HAW HAMBURG

BACHELORARBEIT

Facial Expression Recognition mittels verschiedener CNNs im Vergleich mit anschließender Robustheitsprüfung

vorgelegt am 22. November 2022

Yannic Kühn
[redacted]

Erstprüferin: Prof. Dr. Larissa Putzar
Zweitprüfer: Thorben Ortmann

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Medientechnik
Finkenau 35
20081 Hamburg

Stichworte:

Facial Expression Recognition, Künstliche Neuronale Netze, CNN, ResNet, Bildmanipulation, Robustheit

Kurzzusammenfassung:

In dieser Arbeit wird Facial Recognition behandelt: eine Technik, die es Computern erlauben soll Emotionen im menschlichen Gesicht zu erkennen. Zu diesem Zweck werden unterschiedliche CNNs untersucht und ein Modell erstellt, welches am FER2013-Datensatz trainiert wird. In einer anschließenden Reihe an unterschiedlichen Bildmanipulationen wird dieses Modell auf seine Robustheit überprüft. Dazu wird das entsprechende Netz auf den manipulierten Daten neu getestet und neu trainiert. In einer anschließenden Gegenüberstellung der so entstehenden Modelle wird ausgewertet, welche Manipulationen der Facial Expression Recognition schaden.

Keywords:

Facial Expression Recognition, Artificial Neural Networks, CNN, ResNet, Image Manipulation, robustness

Abstract:

This thesis deals with Facial Expression Recognition: a technique, that allows computers to recognize different emotions in the human face. To accomplish this, different CNNs will be put to the test and a model is created that is trained on the FER2013-Dataset. Following this: several different image manipulations will be used on the FER2013-Dataset and the model's robustness will be tested. The model will be tested and trained again on the manipulated version of the dataset. In a subsequent comparison it will be analyzed which manipulations hinder Facial Expression Recognition.

Inhaltsverzeichnis

1 Einleitung 1
1.1 Zielsetzung 1
1.2 Gliederung 1
2 Begrifflichkeiten und Definitionen 2
2.1 Künstliche Neuronale Netzwerke 2
2.1.1 Activation Functions 3
2.1.2 Forward Pass 4
2.1.3 Loss Funktion 4
2.1.4 Back Propagation 4
2.1.5 Learning Rate / Scheduler 5
2.1.6 Epochs / Steps und Batch Size 5
2.1.7 Optimizers 5
2.1.8 Accuracy 6
2.1.9 Over-/ Underfitting 6
2.2 Convolutional Neural Networks 6
2.2.1 Convolutional Layers 7
2.2.2 Pooling Layers 8
2.2.3 Fully-connected Layers 8
2.2.4 Dropout Layers 8
2.2.5 Loss Layer 8
2.2.6 Zusammensetzung von CNNs mit Beispielen 8
2.2.7 Fortschritte in CNNs in den letzten 30 Jahren 10
2.3 Facial Emotion Recognition 10
3. Erstellen eines KNNs zur Klassifizierung des FER2013-Datensatzes 13
3.1 Problemstellung 13
3.2 Ablauf 13
3.3 Versuchsaufbau des Trainings 13
3.4 Aufbau des Netzes 14
3.5 Datensätze 15
3.5.1 FER2013-Datensatz 15
3.5.2 ImageNet-Datensatz 15
3.6 Tools und Bibliotheken 15
3.7 Vergleich verschiedener Learning Rate / Scheduler 16
3.8 Training mit verschiedenen CNNs 17

3.8.1 VGG16 17
3.8.2 ResNet50 18
3.9 Vergleich der Ergebnisse 21
4. Manipulation der Daten und Robustheitsprüfung des Netzes 22
4.1 Ablauf 22
4.2 Manipulation von FER2013 23
4.3 Testen des Basismodells auf manipulierten Testdaten 25
4.4 Trainieren mit manipulierten Daten 28
4.5 Testen von manipulierten Modellen 28
4.5.1 Testen des Eye-Occlusion-Modells 29
4.5.2 Testen des Mouth-Occlusion-Modells 33
4.5.3 Testen des Salt-and-Pepper-Noise-Modells 36
4.6 Auswertung zur Stabilität 38
5 Fazit 40
Literaturverzeichnis 41
Abbildungsverzeichnis 44
ANHANG 45
Eigenständigkeitserklärung 48

1 Einleitung

Seit Computer für die breite Bevölkerung zugängig sind, ist eine der größten Herausforderungen für viele, wie man sich am besten mit ihnen verständigt. Ob per Tastatur, Maus oder Touchscreen, der Computer geht nur auf das ein, was wir ihm ausdrücklich als Befehl mitteilen. So denken Menschen jedoch nicht. Eine Konversation zwischen zwei Menschen geht weit über das hinaus was gesagt wird. Gestik, Mimik oder Stimmlage spielen ebenso stark eine Rolle wie die gesprochenen Worte. Naheliegend wäre es demnach, eine Möglichkeit zu erschaffen bei der ein Computer nicht nur präzise unseren Befehlen folgt, sondern auch auf unsere Emotionen eingeht. Der Forschungsbereich, der sich um die Bewältigung dieser Aufgabe kümmert, wird Affective Computing genannt. Er umfasst mehrere Gebiete. Einer dieser Teilbereiche des Affective Computings nennt sich Facial Expression Recognition oder kurz FER. Er befasst sich mit der Aufgabe Emotionen in menschlichen Gesichtern zu erkennen, um einem Computer die Möglichkeit zu verleihen die Stimmungslage seines Benutzers einzuschätzen und eventuell anders auf sonst gleiche Befehle zu reagieren. In der vorliegenden Arbeit wird mit Hilfe von Convolutional Neural Networks versucht dieses Problem zu lösen und anschließend analysiert, wie robust diese Lösungen unter dem Einfluss verschiedener Störungen sind.

1.1 Zielsetzung

Ziel dieser Arbeit ist es verschiedene Algorithmen zu erstellen, die in der Lage sind Facial Expression Recognition durchzuführen, sie zu bewerten und auf ihre Robustheit zu testen. Zu diesem Zweck werden Neuronale Netzwerke verwendet, die mit verschiedenen Netzwerkarchitekturen auf dem FER2013 Datensatz trainiert werden sollen. Eine Manipulation der Trainings- und Testdaten soll im Rahmen einer Robustheitsanalyse zeigen, inwiefern die Netze unter der Verwendung dieser Daten leiden und ob die entstehenden Modelle trotz Manipulation noch in der Lage sind, Emotionen in Gesichtern zu klassifizieren.

1.2 Gliederung

In Kapitel 2 werden zunächst Begrifflichkeiten und Definitionen erklärt, die für das Verständnis der Arbeit wichtig sind. In Kapitel 3 folgt das Trainieren eigener Modelle auf dem FER2013 Datensatz, und es werden verschiedene Netzwerkarchitekturen getestet. Das Hauptaugenmerk liegt auf dem Vergleich verschiedener Learning Rate Scheduler und der Gegenüberstellung verschiedener CNNs. Das am besten abschneidende Modell wird anschließend in Kapitel 4 einer Robustheitsprüfung unterzogen. Zu diesem Zweck werden im ersten Teil Manipulationen des FER2013 Datensatzes durchgeführt. Im Folgenden wird das erarbeitete Modell auf diese manipulierten Daten getestet. Weiterhin wird in Kapitel 4 beschrieben, wie das getestete Netz neu auf den manipulierten Daten trainiert. Ziel ist zu überprüfen, wie sich diese neuen Modelle von dem ursprünglichen Modell (Kapitel 3) unterscheiden. Abschließend werden die Ergebnisse von Kapitel 3 und 4 in einem Fazit zusammengefasst.

2 Begrifflichkeiten und Definitionen

In diesem Kapitel werden Grundlagen und Definitionen erklärt, die für das Verständnis von Facial Expression Recognition und Neuronalen Netzen notwendig sind.

2.1 Künstliche Neuronale Netzwerke

Künstliche Neuronale Netze (Netzwerke), kurz KNN, sind Strukturen, die von Nervensystemen in der Natur inspiriert sind. Sie werden oft in Bereichen des maschinellen Lernens verwendet, da mit ihnen Aufgaben zu bewältigen sind, welche mit klassischen Algorithmen sehr schwer oder gar nicht lösbar sind. KNNs setzen sich aus Schichten (engl. Layern) zusammen, die wiederum aus einzelnen Neuronen bestehen. Im Allgemeinen haben KNNs ein Input-Layer, ein Output-Layer und dazwischen liegende, sogenannte Hidden-Layer. [14, Seite 18f] In Abbildung 1 ist der mögliche Aufbau eines solchen Netzes dargestellt.

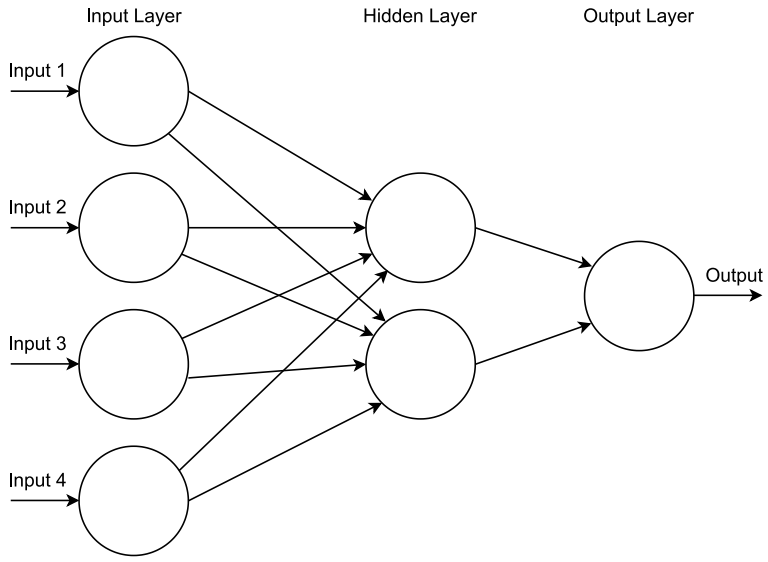

Abbildung 1: Darstellung eines KNNs (adaptiert nach der Grafik in [5] Seite 2.)

