neben der reinen Betrachtung der Performance der Netzstrukturen auch der Einfluss der Münzseiten auf die Vorhersage untersucht. Hierfür wurde sich an einer Visualisierungsmethode namens *GradCam* bedient, welche den Gradienten der Loss Funktion der letzten Faltungsschicht des CNNs bildet [34]. Mit diesen ermittelten Werten für die Bildbereiche der Münzen konnte eine Einfärbung vorgenommen und visuell dargestellt werden. In Abbildung 4.6 ist ein Beispiel für eine solche Visualisierung anhand einer Münze der Klasse I gegeben. Hierbei gibt die Farbe eines Bereichs die Wichtigkeit dessen an. Je wärmer (röter) die Farbe, desto wichtiger und je kälter (blauer), desto irrelevanter ist der jeweilige Bereich. In den Teilbildern (a) und (b) ist das Bild als einzelne Elemente gegeben, welches in das Dual-Input Modell gegeben wurden. Gleichermaßen stellt das Bild (c) dieselbe Münze in Merged Form und damit als Input für das Single-Model dar. Es ist hier sehr schön zu erkennen, dass die

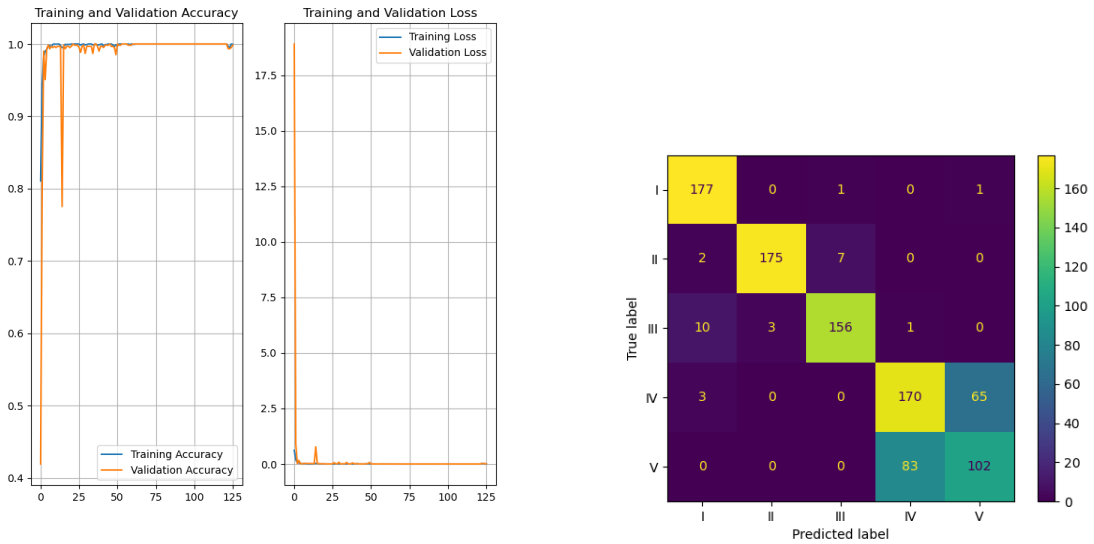


Abbildung 4.5: Trainingsverlauf und Confusion Matrix für das Dual-Input Modell

beiden Modellstrukturen unterschiedliche Merkmale als relevant identifizieren. So beschränkt sich das Single-Input Model in diesem Beispiel primär auf die rechte Hälfte, also der Rückseite der Münze, wohingegen im Dual-Input Modell Informationen beider Münzen als relevant für die Klassifikation erfasst werden. Zudem ist hier zu erkennen, dass sich auch die relevanten Bereiche der Rückseite zwischen den beiden Modellen unterscheiden. Im Bild (b) wird sich auf das Zentrum der Münze konzentriert und nimmt nach außen hin an Wichtigkeit ab. Für das Merged Bild (c) hingegen wird nahezu die komplette Münze als relevant markiert und hat damit in der Gesamtfläche eine Bedeutung für die Klassifikation. Dies lässt sich auch an weiteren Beispielen des Experiments beobachten, woraus sich auch hier ableiten lässt, dass die Modellstruktur unterschiedliche Bereiche als wichtig für die Klassifikation erkennen lässt.

Mithilfe der Gradientenanalyse von GradCam konnten wir den Informationswert eines Bildes

Class	Obverse	Reverse	Importance
All	$1.7284e-05$	$3.0516e-05$	Reverse
I	$1.6333e-05$	$2.8405e-05$	Reverse
II	$1.7986e-05$	$3.1161e-05$	Reverse
III	$1.7882e-05$	$2.7562e-05$	Reverse
IV	$1.5428e-05$	$3.2327e-05$	Reverse
V	$1.9346e-05$	$3.2305e-05$	Reverse

Tabelle 4.2: Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz

ermitteln und diese miteinander vergleichen, um Rückschlüsse über die Wichtigkeit der Münzseite im Dual-Input Modell ziehen zu können. In der Tabelle 4.2 können wir für die einzelnen