Jedes dieser Neuronen hat eine Aktivierungsfunktion (2.1.1), 1-n Inputs und einen Outputwert, welcher an nachfolgende Neuronen weitergeben wird. Die Verbindung der Neuronen hat außerdem ein Gewicht (engl. Weight), welches die Wichtigkeit dieser Verbindung darstellt. Außerdem können Neuronen noch einen sogenannten Bias besitzen, einen Wert, der dem Ergebnis der Aktivierungsfunktion hinzuaddiert wird. Mit Hilfe des Bias kann der Output des Neurons nach oben oder unten verschoben werden [14, Seite 19f]. Abbildung 2 zeigt ein solches Neuron.

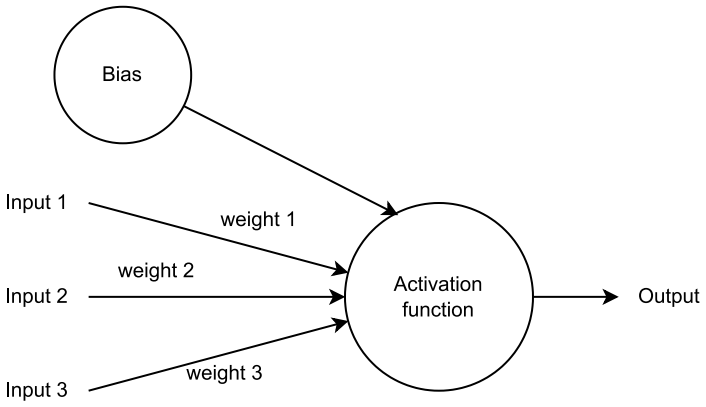

Abbildung 2: Darstellung eines Neurons (eigene Grafik.)

Es gibt verschiedene Arten von Problemen, die man mit KNNs lösen kann. Diese Arbeit dreht sich ausschließlich um Klassifikation, also um die Einordnung eines Inputs (in diesem Fall: Bilder) in eine bestimmte Klasse. Ein klassisches Beispiel hierfür wäre beispielsweise die Klassifizierung eines auf einem Bild dargestellten Tieres in seine entsprechende Rasse oder Gattung.

Bei Klassifikationen ist entscheidend, dass das letzte Layer (Output-Layer) eine bestimmte Form hat. Dieses Layer sollte eine Dimension haben, die der Anzahl der möglichen Klassen entspricht, in welche eingeordnet werden soll.

2.1.1 Activation Functions

In einer Aktivierungsfunktion (activation function) wird beschrieben, was ein Neuron mit den übergebenen Input-Daten machen soll. Es gibt mehrere verschiedene Funktionen die hierfür Anwendung finden. Oft verwendete Aktivierungsfunktionen sind: Sigmoid, tanh, ReLU und Linear. [14, Seite 21]

Bei der Sigmoidfunktion (siehe Formel 1) wird der Input zu einem Wert zwischen 1 und 0 transformiert. Sehr Große Werte werden zu 1 und sehr kleine Werte werden zu 0. [14, Seite 21]

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Ähnlich verhält sich tanh (Tangens Hyperbolicus) (siehe Formel 2). Der Hauptunterschied zu Sigmoid ist, dass tanh einen Bereich von -1 bis 1 abdeckt und somit auch Werte im negativen Bereich repräsentieren kann. [14, Seite 21]

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2}$$

Die Lineare Funktion, auch bekannt als Identitätsfunktion (siehe Formel 3), hingegen gibt einfach den übergebenen Eingabewert zurück. Diese Funktion wird z.B. in ResNet verwendet, um den Eingabewert eines Residual Blocks zu übergeben. Mehr dazu in Abschnitt 2.2.6.

$$f(x) = x \tag{3}$$

Die ReLU-Funktion (Rectified Linear Unit) (siehe Formel 4) ist ähnlich zur Identitätsfunktion, sie gibt positive Eingabewerte ohne Änderung zurück. Negative Werte werden, anders als bei Linear, als 0 zurückgegeben. [14, Seite 21]

$$f(x) = \max(0, x) \tag{4}$$

Bei der Softmax-Funktion (Formel 5), auch normalisierte exponential Funktion genannt, sind $x_1, x_2, \ldots, x_N$ die Inputwerte des Layers. Der Outputwert $f(x_i)$ steht für die Wahrscheinlichkeit, dass der Input in die i-te Kategorie gehört. [20]

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_i}} (i = 1,2, \ldots, N) \tag{5}$$

Sie wird oft für das letzte Layer verwendet, welches in die zu klassifizierenden Klassen einteilt.

2.1.2 Forward Pass

Zu Beginn eines Trainingsprozesses werden sowohl Weights als auch Biases zufällig gewählt und dann im Laufe des Trainings per Back Propagation angepasst (siehe Abschnitt 2.1.4), um die Vorhersagen des Modells bzw. des KNNs langsam zum gewünschten Ergebnis zu führen. Die Weights werden aus einem vorher bestimmten Intervall ausgewählt. Die Biases werden z.B. als Biasneuron eingeführt, welches allen Neuronen einen bestimmten Wert, den Bias, hinzuaddiert. [14, Seite 20]

2.1.3 Loss Funktion

In einer Loss Funktion wird der Fehler des Modells errechnet, also die Abweichung der errechneten Werten von denen in den Trainingsdaten vorgegebenen Werten berechnet. Der Durchschnitt der Lossfunktionen wird „Cost-Function“ genannt [14, Seite 23].

2.1.4 Back Propagation

„Die Ergebnisse der Cost-Funktion, also die Abweichung der errechneten Werte von den Werten in den Trainingsdaten werden rückwärts durch das neuronale Netz gespeist, bis alle Neuronen in den Hidden Layers eine Fraktion des Fehlerwerts in Abhängigkeit von ihrem Beitrag zum Output erhalten haben. Dieser Vorgang nennt sich Backpropagation.“ [14, Seite 23]

Nun ist es die Aufgabe eines Optimierungsalgorithmus, die Weights für den nächsten Forward Pass anzupassen. Dafür kommen unterschiedliche Algorithmen, die sich grob in zwei Kategorien einteilen lassen: derivate-free oder gradientenbasierte Optimierung. In der Praxis werden für tiefere Netze heute fast ausschließlich gradientenbasierte Verfahren angewendet, da sie schneller sind. Einer dieser Algorithmen ist der Stochastik Gradient Descent, bei dem sich dem globalen Minimum der Kostenfunktion angenähert wird. Das bedeutet, es wird der Punkt mit dem kleinsten Fehlerwert gesucht. Diese Rate der Anpassung wird als Learning Rate bezeichnet. genannt. [14, Seite 23]

2.1.5 Learning Rate / Scheduler

Die Learning Rate beeinflusst die Schrittgröße, mit der sich diesem Minimum angenähert wird. Modelle mit einer kleinen Learning Rate lernen langsamer, aber präziser, da die Wahrscheinlichkeit bei der Anwendung von kleinen Schritten deutlich geringer ist, das Ziel stark zu überschreiten. Daher ist es oft sinnvoll, zu Beginn des Trainings mit einer größeren Learning Rate zu starten um schnell eine grobe Lösung zu finden, und im weiteren Trainingsverlauf das Ergebnis durch eine die abnehmende Learning Rate verfeinern. [14, Seite 24]

In [4] stellen die Autoren mehrere Möglichkeiten vor, wie, die Learning Rate während des Trainings zu verringert werden kann; die angewandten Algorithmen nennt man Learning Rate Scheduler. Dieses soll an vier Beispielen gezeigt werden:

(1) Ein Constant Learning Rate Scheduler ist die einfachste Art, um die Learning Rate zu reduzieren. Dabei wird die Learning Rate in einem vorher festgelegten Abstand um einen vorher festgelegten Wert reduziert.

(2) Mit dem Verfahren des Coisine Annealing wird die Learning Rate erst schnell reduziert und dann wieder erhöht. Es gibt zwei Varianten dieser Technik: Bei dem „Warm restart“ werden „gute“ Weights als Startpunkt genommen, bei dem „Cold restart“ nicht. [15]

(3) Die RLRP (Reduce Learning Rate on Plateau) verfolgt einen anderen Ansatz: hier wird die Learning Rate anhand des Trainingsverlaufes reduziert. Die Learning Rate nimmt immer dann ab, wenn ein Wert beim Training stagniert. Dieser Wert kann vorher festgelegt werden. Eine Möglichkeit hierfür wäre der Validation-Loss des Modells. In Experimenten erhalten die Autoren [4], mittels RLRP das beste Ergebnis beim FER2013 Datensatz.

(4) Beim Exponential Decay wird die Learning Rate jede Epoche, oder alle X steps, mit einem Faktor < 1 multipliziert. Beim Exponential Decay wird die Learning Rate jeder Epoche, oder alle X steps, mit einem Faktor < 1 multipliziert. Dadurch nimmt die Learning Rate in den ersten Epochen schnell, um dann im Verlauf des Trainings immer langsamer kleiner zu werden. Dadurch nimmt die Learning Rate in den ersten Epochen schnell ab, um dann im weiteren Verlauf des Trainings immer langsamer kleiner zu werden. [18]

2.1.6 Epochs / Steps und Batch Size

Eine andere Einstellung beim Training eines Neuronalen Netzes ist die Batch Size. Diese gibt an wie viele Trainingsdaten gleichzeitig durch das Netz geschleust werden. Hat ein Batch ein Netz durchlaufen, spricht man von einem Step, und von einer Epoche, wenn alle Daten das Netz einmal durchlaufen haben. Die Größe der Batch Size kann Auswirkungen auf die Performance eines Modells haben. [14]

2.1.7 Optimizers

Neben dem in 2.1.4 erwähnten Optimizer SGD gibt es noch andere Algorithmen. Der Adam Optimizer wird sehr häufig angewendet, und auch für alle Versuchen der vorliegenden Arbeit gewählt. Wie auch SGD handelt es sich bei Adam um einen auf

Gradienten basierenden Optimierungsalgorithmus. Ein Vorteil von Adam ist seine geringen Speicheranforderung. [16]

2.1.8 Accuracy

Mit dem Wert Accuracy wird die Genauigkeit eines Modells bei der Klassifikation eines Datensatzes gemessen. Sie wird wie folgt errechnet (siehe Formel 6):

$$Accuracy = \frac{Anzahl\ der\ korekkten\ Vorhersagen}{Gesamtanzahl\ der\ Vorhersagen} \quad (6)$$

Beim Training von Modellen wird oft zwischen der Test-Accuracy und der Validation-Accuracy unterschieden. Damit sind die Genauigkeiten, bei der Klassifikation des Test- und des Validierungsdatensatzes gemeint. [28, Seite 213]

2.1.9 Over-/ Underfitting

Eines der Hauptziele beim Trainieren eines Modells ist, sowohl Over- als auch Underfitting zu vermeiden. Unter Underfitting versteht man, dass das Netz nicht genug aus den Trainingsdaten gelernt hat. Es ist dem Netz in diesem Fall weder möglich, die Trainingsdaten noch die Testdaten zu approximieren. Um ein Underfitting zu vermeiden könnte man beispielsweise das Netz länger trainieren lassen oder seine Komplexität erhöhen.