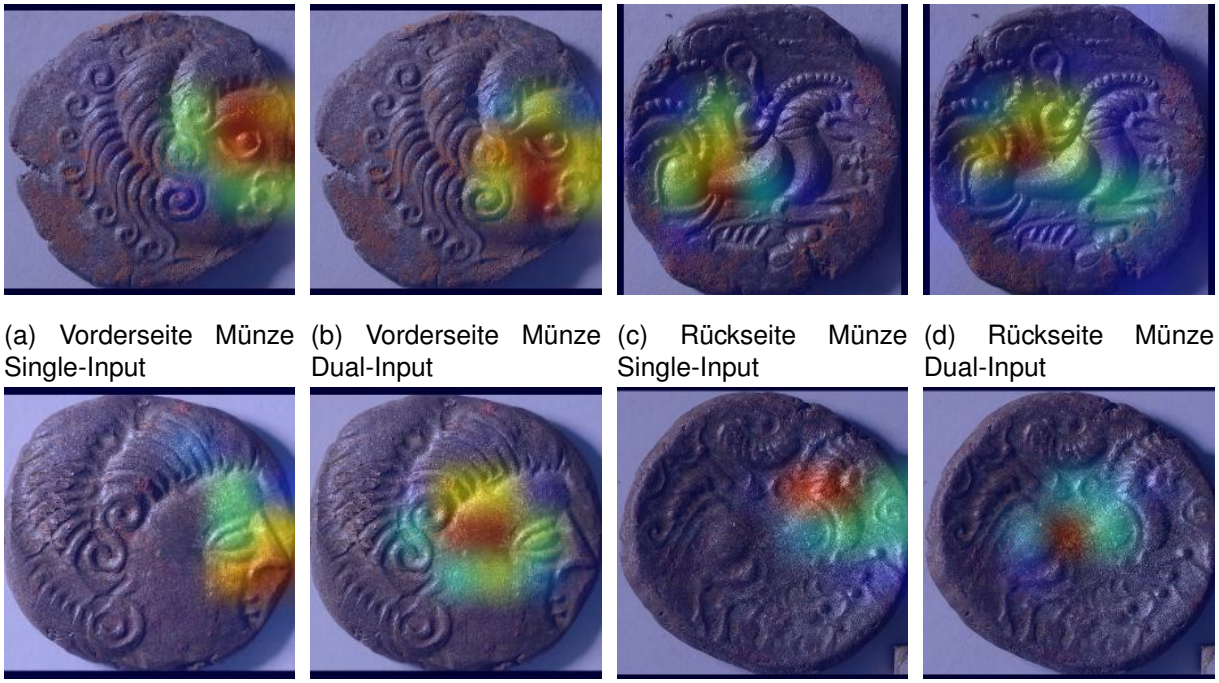
(a) Vorderseite Münze Klasse I (b) Rückseite Münze Klasse I (c) Merged Münze Klasse I

Abbildung 4.6: GradCam Visualisierung einer keltischen Münze im Vergleich zwischen Single-Input (c) und Dual-Inut Modell (a, b)

Klassen die Mittelwerte der Informationsgehälter für dir Rückseiten beziehungsweise der Vorderseiten sehen. Gleiches ist in der Zeile *All* im Makroumfeld angegeben. Es hat sich gezeigt, dass für alle Klassen sowie im Makroumfeld die Rückseite einen höheren Informationsgehalt für die Münzklassifikation nach der GradCam Analyse aufweist.
Mithilfe der GradCam Analyse war es uns außerdem möglich, die relevanten Flächen der Münzseiten, welche sich durch das Single sowie Dual-Input Modell ergeben, miteinander zu vergleichen. Ziel ist es hier zu betrachten, ob die alleinstehenden Modelle die gleichen Flächen auf einer Münzseite markiert haben wie das Concat Modell. In Abbildung 4.7 ist ein Auszug aus den Ergebnissen anhand von zwei Münzen für eben diese Betrachtung gegeben. Wir können hier sehen, dass innerhalb der Münzen die gefundenen Bereiche mit relevanten Informationen zueinander ähnlich, jedoch zueinander leicht verschoben sind. Es ergibt sich hiernach eine Vergleichbarkeit der Ergebnisse und markierten Bereiche zwischen den beiden Single-Input und des Multi-Input Modells.
Eine Besonderheit, welche sich im Rahmen des Experiments ergeben hat, ist die Art der Datenaufbereitung der zugeführten Bilder. Für die Single-Input Modelle konnten die besten Ergebnisse unter Hinzunahme der Resnet50-Preprocessing Funktion erzielt werden. Diesen Aufruf können wir im Codeauszug 4.3.2 der Zeile 4 entnehmen. Für das Multi-Input hingegen wurden die besten Ergebnisse ohne das Verwenden der Processing Funktion erzielt. Für den interessierten Leser sind die Ergebnisse der Durchläufe für das Concat Modell im Anhang 5 gegeben.

4.3.2 Versuch B: Angepasster Datensatz

In diesem Versuchsabschnitt wurden die Daten mit verschiedenen Methoden angepasst und dem Modell zugeführt. Hierbei wurde Transformationen wie Rotationen, Vergrößerungen, Spieglungen sowie Scherungen durchgeführt. In der Literatur wird diese Bearbeitung der Daten auch



(a) Vorderseite Münze Single-Input (b) Vorderseite Münze Dual-Input (c) Rückseite Münze Single-Input (d) Rückseite Münze Dual-Input

(e) Vorderseite Münze 2 Single-Input (f) Vorderseite Münze 2 Dual-Input (g) Rückseite Münze 2 Single-Input (h) Rückseite Münze 2 Dual-Input

Abbildung 4.7: Vergleich zwischen relevanten Informationsbereichen auf zwei Münzen mit GradCam

als Datenerweiterung (engl. Data Augmentation) beschrieben [37]. Diese Art der Anpassung kann verschiedene Vorteile für die Modellierung liefern. So ergibt sich beispielsweise durch das Spiegeln oder Drehen des Bildes ein größerer Datensatz, welcher für Trainingszwecke verwendet werden kann. Ein weiterer Vorteil besteht darin, dass eine verbesserte Robustheit des Modells gegenüber Variationen und Verzerrungen in realen Szenarien erreicht wird. Durch die Transformation und Erzeugung von Variation in der Datengrundlage wird gleichermaßen das Risiko des Overfittings reduziert. Dies begründet sich darin, dass durch die künstlich erzeugten Unterschiede ein Rauschen eingefügt wird und das Modell davon abgehalten wird, sich zu sehr auf spezifische Merkmale oder Muster in den Trainingsdaten zu verlassen. Dadurch wird das neuronale Netz gezwungen, generalisiertere Repräsentationen zu lernen. Ein weiterer Vorteil der Augmentation liegt in dem Behandeln von Klassenungleichgewichten, also einer nicht homogenen Verteilung von Objekten entlang der Klassen, da hier durch die künstliche Datenerzeugung genau diesem Zustand entgegen gewirkt werden kann.

Im Folgenden betrachten wir die Ergebnisse der Versuchsreihe mit Augmentation. Die Bearbeitung der Bilder wurde mithilfe der *ImageDataGenerator* Klasse von tensorflow umgesetzt.

```
1  from keras.preprocessing.image import ImageDataGenerator
2
```

```
    data_generator = ImageDataGenerator(
        preprocessing_function=tf.keras.applications.resnet50.preprocess_input,
        validation_split=0.2,
        rotation_range=45,
        zoom_range=0.1,
        horizontal_flip=True,
        shear_range=0.2)
```

Listing 4.1: Beispiel DataAugmentation für ImageDataGenerator Klasse

Aus dem obigen Pythoncode können wir sehen, dass wie zu Anfang beschrieben vier Transformationen genutzt wurden. Neben einer Rotation von bis zu $45°$, einem Zoom von 0.1 welcher sowohl eine Vergrößerung als auch eine Verkleinerung erzeugt. Die anderen beiden Transformationen behandeln eine Spiegelung an der y-Achse, sowie einer Scherung. In Tabelle 4.3 können wir die Ergebnisse aus dem Versuchsdurchlauf für unsere Metriken und den Modellstrukturen erkennen. Auch hier ergibt sich ein ähnliches Bild wie im Versuch aus Kapitel 4.3.1.

Metric	Obverse	Reverse	Merged	Concatenate
accuracy	0.773	0.8598	0.7758	0.8159
precision	0.7859	0.8743	0.7877	0.8263
recall	0.7881	0.8577	0.7892	0.8251
f1	0.7861	0.8587	0.7883	0.825

Tabelle 4.3: Ergebnisse der Modelle in den Metriken *accuracy*, *precision*, *recall*, *f1* für Bilder mit Augmentation

Für unsere Single Input Modelle ergab sich wieder im Modell für die Rückseite das beste Ergebnis mit 0.8587 im F1-Score, während die beiden anderen Modelle ähnliche Werte für die Metrik ergaben. Auch im Multi-Input Modell wurde durch die Transformation der Bilder keine Verbesserung erzielt. Für die Informationswerte, welche aus der GradCam Analyse gefolgert sind, wurde ebenso festgestellt, dass die Rückseite die meiste Information enthält. Eine entsprechende Auswertung ist im Anhang 5 gegeben.
Neben den vier Transformationen wurde gleichermaßen auch geprüft, ob weniger Augmentation einen Mehrwert liefern können. Hierfür wurde lediglich eine Spiegelung an der y-Achse verwendet. Auch hier hat sich keine Verbesserung der Metriken oder Wichtigkeit der Rückseite erkennen lassen. Die Ergebnisse sind im ebenfalls im Anhang 5 zu finden. Für beide Versuchsdurchläufe, mit transformierten Bildeingaben, hat sich jedoch der Unterschied zwischen den Informationsgehältern der jeweiligen Münzseiten im Vergleich zum Versuch ohne Transformation verringert.

(a) Vorderseite der Münze (b) Rückseite der Münze (c) Münze als Merged

Abbildung 4.8: GradCam Visualisierung einer keltischen Münze im Vergleich zwischen Singleinput (c) und Multiinut Modell (a, b)