Beim Overfitting hingegen lernt das Netz zu „nah“ an den Trainingsdaten indem es diese auswendig lernt. Es wird danach die Trainingsdaten sehr sehr gut klassifizieren, können, aber Testdaten oder andere Daten nicht gut erkennen.

Während es beim Underfitting helfen kann, die Komplexität des Netzes zu erhöhen oder mit einer höheren Epochenzahl zu trainieren, helfen beim Overfitting oft gegenteilige Ansätze. Ein Netz kann z.B. auch zu lange trainieren indem es die gleichen Daten zu oft angeguckt und zu viel über sie lernt, und dadurch ähnliche, aber nicht gleiche Daten zunehmend schlechter erkennt. [14, Seite 25]

Es gibt mehrere Möglichkeiten Overfitting zu verhindern. Die drei am häufigsten verwendeten Lösungen sind L1 Regularisation, L2 Regularisation und Drop Out Regularisation. Bei der L1 Regularisation werden einige Weights auf 0 gesetzt, bei L2 auf einen Wert nahe 0. Durch beide Vorgänge verringert sich die Komplexität, da bestimmte Verbindungen weniger Einfluss haben. Beide Methoden haben Vor- und Nachteile: Bei L1 entstehen sehr einfache Modelle, die komplexe Muster nur schlecht erkennen, L2 hingegen ist anfälliger für starke Abweichungen in Teilen der Trainingsdaten ist. [14, Seite 26]

Drop Out Regularisation wird in Abschnitt 2.2.4 genauer erläutert.

2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) sind ähnlich zu klassischen Neuronalen Netzen. Auch sie bestehen aus Neuronen, die sich im Lernprozess selbst optimieren. Neuronen erhalten einen Input und verarbeiten diesen auf unterschiedliche Weise. Im letzten Layer werden auch hier die Klassen abgebildet zwischen denen klassifiziert werden soll.

Der Hauptunterschied zu klassischen Neuronalen Netzen ist, dass CNNs hauptsächlich für Mustererkennung in Bildern verwendet werden, da sie sich dafür besonders gut eignen. Sie brauchen wesentlich weniger Layer und Parameter als andere klassische Varianten von Neuronalen Netzen indem sie in Bildern bestimmte Features herausheben und an diesen lernen können.

Um dies zu erreichen, werden CNNs in drei fundamentale Bestandteile unterteilt: Convolutional Layers, pooling Layers und fully-connected Layers.

Außerdem braucht ein CNN ein Input Layer in der Form des zu verarbeitenden Bildes, sowie ein Output Layer passend zu den zu klassifizierenden Klassen. [5]

2.2.1 Convolutional Layers

Ein Convolutional Layer verwendet NxN große Matrizen, auch Kernels oder Konvolutionsmatrizen genannt, um Features in Bildern hervorzuheben.

Diese Kernels sind meist klein, beispielsweise 3x3 Pixel, und werden Schritt für Schritt über das Bild geschoben. Bei jeden dieser Schritte wird ein Skalarprodukt aus den Werten des Kernels und denen des Bildes berechnet. Diese werden anschließend addiert, wodurch sich für das mittlere Pixel eine von Kernel gewichtete Summe von sich und den umgebenden Pixeln ergibt. Das Ergebnis wird festgehalten und der Kernel bewegt sich weiter.

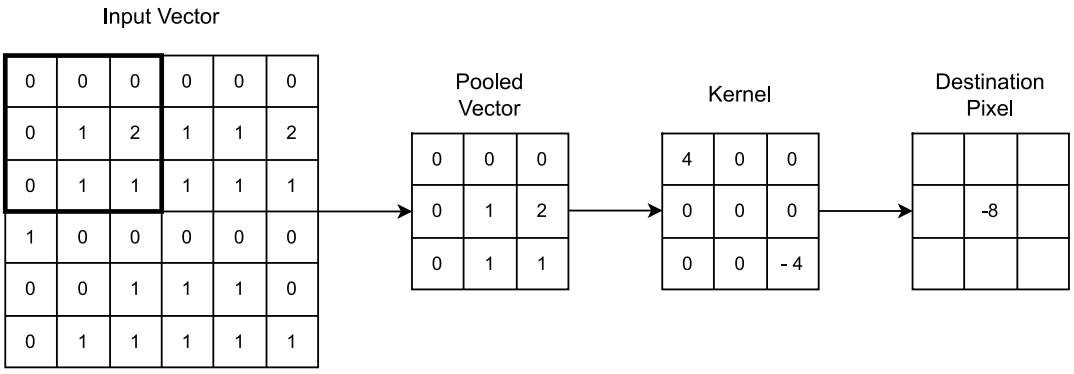

Abbildung 3: *Visuelle Repräsentation einer Konvolution, der eingerahmte Bereich entspricht des dem Input.Vektors wird mit einem Kernel zu einem Destination Pixel umgewandelt [5, Seite 6]*

In Abbildung 3 sieht man eine Repräsentation für eine Convolution. Der im ersten Teil der Abbildung eingerahmte Bereich, wird im Zuge der Convolution zu einem Pixel umgewandelt. Im letzten Teil (Destination Pixel) der Abbildung ist derselbe Bereich, welcher im ersten Teil eingerahmt ist (Input Vector). Das entstandene Pixel würde also die Mitte des Rahmens (im Input Vector eine 1) ersetzen. Das Ergebnis dieses Prozesses wird Activation Map genannt.

Anders als bei klassischen KNN Layern (fully-connected Layer) sind Neuronen in Convolutional-Layern nur mit einer kleinen Region des Input-Layers verknüpft. Die Ausmaße dieser Region bezeichnet man oft als „receptive field size“. Dadurch hat ein Neuron in diesen Layern wesentlich weniger Weights, die beim Training angepasst werden müssen, und kann viel schneller trainiert werden.

Da bei dieser Berechnung das Input-Bild kleiner wird, ist es manchmal nötig sogenanntes Zero Padding anzuwenden. Da bei dieser Berechnung das Input-Bild kleiner

wird, ist es manchmal nötig, ein sogenanntes Zero Padding anzuwenden. Bei diesem Verfahren wird außen eine oder mehrere Reihen von Nullen als Pixel eingefügt. Wenn dann Konvolutionen über das Bild laufen, werden nur diese Nullen abgeschnitten und das Bild behält seine Originalgröße. [5, Seite 5ff]

2.2.2 Pooling Layers

In Pooling Layern ,wird die Dimensionalität und damit die Nummer der anzupassenden Parameter des Models reduziert. Sie werden oft nach einem Convolutional-Layer eingesetzt.Bei den meisten CNNs wird hier das sogenannte „max-pooling-layer“ verwendet. Hier läuft ein 2x2 großer Kernel in zwei Pixel Schritten über das Bild. Mit Hilfe der MAX-Funktion wird nur der größte Wert der Pixel festgehalten. Es ergibt sich am Ende ein Bild, welches nur noch 25% der Größe des Ursprungsbildes beinhaltet. Es ist auch möglich, dass der Kernel eine 3x3 Dimension hat. Das wird als „overlapping pooling“ bezeichnet. [5, Seite 8]

2.2.3 Fully-connected Layers

Ein fully-connected Layer ist ein Layer, in dem jedes Neuron mit den Neuronen der beiden „Nachbar Layer“ verbunden ist. Diese werden ebenfalls in klassischen KNNs verwendet. [5, Seite 8]

2.2.4 Dropout Layers

“Ein Dropout Layer ist eine einfache Technik, um Overfitting in KNNs zu reduzieren. Ein Dropout Layer hat nur einen Parameter: Die Dropoutrate, welche die Chance beschreibt, mit der Verbindungen im Netzwerk fallengelassen werden. Wenn die Dropoutrate z.B. 0,5 beträgt, werden im Training 50% der Verbindungen des vorhergehenden und nachfolgenden Layers verworfen.“ [7] Dadurch muss sich das Netz bei jedem Trainingsdurchlauf anpassen und das Overfitting wird reduziert.

2.2.5 Loss Layer

“Loss Layer berechnen den gesamten Verlust und Fehler zwischen dem gegebenen und gewünschten Output.“ [6] Das bedeutet, ein Loss Layer prüft, wie gut ein Modell eine bestimmte Aufgabe erfüllt. Anhand dieser Ergebnisse wird das Modell beim Training angepasst.