4.3.3 Zusammenfassung

Zusammenfassend lässt sich aus dem Versuch ableiten, dass für die Klassifikation der keltischen Münzen die Rückseite als wichtigster Input zu werten ist. Hier hat sich gezeigt, dass für alle Versuchsszenarien das Single-Input Modell der Rückseite am besten performt hat. Gleichermaßen konnte durch die Betrachtung der durch GradCam markierten Flächen erkannt werden, dass sowohl auf den Münzseiten der Single-Input als auch bei dem Multi-Input ähnliche Flächen als markant und damit wichtig für die Klassifikation identifiziert wurden (vgl. Abbildung 4.7) und zudem über alle Versuche hinweg die Rückseite mehr Information für Klassifikation der Münze liefern konnte. Darüber hinaus hat sich durch den Versuch gezeigt, dass das Verbinden der beiden Münzseiten in einem Modell mittels des Concat Layer ein Mehrwert für Klassifikationsgüte gegenüber des Merged Bildes liefern lässt. Dies begründet sich zudem auch darin, dass wir das Auftreten von markierten Flächen in den schwarzen Bereichen, wie sie im Merged Modell auftreten, verringert haben (siehe Abbildung 4.8). Anhand der Grafik können wir sehen, dass für dieselbe Münze im Multi-Input Modell Flächen auf der Münze und beim Merged Modell Flächen außerhalb der Münze als relevant markiert wurden. Somit konnten wir für unseren Datensatz, das Markieren von schwarzen Bereichen der zusammengeführten Münzseiten auf einem Bild durch das Betrachten der Münzen als einzelne Eingaben im Multi-Input Modell verhindern. Mit dieser Erkenntnis knüpfen wir an ein bekanntes Forschungsproblem an, welches bereits in anderen Forschungsprojekten für Bilder mit zusammengefügten Münzseiten beobachtet werden konnte [13].

5 Fazit

Das Klassifizieren von antiken Münzen stellt eine anspruchsvolle Herausforderung sowie eine große Anstrengung für die Numismatiker dar [5]. Im Rahmen dieser Arbeit haben wir anhand eines Datensatzes von keltischen Münzen verschiedene Strukturen von neuronalen Netzen miteinander im Kontext zu deren Klassifikationsgüte verglichen. Hierbei wurde insbesondere der Zusammenhang untersucht, ob das Zuführen beider Münzseiten einen Vorteil liefert. Aus dem in dieser Arbeit beschriebenen Versuch konnten wir sehen, dass die besten Vorhersagen mit dem Singleinput Modell, welches auf die Rückseiten trainiert ist, erreicht wurde. Gleichermaßen hat sich jedoch das Multi-Input Modell, welches Vorder- und Rückseiten in zwei parallelgeschaltenen CNNs über den Concat Layer verbindet, als eine vielversprechende Alternative erwiesen, da wir für unseren Versuch damit die zweitbesten Ergebnisse beobachten konnten. Darüber hinaus konnte mithilfe des Multiinput Modells unter Hinzunahme der GradCam Analyse beobachtet werden, dass für jeden Versuchsdurchlauf die Rückseite einen höheren Informationsgehalt für die jeweilige Klassifikation geliefert hat, was ebenfalls unsere Beobachtung hinsichtlich der Modellgüte bei dem Rückseitenmodell als beste Methode bestätigt. Des Weiteren hat sich gezeigt, dass alle Modellstrukturen die meisten Probleme bei der Vorhersage zwischen Klasse IV und V hatten, was sich mittels eines individuellen Trainings auf eben diesen nicht korrigieren ließ (vgl. Kapitel 4.3.1). Innerhalb des zweiten Versuchsdurchlaufs konnte beobachtet werden, dass die verwendeten Data Augmentation Methoden auf unserem Datensatz keine nachweisbaren Vorteile in der Vorhersagequalität geliefert haben. Hier hat sich jedoch gezeigt, dass der Informationswert der Rückseiten gegenüber des Informationswertes der Vorderseite zurückgegangen ist (vgl. Tabelle 1 und Tabelle 3). Bezogen auf den Forschungsgegenstand dieser Arbeit, ob ein Mutliinput Modell mittels Concat Layer einen Mehrwert liefert, hat sich hier klar ein Vorteil gezeigt, wenn die Bilder nicht nebeneinander in einem, sondern als zwei voneinander getrennte Inputs in das neuronale Netz geliefert werden. Es hat sich über alle Versuche hinweg eine Verbesserung von dem Merged Modell gegenüber dem Concat Modell von 3% bis 6% im F1-Score zeigen lassen. Ein weiterer Vorteil des Multiinput Modells gegenüber dem Merged Ansatz ist, wie in Kapitel 4.3.3 beschrieben, die Reduzierung der Fälle, in welchen das Modell Teile des schwarzen Bereiches als wichtig identifiziert, da hier deutlich weniger schwarze Hintergrundfläche auf den Bildern vorhanden ist.

Abschließend lässt sich somit sagen, dass die Zusammenführung von Informationen mittels Concat Layer eine vielversprechende Alternative zu den Single-Input Modellen darstellt. Für anschließende Forschung wäre es interessant, eine weitere Optimierung des Multiinput Modells zu betrachten. Hierbei könnten Optimierungsmethoden wie Dropout Schichten, sowie dem Einfrieren von Layern innerhalb des verwendeten CNNs, um ein Transfer Learning zu anzuwenden und ein Anpassen der Gewichte der vorgelagerten Schichten zu vermeiden [36] tiefergehend betrachtet werden.

Literatur

[1] Martín Abadi u. a. „Tensorflow: A system for large-scale machine learning“. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, S. 265–283.

[2] Hervé Abdi, Dominique Valentin und Betty Edelman. *Neural networks*. 124. Sage, 1999.

[3] Abien Fred Agarap. „Deep learning using rectified linear units (relu)“. In: *arXiv preprint arXiv:1803.08375* (2018).

[4] Saad Albawi, Tareq Abed Mohammed und Saad Al-Zawi. „Understanding of a convolutional neural network“. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, S. 1–6.