2.2.6 Zusammensetzung von CNNs mit Beispielen

Aus den oben beschriebenen Layern kann nun ein CNN zusammengesetzt werden. Dafür gibt es unendlich viele Möglichkeiten. Hier ein paar Beispiele:

LeNet:

Von Yann LeCun et al. im Jahr 1989 vorgestellt, ist LeNet (Abbildung 4) eins der ersten CNNs und wurde dafür verwendet, handschriftliche Ziffern in 32x32 Graustufenbildern zu unterscheiden. im Jahr 1989 vorgestellt, ist LeNet eines der ersten CNNs und wurde dafür verwendet, handschriftliche Ziffern in 32x32 Graustufenbildern zu unterscheiden. Es besteht aus 7 Layern,welche Convolutional Layer, Pooling Layer, Fully Connected Layer und ein Output-Layer beinhalten. [6, 24, 8]

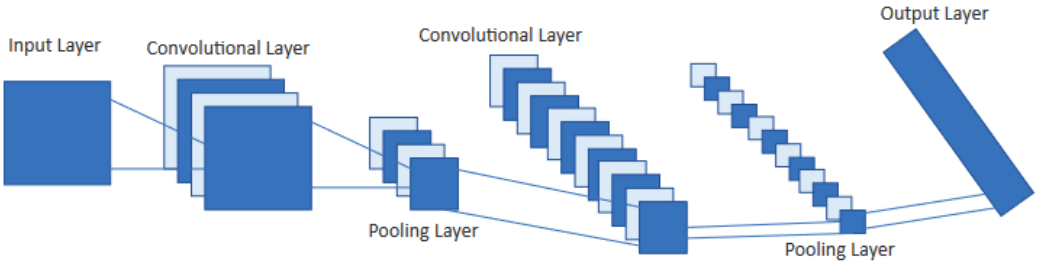

Abbildung 4: LeNet-1 [24, Seite 71]

VGG:

Das in [12] vorgestellte CNN VGG (Visual Geometry Group) fällt vor allem durch seine Simplizität auf. Diese Architektur nutzt kleine Konvolutionskernals (3x3) und verbindet sie mit Max Pooling und Fully Connected Layern. Daraus werden Blöcke gebildet, die aus zwei bis drei Konvolutionslayern und einem Max Pooling Layer bestehen.

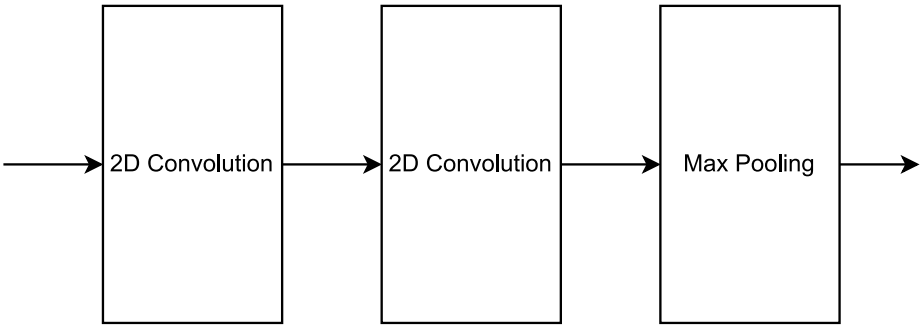

Abbildung 5: VGG Architektur. Adaptiert nach einem Ausschnitt der Grafik 3 [10, Seite 4]

Es gibt mehrere Versionen von VGG-Netzen. Zwei Beispiele sind VGG 16 und VGG19, die sich durch ihre Anzahl der Konvolutionslayer unterscheiden. VGG16 beinhaltet 16 und VGG19 19 solcher Layer.

Die VGG-Architektur (Abbildung 5) hat ein Problem, wenn die Tiefe zu hoch wird: Durch die Funktionsweise der Back Propagation (s.o.) wird bei einer hohen Tiefe des Modells, und durch die fortlaufende Multiplikation der Gradienten die Updates der oberen Layer nur sehr langsam laufen. Dadurch erhöht sich die Trainingszeit, und bei tiefen Modellen sinkt die Genauigkeit anstatt zu steigen [10, 12]. Dieses Problem wird „Vanishing Gradient“ genannt. [14, Seite 22] In [10] wird hierzu eine Lösung angeboten. Das daraus entstehende CNN nennt sich ResNet.

ResNet:

In [10] beschreiben die Autoren ihre Lösung für das Vanishing Gradient Problem [14, Seite 22], um tiefere CNNs zu ermöglichen und so eine höhere Klassifikationsgenauigkeit zu erreichen. Dafür verwenden sie sogenannte „Residual Masking“ Blöcke als Bausteine für ihr CNN. Bei diesen wird über die Identitätsfunktion der Input eines Blockes zum Output addiert. Wenn wir also von einem Mapping von G(x) ausgehen, bei dem G für das Mapping des gesamten Blockes steht, ergibt sich so eine Funktion (siehe Formel 7):

$$G(x) = F(x) + x \tag{7}$$

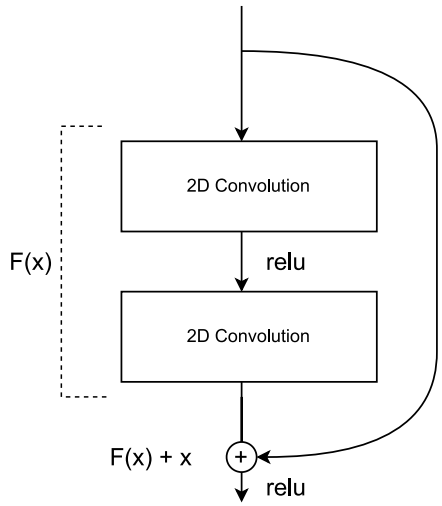

Abbildung 6: Residual Block. Adaptiert nach Grafik 2 in [10]

F steht in Abbildung 6 für die Funktion des Blockes, dem zur Vervollständigung der Input x hinzuaddiert wird. Hierdurch entsteht ein lokaler Gradient von 1. Da der lokale Gradient bei der Back Propagation durch das Modell gleichbleibt, wurde das Vanishing Gradient Problem gelöst. Dies führt dazu, dass auch Modelle mit deutlich mehr Layern besser lernen können, als bei VGG. In [10] zeigen He et al., dass sogar Modelle mit 152 Layern noch eine Verbesserung im Vergleich zu 101 Layern darstellen. Allerdings erreicht auch dieses Verfahren seine Grenzen. In 1202 Layern erreicht das Netz eine schlechtere Testgenauigkeit als beim 152-Layer Netz [10, Tabelle 6]. In ihrer Arbeit argumentieren die Autoren, dass dieses vermutlich auf Overfitting zurückzuführen ist, da bei beiden Netzen der Trainings-Loss vergleichbar ist.

2.2.7 Fortschritte in CNNs in den letzten 30 Jahren

Seit CNNs in den 1990ger Jahren vorgestellt wurden haben sie ein großes Potential in der Bildverarbeitung gezeigt. Hauptsächlich limitiert durch fehlende Rechenpower und unzureichende Datensätze, wurden sie jedoch erst den 2010er Jahren ein in der Praxis ernst zu nehmendes Mittel der Bildklassifikation. Über die Jahre gab es mehrere Fortschritte, wie die Einführung von ReLU, verschiedene Pooling-Methoden oder die Batch Normalization gegeben, welche CNNs immer mächtiger machten. Außerdem gab es Fortschritte bei Optimizern wie Adam. [4]

2.3 Facial Emotion Recognition

FER (Facial Emotion Recognition), auch FEA (Facial Expression Analysis [26]) genannt, ist ein Bereich des Affective Computings, bei dem Emotionen im menschlichen Gesicht erkannt werden sollen. Dies kann in der Praxis viele Anwendungen haben, wie z.B.: Emotionssensible Roboter, Müdigkeitsüberwachung bei Autofahrern oder interaktives Game Design. [26 Seite 1f] Wie beim Affective Computing, ist es das Ziel von FER, die Schnittstelle zwischen Menschen und Maschine einfacher zu gestalten und zu ermöglichen, dass Computer Menschen besser verstehen und deren Befehle besser interpretieren können.

Bei FER-Systemen unterscheidet man zwischen static und dynamic sequence FER. Bei statischen FER-Methoden wird versucht, eine Emotion anhand eines einzelnen Bildes zu erkennen. Dynamische Verfahren zielen stattdessen darauf ab, Emotionen aus den zeitlichen Zusammenhängen zwischen mehreren Bildern zu erkennen. Dabei werden Videos unterschiedlicher Länge analysiert und oft werden LSTM-Modelle eingesetzt. Des Weiteren gibt es Ansätze andere Informationen, z.B. Audio, für die Emotionserkennung zu benutzen. [3] Die vorliegende Arbeit bezieht sich hier ausschließlich auf statische FER.

Bei FER treten mehrere typische Probleme auf, die dafür sorgen, dass FER im Anwendungsfall noch immer vor schwierigen Herausforderungen steht. So stehen für FER wenig gute Trainingsdatensätze zur Verfügung, die Modelle für Situationen im echten Leben ausreichend trainieren könnten. [3] In Gesichtern von Menschen sind viele Informationen enthalten, wie beispielsweise die Hautfarbe, die Dichte der Augenbrauen, das Alter oder die Farbe der Augen. Viele dieser Informationen sind aber für den Sinn von FER völlig unbedeutend und müssen daher vom Modell unbeachtet bleiben. Stellt man z.B. einen Datensatz zusammen, in dem mehr Frauen als Männer als Beispiel für eine Emotion benutzt wurden, könnte, könnten Bildern von Frauen fälschlicherweise öfter diese Emotion zugeordnet wird als Männern.

Datensätze zu erstellen, die frei von Bias sind, und dennoch ausreichend viele Bilder enthalten, ist daher schwierig und sehr aufwändig. [2] Ohne solche Datensätze wiederum ist das Trainieren von Modellen für FER ohne starkes Overfitting jedoch sehr schwierig.

Emotionen können außerdem unterschiedlich starke Variationen aufweisen, die berücksichtigt werden müssen, wodurch die benötigte Menge an Daten stark erhöht wird.

Auch wenn FER in den letzten Jahren große Fortschritte gemacht hat, stellen sich Anwendungsfälle im echten Leben oft als schwierig heraus.

Abbildung 7: Vergleich von Labor- zu Real World Bedinungen [3]

In Abbildung 7 ist der Unterschied zwischen laborerzeugten Bildern und denen aus dem echten Leben dargestellt. Zu erkennen ist, dass sie sich in ihrer Intensivität stark unterscheiden können.