[5] Hafeez Anwar u. a. „Ancient coin classification using reverse motif recognition: Image-based classification of roman republican coins“. In: *IEEE Signal Processing Magazine* 32.4 (2015), S. 64–74.

[6] Kai Arulkumaran u. a. „Deep reinforcement learning: A brief survey“. In: *IEEE Signal Processing Magazine* 34.6 (2017), S. 26–38.

[7] Raouf Boutaba u. a. „A comprehensive survey on machine learning for networking: evolution, applications and research opportunities“. In: *Journal of Internet Services and Applications* 9.1 (2018), S. 1–99.

[8] Rich Caruana und Alexandru Niculescu-Mizil. „An empirical comparison of supervised learning algorithms“. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, S. 161–168.

[9] M Emre Celebi und Kemal Aydin. *Unsupervised learning algorithms*. Bd. 9. Springer, 2016.

[10] Tianqi Chen und Carlos Guestrin. „Xgboost: A scalable tree boosting system“. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, S. 785–794.

[11] ClaReNet. *Münzkataloge und der OCC – Pfadabhängigkeiten und digital agency*. URL: https://clarenet.hypotheses.org/ (besucht am 24. 06. 2023).

[12] Pádraig Cunningham, Matthieu Cord und Sarah Jane Delany. „Supervised learning“. In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Springer, 2008, S. 21–49.

[13] IR on-coin datasets. *Image Recognition approach on the Corpus Nummorum dataset*. URL: https://github.com/Frankfurt-BigDataLab/IR-on-coin-datasets (besucht am 24. 06. 2023).

[14] Peter Dayan, Maneesh Sahani und Grégoire Deback. „Unsupervised learning“. In: *The MIT encyclopedia of the cognitive sciences* (1999), S. 857–859.

[15] Arden Dertat. *Applied Deep Learning - Part 1: Artificial Neural Networks*. URL: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6 (besucht am 04. 06. 2023).

[16] Zoubin Ghahramani. „Unsupervised learning“. In: *Summer school on machine learning*. Springer, 2003, S. 72–112.

[17] Kaiming He u. a. „Deep residual learning for image recognition“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, S. 770–778.

[18] Jersey Heritage. *Celtic Coin Collection*. URL: https://catalogue.jerseyheritage.org/categories/celtic-coin-hoard/ (besucht am 24. 06. 2023).

[19] John J Hopfield. „Artificial neural networks“. In: *IEEE Circuits and Devices Magazine* 4.5 (1988), S. 3–10.

[20] Xiaowei Huang, Gaojie Jin und Wenjie Ruan. „Machine Learning Basics“. In: *Machine Learning Safety*. Springer, 2012, S. 3–13.

[21] Yi Yao Huang und William Yang Wang. „Deep residual learning for weakly-supervised relation extraction“. In: *arXiv preprint arXiv:1707.08866* (2017).

[22] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://de.mathworks.com/discovery/reinforcement-learning.html.

[23] H Jabbar und Rafiqul Zaman Khan. „Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)“. In: *Computer Science, Communication and Instrumentation Devices* 70.10.3850 (2015), S. 978–981.

[24] Brady Kieffer u. a. „Convolutional neural networks for histopathology image classification: Training vs. using pre-trained networks“. In: *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2017, S. 1–6.

[25] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas u. a. „Supervised machine learning: A review of classification techniques“. In: *Emerging artificial intelligence applications in computer engineering* 160.1 (2007), S. 3–24.

[26] Zewen Li u. a. „A survey of convolutional neural networks: analysis, applications, and prospects“. In: *IEEE transactions on neural networks and learning systems* (2021).

[27] Bing Liu und Bing Liu. „Supervised learning“. In: *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data* (2011), S. 63–132.

[28] Dengsheng Lu und Qihao Weng. „A survey of image classification methods and techniques for improving classification performance“. In: *International journal of Remote sensing* 28.5 (2007), S. 823–870.

[29] Vladimir Nasteski. „An overview of the supervised machine learning methods“. In: *Horizons. b* 4 (2017), S. 51–62.

[30] Keiron O'Shea und Ryan Nash. „An introduction to convolutional neural networks“. In: *arXiv preprint arXiv:1511.08458* (2015).

[31] Ç F Özgenel und A Gönenç Sorguç. „Performance comparison of pretrained convolutional neural networks on crack detection in buildings“. In: *Isarc. proceedings of the international symposium on automation and robotics in construction*. Bd. 35. IAARC Publications. 2018, S. 1–8.

[32] Gopinath Rebala u. a. „Machine learning definition and basics“. In: *An introduction to machine learning* (2019), S. 1–17.

[33] Tal Ridnik u. a. „Imagenet-21k pretraining for the masses“. In: *arXiv preprint arXiv:2104.10972* (2021).

[34] Ramprasaath R Selvaraju u. a. „Grad-cam: Visual explanations from deep networks via gradient-based localization“. In: *Proceedings of the IEEE international conference on computer vision*. 2017, S. 618–626.

[35] John Shalf. „The future of computing beyond Moore's Law“. In: *Philosophical Transactions of the Royal Society A* 378.2166 (2020), S. 20190061.

[36] Hoo-Chang Shin u. a. „Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning“. In: *IEEE transactions on medical imaging* 35.5 (2016), S. 1285–1298.

[37] Connor Shorten und Taghi M Khoshgoftaar. „A survey on image data augmentation for deep learning“. In: *Journal of big data* 6.1 (2019), S. 1–48.

[38] Ray J Solomonoff. „Machine learning-past and future“. In: *Dartmouth, NH, July* (2006).

[39] Eric V Strobl und Shyam Visweswaran. „Deep multiple kernel learning“. In: *2013 12th International Conference on Machine Learning and Applications*. Bd. 1. IEEE. 2013, S. 414–417.

[40] TechVidvan. *Supervised Learning Algorithm in Machine Learning*. URL: https://techvidvan.com/tutorials/supervised-learning/ (besucht am 20. 10. 2020).

[41] Robert E Uhrig. „Introduction to artificial neural networks“. In: *Proceedings of IECON'95-21st Annual Conference on IEEE Industrial Electronics*. Bd. 1. IEEE. 1995, S. 33–37.

[42] Ž Vujović u. a. „Classification model evaluation metrics“. In: *International Journal of Advanced Computer Science and Applications* 12.6 (2021), S. 599–606.

[43] Daniel Westreich, Justin Lessler und Michele Jonsson Funk. „Propensity score estimation: neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression“. In: *Journal of clinical epidemiology* 63.8 (2010), S. 826–833.

[44] XGBoost. *Introduction to Boosted Trees*. URL: https://xgboost.readthedocs.io/en/latest/tutorials/model.html (besucht am 01. 06. 2023).

[45] Daniela Xhemali, Chris J Hinde und Roger Stone. „Naïve bayes vs. decision trees vs. neural networks in the classification of training web pages“. In: (2009).

[46] Bayya Yegnanarayana. *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[47] Xue Ying. „An overview of overfitting and its solutions“. In: *Journal of physics: Conference series*. Bd. 1168. IOP Publishing. 2019, S. 022022.

[48] Guoqiang Peter Zhang. „Neural networks for classification: a survey“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.4 (2000), S. 451–462.

[49] Ling Zhang und Bo Zhang. „A geometrical representation of McCulloch-Pitts neural model and its applications“. In: *IEEE Transactions on Neural Networks* 10.4 (1999), S. 925–929.

[50] Jinming Zou, Yi Han und Sung-Sau So. „Overview of artificial neural networks“. In: *Artificial neural networks: methods and applications* (2009), S. 14–22.

Appendices

Anhang: Grafische Ergebnisse des Multiinput Modells für die Klassen IV und V

In diesem Teil des Anhangs sind die Trainingskurven sowie die Verwirrungsmatrix gegeben. Hierbei wurde das Modell betrachtet, welches nur auf die Klassen IV und V trainiert wurde.

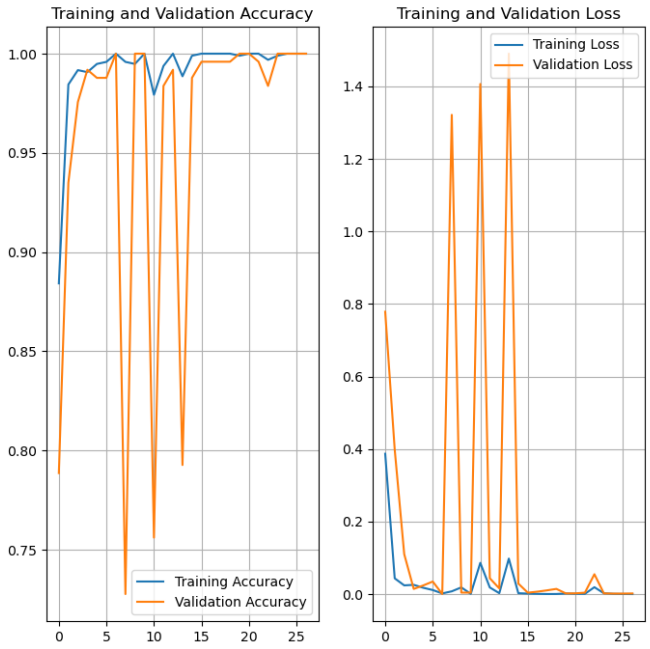


Abbildung 1: Trainingsverlaufskurven für die Accuracy und den Loss für das Dual-Input Modell auf den Klassen IV und V