Des Weiteren werden manche Emotionen in anderen Kulturen unterschiedliche dargestellt, weshalb oft nur Emotionen klassifiziert werden können, bei denen dies nicht der Fall ist. Im Jahre 1971 stellten Ekman und Friesen sechs primäre Emotionen vor [33], welche alle distinktive Inhalte aufweisen. Typischen Emotionen, die frei von kulturellen Einflüssen oder Ähnlichem sind, umfassen: Angst, Wut, Glücklichkeit, Trauer, Überraschung und Ekel. [1] Zusammen mit einer neutralen Klasse wurden diese Emotionen in Abbildung 7 dargestellt.

Eine weitere Schwierigkeit beim Erstellen von Trainingsdatensätzen ist, dass auch für Menschen das Erkennen von Emotionen oft schwierig ist. So liegt der später in der vorliegenden Arbeit verwendeten FER2013 Datensatz nur bei einer menschlichen Klassifizierungsgenauigkeit von 65 -68%. [2, 4]

3. Erstellen eines KNNs zur Klassifizierung des FER2013-Datensatzes

In diesem Kapitel wird das Basismodell für Kapitel 4 erstellt. Zu diesem Zweck wurden unterschiedliche Learning Rate Scheduler und CNNs getestet. Das Modell, welches am Ende dieser Versuchsreihe die beste Test-Accuracy auf dem FER2013 Datensatz hatte, wurde als Basismodell für die darauffolgenden Versuche festgehalten.

3.1 Problemstellung

Da das Ziel der vorliegenden Arbeit eine Robustheitsprüfung eines KNNs unter Verwendung manipulierter Daten ist, musste zunächst ein KNN erstellt werden welches den FER2013 Datensatz (siehe Abschnitt 3.5.1) gut klassifiziert. Hierzu wurden verschiedene Learning Rate Scheduler und CNNs verglichen.

3.2 Ablauf

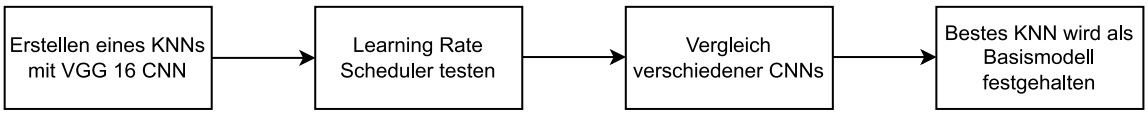

Abbildung 8: Ablauf der Schritte in Kapitel 3

Zunächst wurde ein KNN erstellt, welches als CNN das VGG16-CNN verwendete. Anhand dieses Netzes sollten dann verschiedene Learning Rate Scheduler getestet werden. Der am besten abschneidende Learning Rate Scheduler wurde bestimmt und es folgte der Vergleich verschiedener CNNs. Hierzu wurde das CNN (VGG 16) im KNN des ersten Netzes durch ResNet50 im zweiten und ResNet152 im dritten Test ersetzt. Es wurde ein Vergleich zwischen den so entstehenden drei Netzen angestellt und das Netz, welches den FER2013-Datensatz mit der höchsten Accuracy klassifizierte, wurde für das folgende Kapitel als Basismodell festgehalten. Alle in diesem Versuch verwendeten CNNs wurden auf dem ImageNet-Datensatz (siehe Abschnitt 3.5.2) vortrainiert.

3.3 Versuchsaufbau des Trainings

Im Folgenden werden die Versuche zum Trainieren eigener Modelle mit vortrainierten CNNs beschrieben. Alle Netze wurden auf dem FER2013 Datensatz trainiert und alle Versuche hatten Gemeinsamkeiten. Das Ziel war, so viel wie möglich bei jedem Training unverändert zu lassen um die Performance der verschiedenen CNNs zu isolieren.

Spätere Experimente trainierten im Haupttrainingsdurchlauf nur 200 Epochen, während einige frühere Versuche 300 Epochen trainierten. Diese Änderung wurde vorgenommen, da nach 150 Epochen keine oder nur eine unwesentliche Änderung am Ergebnis festzustellen war. Da ein Trainingsdurchlauf mehrere Stunden in Anspruch nehmen kann, wurde die Trainingsdauer verkürzt, um mehr Experimente durchführen zu können.

Es wurde ein Netz erstellt, welches aus einem vortrainierten CNN und einem anschließendem Klassifizierungsnetzwerk aus Dense-Layern und Dropout-Layern bestand. Der CNN-Teil des Netzes wurde daraufhin eingefroren und das Netzwerk für 300

Epochen (200 Epochen bei späteren Versuchen) auf dem FER2013 Datensatz trainiert. Anschließend wurde das Einfrieren des CNNs aufgehoben und das gesamte Netz für weitere 50 Epochen nochmals auf demselben Datensatz trainiert. Diesen zweiten Trainingsdurchlauf, der mit kleinerer Learning Rate durchgeführt wird, nennt man „Fine Tuning“. Fine Tuning wird angewandt, um zu Beginn des Trainings die vortrainierten Werte des CNNs nicht zu überschreiben. Es werden zunächst nur die hinteren Layer des Netzes trainiert. Wenn dieses Training abgeschlossen ist, lohnt es sich, die Gewichte des CNNs für den aktuellen Datensatz anzupassen, da so die Leistung des Modells nochmal gesteigert werden kann. Am Ende jedes Trainings wurden die Weights der Epoche gespeichert, welche die beste Validation-Accuracy aufwiesen.

Alle Versuche wurden mit dem Adam Optimizer und Sparse Catigorical Crossentropy als Loss Function durchgeführt. Bei der Wahl der Loss Funktion wurde sich an [30, Seite 113] orientiert, da zwischen mehr als zwei Klassen unterschieden werden sollte und als Ausgabe eine ganze Zahl gewünscht war. Beim Optimizer wurde sich an [25, Seite 319] orientiert. Auch hier verwenden die Autoren Adam für Deep Learning FER.

3.4 Aufbau des Netzes

Der Aufbau des Netzes ist bei allen Versuchen derselbe und unterscheidet sich nur beim verwendeten CNN. Bei einigen Experimenten wurde ein etwas anderer Übergang vom CNN zum Klassifizierungsnetzwerk eingesetzt. Diese Änderung hatte allerdings keine Auswirkung auf die Performance, da dasselbe Netz (mit Ausnahme des verwendeten CNNs) entstand.

Das Netz begann mit einem Input Layer mit der Form (224, 224, 3), obwohl es sich bei dem FER2013 Datensatz um 48x48 Pixel Graustufenbilder handelte. Dies wurde nur dafür gemacht, um einen einfacheren Übergang vom Input in das CNN zu ermöglichen, da diese oft RGB-Bilder erwarten. Zu diesem Zweck wurden vor dem Training alle Bilder im Datensatz auf 224x224 RGB-Bilder hochskaliert. Die Bilder wurden in einem Keras Dataset gespeichert, um die Memoryanforderungen beim Training zu reduzieren. So ein Dataset wird verwendet, damit nicht alle Bilder auf einmal in den RAM geladen werden müssen. Stattdessen wird im Training immer nur die aktuell benötigten Daten in den Speicher geladen, wodurch wesentlich größere Datensätze verwendet werden können. [27] Verwendet wurde hierzu die Keras Funktion tensorflow.keras.preprocessing.image_dataset_from_directory. [32]

Als nächstes folgte das jeweilige CNN. Über ein GlobalAveragePooling2D-Layer wurde das CNN mit dem darauf folgenden Dense-Netzwerk verbunden. Dieses bestand aus drei Dense-Layern: je eines mit 128, 64 und 7 Neuronen. Das letzte Layer war das Output-Layer mit 7 Neuronen, da zwischen 7 Emotionen unterschieden werden sollte. Das letzte Layer verwendet die softmax Aktivierungsfunktion während die beiden vorherigen ReLU einsetzen. Zwischen diesen Layern liegen noch zwei Dropout-Layer mit einer Droputrate von 0.2. In der folgenden Abbildung 9 wird das gesamte Netzwerk gezeigt. Ausgetauscht wurde bei jedem Versuch nur das CNN, der Rest bliebt unverändert.

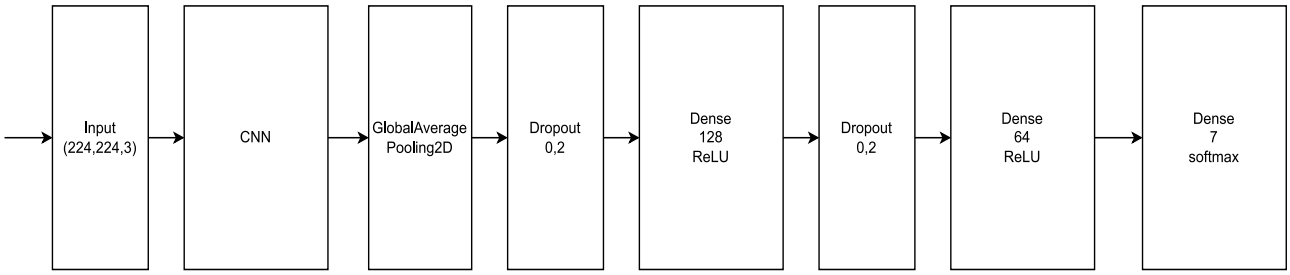

Abbildung 9: Darstellung der verwendeten Netzwerkarchitektur

Der oben beschriebene Unterschied beim Zusammensetzen einiger Modelle bezieht sich darauf, dass einige CNNs von sich aus ein GlobalAveragePooling2D-Layer beinhalten. In diesen Fällen musste es nicht extra hinzugefügt werden. Der Aufbau unterschied sich letztendlich nicht.

3.5 Datensätze

Es folgt eine kurze Vorstellung der vorliegenden Arbeit verwendeter Datensätze.