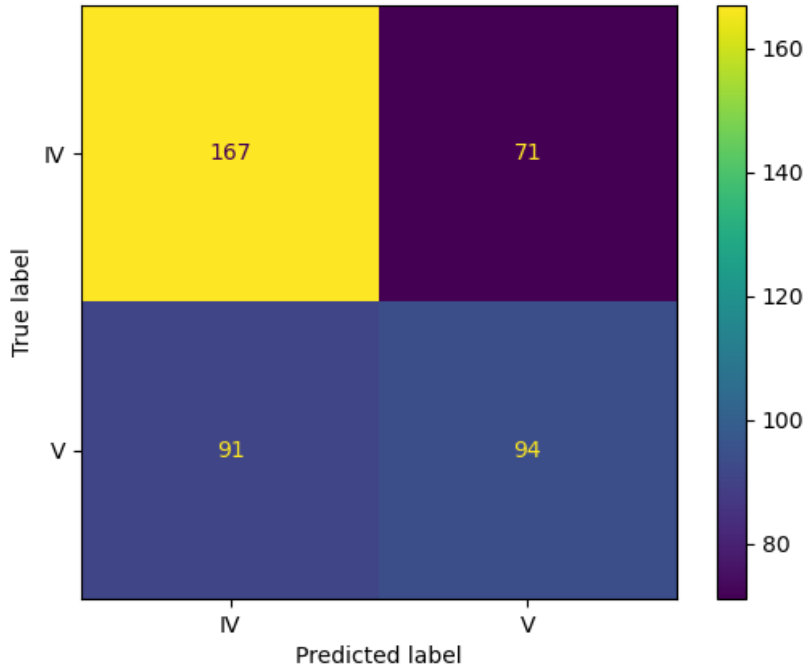


Abbildung 2: Confusion Matrix für das Dual-Input Modell auf den Klassen IV und V

Anhang: Ergebnisse des Versuches B

In diesem Anhang sind die Ergebnisse aus dem Versuchsdurchlauf mit Augmentation gegeben. Hierbei sind die Verwirrungsmatrizen (Abbildung 3 sowie die Übersicht der Informationsgehälter je Münzseite und Klasse gegeben (Abbildung 1).

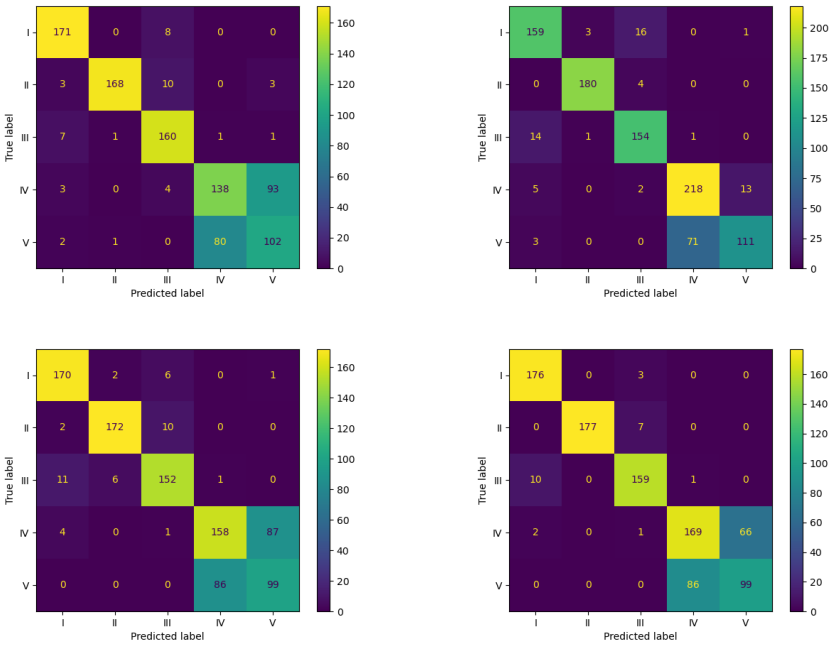


Abbildung 3: Confusion Matrix für Single- und Multi-Input Modelle auf Datensatz mit Augmentation.
Oben Links - Vorderseite | Oben Rechts - Rückseite
Unten Links - Merged | Unten Rechts - Concatenate

class	Obverse	Reverse	Importance
All	$2.3649e-05$	$2.8704e-05$	Reverse
I	$2.0919e-05$	$2.8773e-05$	Reverse
II	$2.1783e-05$	$3.0685e-05$	Reverse
III	$2.6947e-05$	$3.3396e-05$	Reverse
IV	$2.201e-05$	$2.4396e-05$	Reverse
V	$2.7226e-05$	$2.7898e-05$	Reverse

Tabelle 1: Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz mit Augumentation

Anhang: Ergebnisse des Versuches B(2)

In diesem Kapitel des Anahngs betrachten wir die Ergebnisse des Versuchsdurchlaufs mit reduzierter Augmentation. Hierbei sind die Metriken in Tabelle 2, sowie die Verwirrungsmatrizen in Abbildung 4 gegeben. Gleichermaßen ist auch hier wieder eine Übersicht der Informationswerte der Münzseiten mit Tabelle 3.

Metric	Obverse	Reverse	Merged	Concatenate
accuracy	0.7594	0.8536	0.7293	0.8002
precision	0.7744	0.862	0.7499	0.8065
recall	0.7755	0.8544	0.7516	0.8114
f1	0.7736	0.8555	0.7486	0.8082

Tabelle 2: Ergebnisse der Modelle in den Metriken *accuracy*, *precision*, *recall*, *f1* für Bilder mit reduzierter Augumentation

class	avg_front_score	avg_back_score	Importance
All	$2.2965e-05$	$2.6728e-05$	Reverse
I	$2.6813e-05$	$2.8846e-05$	Reverse
II	$1.8338e-05$	$2.2284e-05$	Reverse
III	$2.632e-05$	$3.0991e-05$	Reverse
IV	$2.1269e-05$	$2.576e-05$	Reverse
V	$2.2945e-05$	$2.6427e-05$	Reverse

Tabelle 3: Übersicht über die gemittelten Informationswerte je Klasse und Münzseite sowie für den vollständigen Datensatz mit reduzierter Augumentation

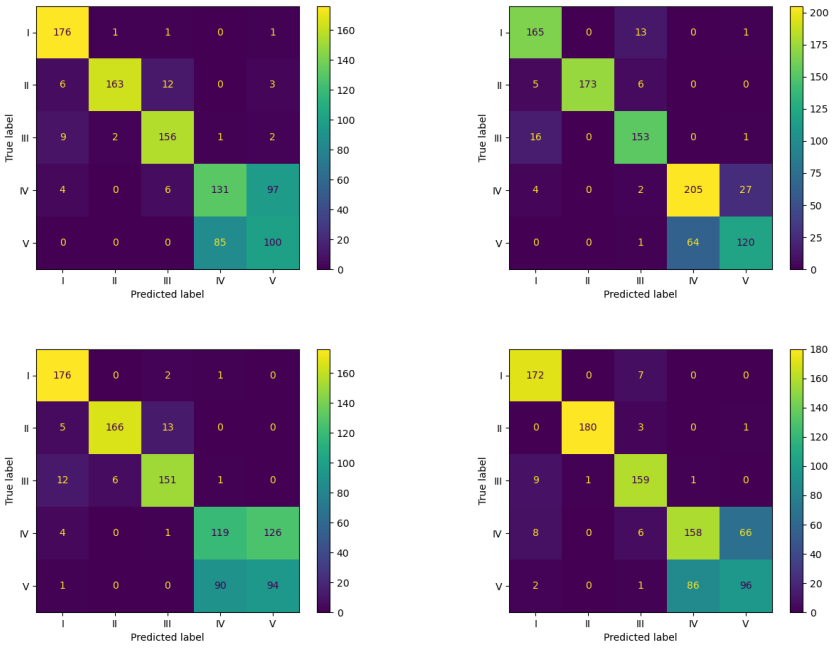


Abbildung 4: Confusion Matrix für Single- und Multi-Input Modelle auf Datensatz mit reduzierter Augumentation. Oben Links - Vorderseite | Oben Rechts - Rückseite
Unten Links - Merged | Unten Rechts - Concatenate

Anhang: Übersicht der Concat Modelle mit Preprocess Funktion

In diesem Kapitel des Anhangs sind die Ergebnisse der Metriken für den Versuchsdurchlauf des Concat Modelle mit Preprocess Funktion gegeben (vgl. Tabelle 4). Hierbei stellt Versuch A den Versuch ohne Augmentation, Versuch B(1) den Versuch mit Augmentation und Versuch B(2) den Versuch mit reduzierter Augmentation dar. In Abbildung 5 sind die Verwirrungsmatrizen für diese Durchläufe gegeben.

	Versuch A	**Versuch B(1)**	**Versuch B(2)**
accuracy	0.6506	0.7354	0.6621
precision	0.7392	0.7556	0.6999
recall	0.6623	0.7391	0.6751
f1	0.6834	0.7375	0.6667

Tabelle 4: Übersicht über die ermittelten Metriken zu den Concat Modellen unter Hinzunahme der Preprocess Funktion.

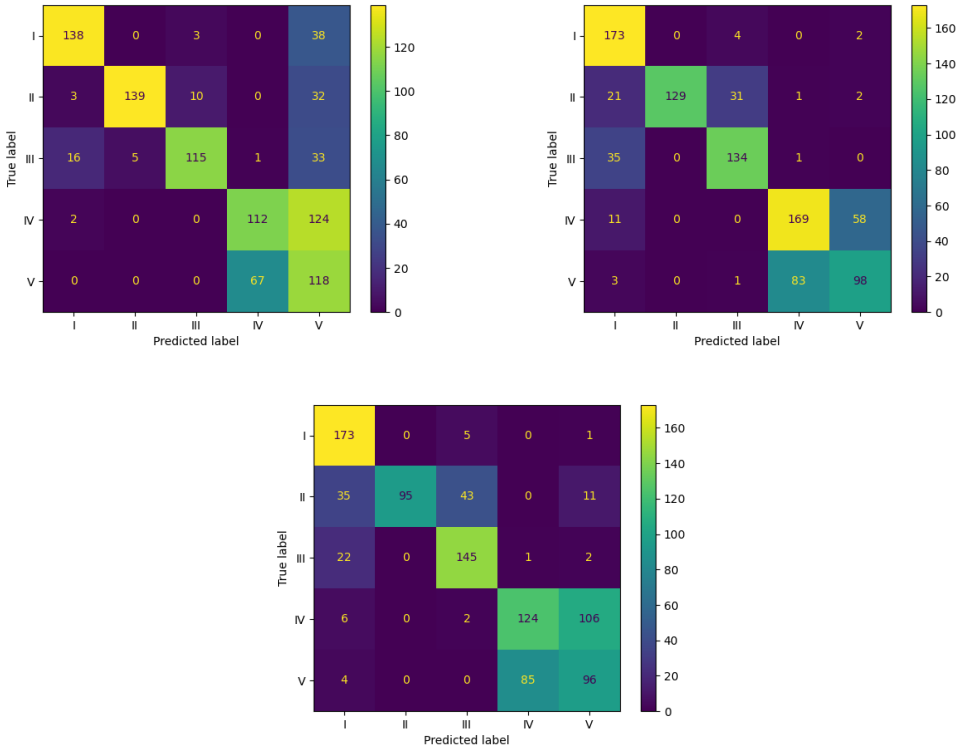


Abbildung 5: Confusion Matrix für Mutli-Input Modelle mit Preprocess Funktion.
Oben Links - Versuch A | Oben Rechts - Versuch B(1) | Unten - Versuch B(2)