3.5.1 FER2013-Datensatz

Bei der „International Conference on Machine Learning" (ICML) 2013 wurde der FER2013-Datensatz vorgestellt, welcher bis heute einer der Vergleichsmaßstäbe für die Leistung von FER-Modellen ist. Die menschliche Genauigkeit bei der Emotionserkennung in Bildern dieses Datensatzes liegt bei ungefähr 65.5%. [4] Der Datensatz besteht aus 35887 48x48 Pixel Graustufenbildern. Auf diesen Bildern werden 7 Emotionen dargestellt: Wut, Ekel, Angst, Glücklichkeit, Trauer, Überraschung und Neutral. Der Trainings-, Validierungsanteil des Datensatzes liegt bei 28709 Bildern und der Testanteil bei 7178. In [4] stellten die Autoren ihren Ansatz zur Klassifikation von Fer2013 vor und erreichten in ihrer Arbeit eine state of the art Performance von 73.28% Testgenauigkeit. Sie verwenden hierfür eine VGGNet Architektur.

3.5.2 ImageNet-Datensatz

ImageNet ist ein Bilddatensatz, der sich anhand der WordNet Hierarchie orientiert. WordNet beschreibt über 100.000 Phrasen die mach auch „synset" nennt. Von diesen Phrasen sind über 80.000 Nomen. In ImageNet wird versucht jedem dieser Phrasen im Durchschnitt 1000 Bilder mit Label zuzuordnen. Alle Bilder sind von Menschen annotiert. [17] Dadurch entsteht ein gewaltiger Bilddatensatz, auf dem die in der vorliegenden Arbeit verwendeten CNNs vortrainiert wurden.

3.6 Tools und Bibliotheken

Als Hauptbibliothek wurde Tensorflow verwendet. „Tensorflow ist eine End-to-End-Open-Source-Plattform für maschinelles Lernen" [31], die aufgrund ihrer umfangreichen und detaillierten Dokumentation ausgewählt wurde. Eng mit Tensorflow verbunden ist Keras, eine Deep Learning API für Python. Sie bietet unter anderem die Möglichkeit Neuronale Netze zu erstellen oder Daten in den Speicher zu laden (siehe Teil 3.4). Aus diesem Grund wurde als Programmiersprache Python verwendet. Für die Erstellung der Confusion Matrizen wurde scikit-learn verwendet. Sowohl numpy als auch OpenCV wurden für die Umstrukturierung von Daten verwendet. Als letztes bietet CUDA die Möglichkeit, eine NVIDIA-Grafikkarte als Recheneinheit für das

Training zu verwenden, wodurch die Trainingszeit stark verkürzt wurde (ohne CUDA dauerte ein Training mehrere Tage, mit CUDA nur um die 10 Stunden).

In der Tabelle1 finden sich die verwendeten Versionen dieser Bibliotheken und Tools.

Tabelle 1: Verwendete Bibliotheken und Tools.

Tool	Version
Python	3.9.12
numpy	1.22.3
Tensorflow	2.6.0
Keras	2.6.0
scikit-learn	1.1.2
CUDA	11.7
OpenCV	4.5.5

3.7 Vergleich verschiedener Learning Rate / Scheduler

In diesem Teil wurden drei verschiedene Arten von Learning Rate Schedulern getestet: Linearer Decay, Cosine Decay with warm Restart und Reduce Learning Rate on Plateau (RLRP). Alle Tests zu Learning Rates wurden Anhand vom VGG16 CNN durchgeführt.

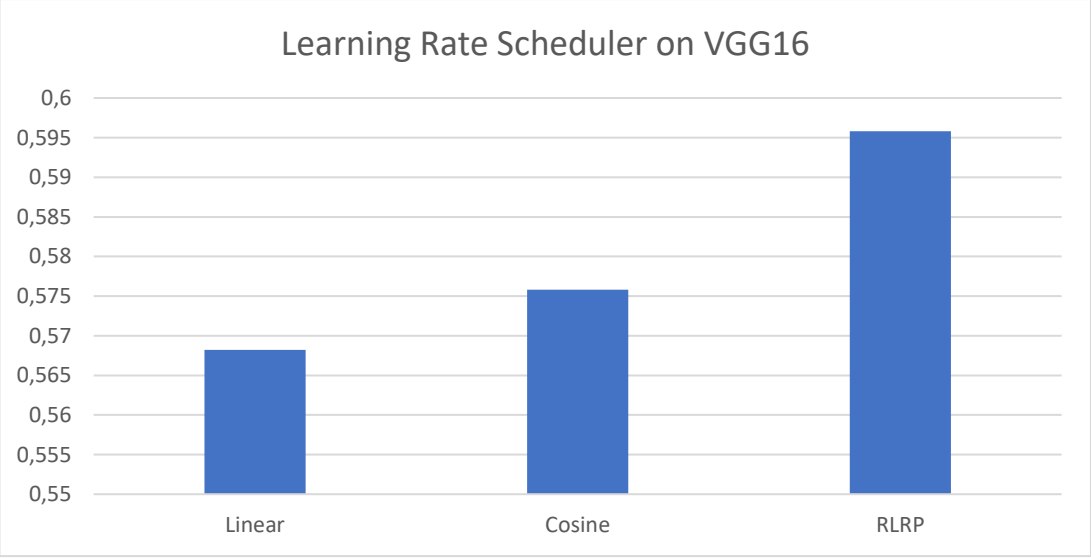

Abbildung 10: Vergleich verschiedener LR Scheduler

Wie in Abbildung 10 zu erkennen ist, hatte RLRP mit 59,58% Testgenauigkeit das beste Ergebnis, gefolgt von Coine Decay mit 57,58% während Linear Decay mit 56,82% am schlechtesten abschnitt. Bei allen Versuchen begann die Learning Rate mit 0.01 beim ersten Trainingsdurchlauf, und die Scheduler verwendeten eine decay Rate (die Rate, mit der die Learning Rate abnimmt) von 0,9.

Anhand dieser Ergebnisse wurde bei allen späteren Experimenten RLRP verwendet. Als Wert zur Messung der Performance (womit RLRP entscheidet, ob die Learning Rate reduziert wird), wurde der Validation Loss verwendet. Es wurde ein Patience

Wert von 5 eingestellt, es konnte also maximal alle 5 Epochen die Learning Rate reduziert werden. Außerdem wurde eine minimale Learning Rate von 1e-5 eingestellt. Unter diesen Wert konnte die Learning Rate also nicht reduziert werden.

Beim Training mit RLRP fiel auf, dass teilweise die Learning Rate trotz Verbesserung vom Validation Loss reduziert wurde. Das lag am min delta Wert. Dieser gibt an um wieviel der gemessene Wert besser werden muss damit die Learning Rate nicht reduziert wird. Da die Ergebnisse allerdings auch ohne Einstellung dieses Wertes für die Fragestellung der vorliegenden Arbeit zufriedenstellend waren, wurden hier keine weiteren Experimente vorgenommen. Eventuell ließe sich die Performance der Modelle jedoch mit einer geschickten Einstellung des min delta Wertes noch verbessern.

Beim Fine Tuning wurde bei allen Experimenten der Exponential Decay Scheduler mit einer initialen Learning Rate von 1e-5, einer Decay Rate von 0.75 und einer Schrittgröße von 10000 verwendet. Beim Fine Tuning wurden keine Vergleichsexperimente von unterschiedlichen Schedulern durchgeführt.

3.8 Training mit verschiedenen CNNs

In diesem Teil wurden drei unterschiedliche CNNs in das KNN aus Teil 3.4 eingefügt und die daraus entstehenden Netze auf dem FER2013-Datensatz trainiert. Da RLRP (in Teil 3.7) am besten abgeschnitten hatte, wurde dieser Scheduler für das Training (nicht aber für das Fine Tuning) verwendet. Bei jedem Netz wurden die Trainingsverläufe beim normalen Training und Fine Tuning analysiert. Anschließend folgte eine Gegenüberstellung der Ergebnisse und das am besten abschneidende (höchste Test-Accuracy) Modell wurde als Basismodell für das folgende Kapitel festgehalten.

3.8.1 VGG16

Das VGG16 Netz belegte Platz 3 mit einer Test Accuracy von 57,58%. Nach dem ersten Trainingsdurchlauf (vor dem Fine Tuning) lag diese bei 47,44%. In Abbildung 11 und 12 ist der Verlauf vom Trainings Loss und Validation Loss dargestellt.

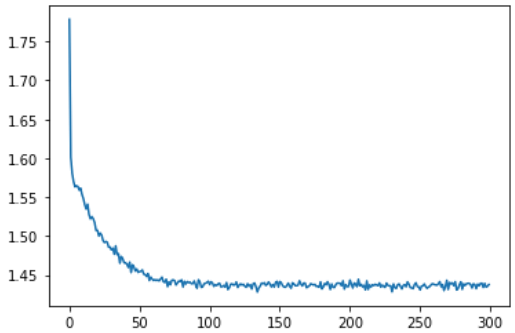

Abbildung 11: Training Loss VGG16

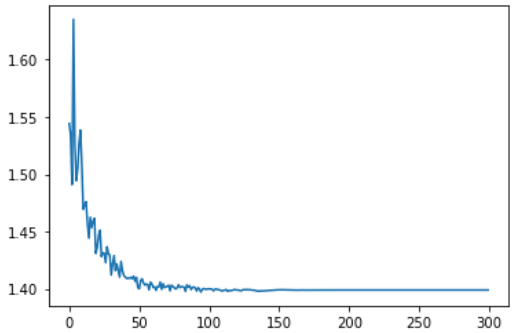

Abbildung 122: Validation Loss VGG16

Die Kurven verlaufen sehr gleichmäßig. Das bedeutet, dass das Modell weder undernoch overfittet. Die Trainings- und Validation-Loss des Fine Tunings sind in Abbildung 13 und 14 dargestellt.

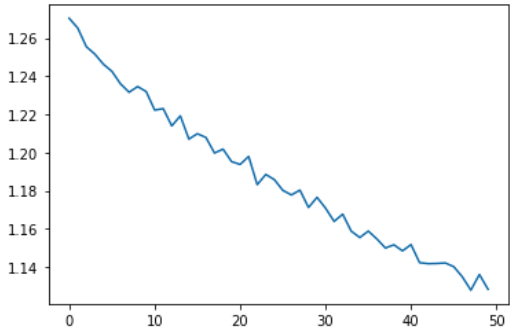

Abbildung 13: Trainings Loss VGG16 Fine Tuning

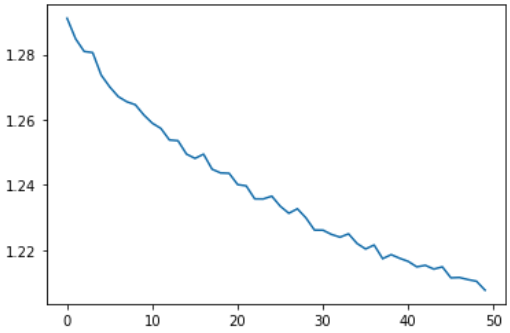

Abbildung 14: Validation Loss VGG16 Fine Tuning

Auch bei diesen Kurven ist weder Under- noch Overfitting zu sehen. Allerdings fällt auf, dass die Kurven bis zum Ende fallen. Auch zu sehen an der Validation Accuracy (Abbildung 15), welche bis zum Ende des Fine Tunings steigt:

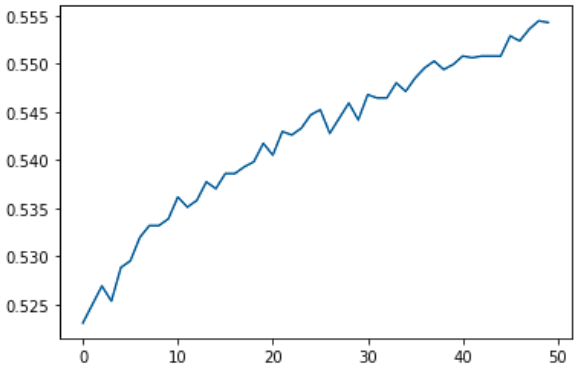

Abbildung 14: Validation Accuracy VGG16 Fine Tuning

Es ist also wahrscheinlich, dass das Modell mit einer höheren Epochenzahl im Fine Tuning besser abgeschnitten hätte.

3.8.2 ResNet50

Den zweiten Platz belegt ResNet50 mit einer Test Accuracy von 65,03%. Hier die Verläufe des ersten Trainings (siehe Abbildung 16 und 17):

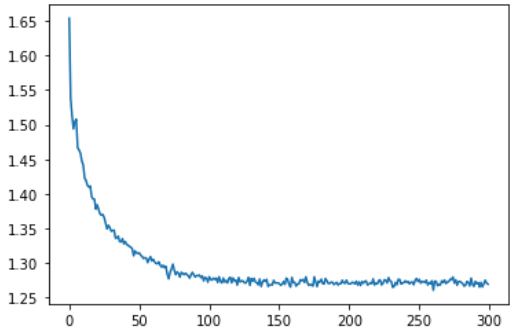

Abbildung 15: Training Loss ResNet50

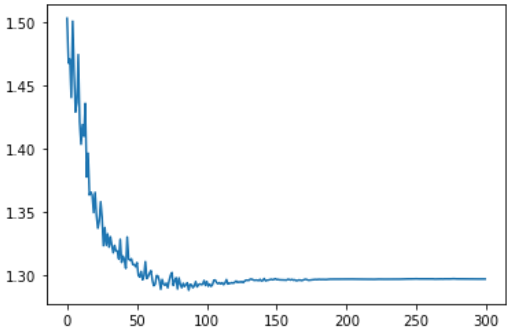

Abbildung 16: Validation Loss ResNet50

Zu erkennen ist beim Validation Loss ungefähr ab Epoche 100 einen leichten Anstieg, was auf leichtes Overfitting hindeutet. Zudem sind ab diesem Punkt kaum noch Verbesserungen im Trainingsloss zu erkennen. Auch bei der Validation Accuracy ist dies zu sehen (siehe Abbildung 18):

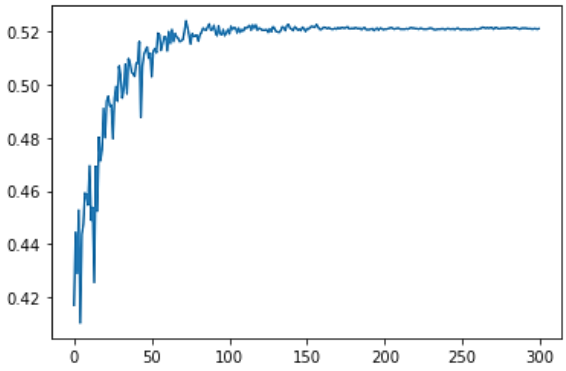

Abbildung 17: Validation Accuracy ResNet50

Aus diesem Grund wurde die Trainingszeit im Versuch mit ResNet152 auf 200 Epochen gekürzt. Vor dem Fine Tuning hatte das Modell eine Test Accuracy von 51,73%. Beim Fine Tuning ergab sich nun zum ersten Mal sehr starkes Overfitting.

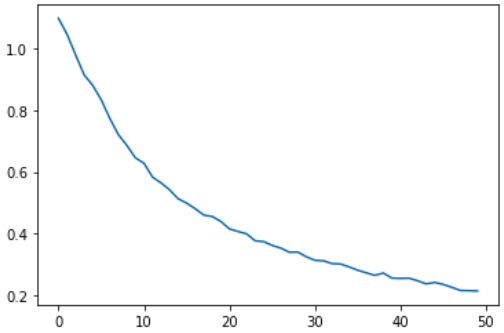

Abbildung 18: Training Loss ResNet50 Fine Tuning

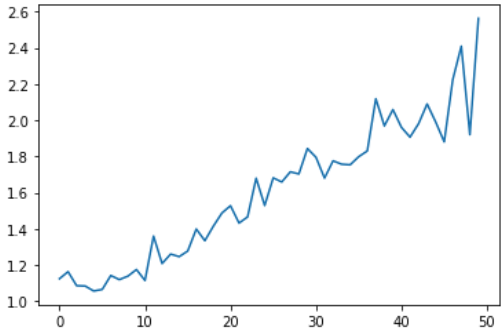

Abbildung 19: Validation Loss ResNet50 Fine Tuning

Beim Validation Loss (Abbildung 20) ist ab Epoche 10 ein starker Anstieg zu sehen, während der Trainings Loss (Abbildung 19) weiter abnimmt. Das deutet auf starkes Overfitting hin. Bei der Validation Accuracy (Abbildung 21) sind starke Sprünge erkennbar. Da am Ende des Trainings jedoch die Weights der Epoche mit der besten Validation Accuracy gewählt wurde, wurde das Endergebnis nicht gestört.

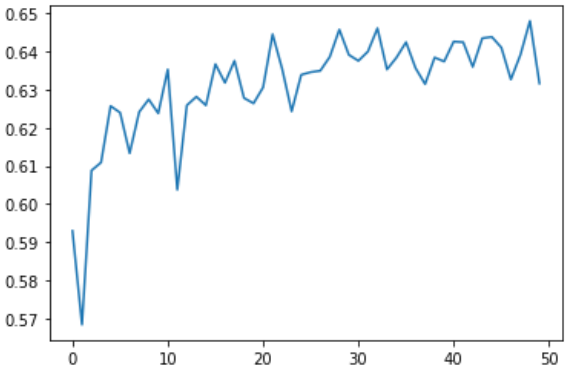

Abbildung 20: Validation Accuracy ResNet 50 Fine Tuning

Dennoch schneidet das Modell mit 65,03% Test Accuracy besser ab als VGG16.

3.8.3 ResNet152

Da das Training vom ResNet152 Netz sehr ähnlich verlief wie das von ResNet50, wird hier nur die Test Accuracy von 65,65% erwähnt. Auch hier ist im Fine Tuning dasselbe Overfitting wie bei ResNet50 zu sehen. Dennoch belegte das Modell Platz 1 und wird daher im folgenden Kapitel als Basismodell verwendet. Zum Vergleich: In [4] stellten die Autoren ihre Lösung für den FER2013 Datensatz vor und erreichten mit ihrem Netz eine Testgenauigkeit von 73,28%, was zu dem Zeitpunkt die state-of-the-art Performance darstellte.

Es folgt eine Confusion Matrix vom Test des Basismodells (ResNet 152 Modell) auf den Testdaten vom FER2013 Datensatz.

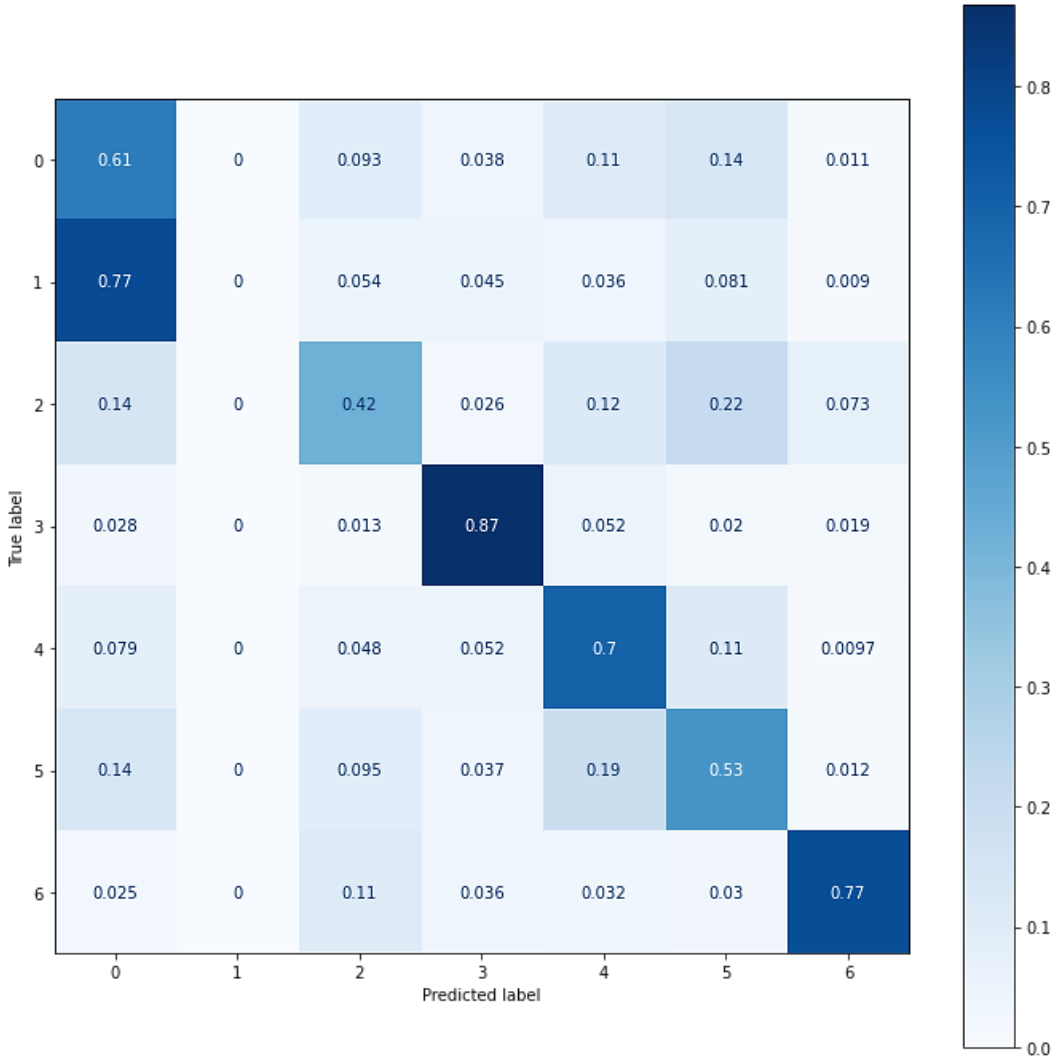

Abbildung 21: Confusion Matrix für Basismodell FER2013-Test

In Abbildung 22 ist dargestellt auf welche Weise „sich das Modell irrt“. Man sieht in dieser Grafik was das Modell für ein gegebenes Label des Testdatensatzes vorhergesagt. In einer „perfekten Welt“ würden in der Hauptdiagonalen nur Einsen stehen (die Hauptdiagonale wäre dunkelblau, alle anderen Werte weiß).

Die Label stehen für die zu klassifizierenden Emotionen:

0= Wut, 1= Ekel, 2= Angst, 3= Glücklichkeit, 4= Neutral, 5= Trauer, 6= Überraschung

In Abbildung 22 sieht man, dass Klasse 3 (Happy) mit 87%, am besten erkannt wird und Klasse 1 (Disgust) am schlechtesten. Man sieht außerdem, dass Klasse 1 nicht nur selten richtig erkannt wird, sondern im Allgemeinen nie als Vorhersage vom Modell „gewählt" wird. Man erkennt, dass mit Ausnahme von Klasse 1 bei allen anderen Klassen am häufigsten die richtige Klasse vorhergesagt wird. Das Modell liegt also meistens richtig, solange es nicht versucht Ekel vorherzusagen.

3.9 Vergleich der Ergebnisse

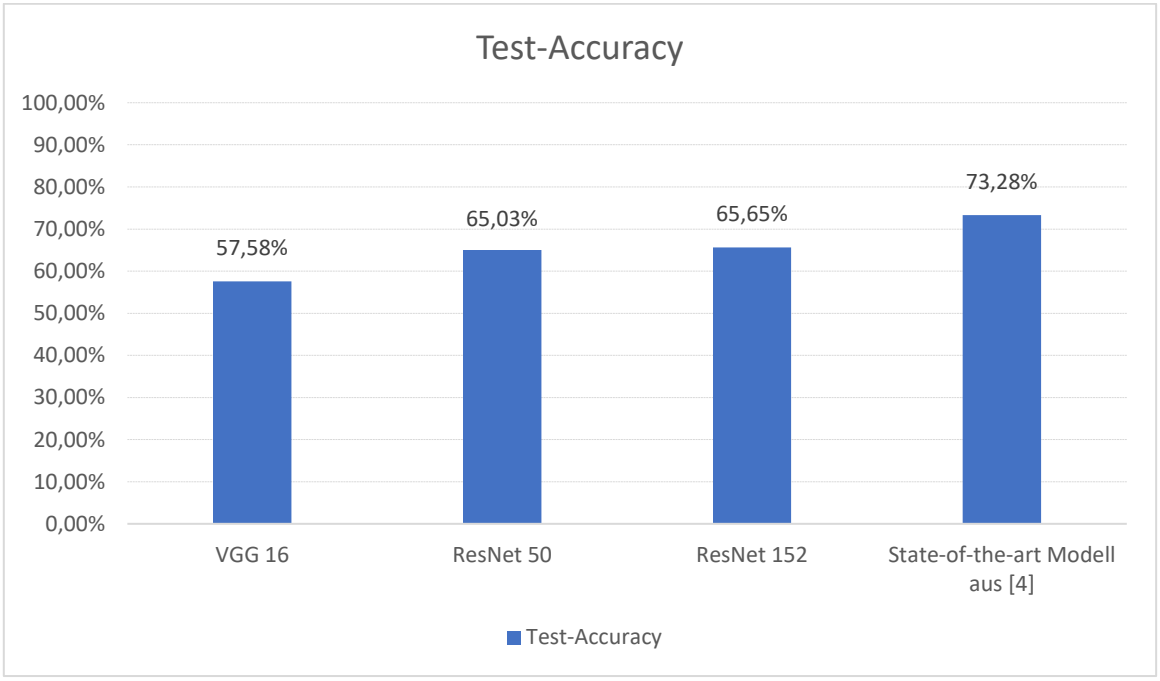

Abbildung 22: Vergleich zwischen CNNs

In Abbildung 23 sieht man den Vergleich der Test-Accuracy, der getesteten CNNs und der state-of-the-art-AI aus [4] als Vergleichswert. Am besten schnitt (state-of-the-art Modell ausgenommen) das ResNet152 mit einer Test-Accuracy von 65.65% ab. Das KNN welches ResNet 152 verwendet wurde daher in Kapitel 4 als Basismodell verwendet.

4. Manipulation der Daten und Robustheitsprüfung des Netzes

In diesem Kapitel soll die Störung der vorher trainierten Modelle überprüfen werden, und ob beziehungsweise wie stark ihre Performance unter der Manipulation der Trainings- und Testdaten leidet. Zu diesem Zweck wurde das bisher am besten abschneidende Model ausgewählt. Es wurde überprüft, wie sich die Test Accuracy verändert, wenn das Modell die Klassen für eine manipulierte Version des FER2013 Datensatzes vorhersagen soll.

Außerdem wurde das Netz auf diesen manipulierten Versionen von FER2013 neu trainiert, um zu überprüfen, wie dasselbe Netz abschneidet, wenn es schon beim Training solche Daten erhält.

Es wurden vier verschiedene Versionen des Datensatzes verwendet um die Stabilität des Netzes zu testen.

Um die Performance zu messen, wurde zum einen die Test Accuracy des Modells betrachtet und zum anderen Confusion-Matrizen erstellt, die einen genaueren Einblick auf die Performance bei der Klassifikation, in Bezug auf einzelne Klassen, geben sollen. Eine mögliche Fragestellung wäre beispielsweise, wie gut das Modell darin ist, Angst in einem Gesicht zu erkennen, wenn beim Training die Augen der Gesichter verdeckt waren.

In Kapitel 3 schnitt das Netz am besten ab, welches ResNet152 als CNN einsetzte. Die Test Accuracy dieses Modells lag bei 65,65% und gilt ab jetzt als Basiswert. Wird im Folgenden vom Basismodell gesprochen, ist damit das in Abschnitt 3.8.3 beschriebene Modell zu ResNet152 gemeint, welches auf der normalen Version von FER2013 trainiert wurde.

4.1 Ablauf

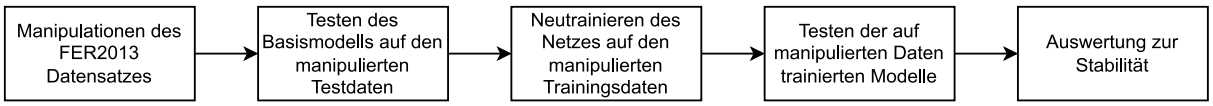

Abbildung 23: Ablauf von Kapitel 4

Zunächst wurde, der FER2013 Datensatz mit unterschiedlichen Methoden manipuliert (Teil 4.2). Es folgte ein Test des Basismodells auf den so entstandenen Testdaten (Teil 4.3). Daraufhin erfolgte ein erneutes Training des Netzes auf den manipulierten Trainingsdaten, welche beim Basismodell verwendet wurden (Teil 4.4). Die so entstandenen Modelle wurden auf allen vier Versionen (dem normalen FER2013 Datensatz und drei manipulierten Versionen) der Testdaten getestet (Teil 4.5). Im Abschluss erfolgte eine Auswertung zur Stabilität des Netzes (Teil 4.6).

4.2 Manipulation von FER2013

Das Ziel der Manipulationen war zu testen, auf welche Veränderung das Basismodell auf welche Weise reagiert und wie Modelle abschneiden, die auf diesen manipulierten Daten überprüft werden sollten.

Zu diesem Zweck wurden zwei verschiedene Methoden der Manipulation angewandt. Zum einen die Verdeckung (Occlusion) bestimmter Bereiche der Daten und zum anderen das Hinzufügen von Rauschen (Noise) im Bild.

Bei der Verdeckung war das Ziel sowohl Augen als auch den Mund in den Bildern in jeweils einer Version des Datensatzes zu überdecken. Zu diesem Zweck wurden mehrere Versuche durchgeführt. Zum Beispiel wurde versucht, mit einem Haarcascade Classifier die Augen oder den Mund in den Bildern zu erkennen und den entsprechenden Bereich in den Bildern schwarz zu färben. Haarcascade Classifier umfassen, Feature und Object Recognition Algorithmen die auf der Erkennung von sogenannten Haar-like Features basieren und wurden in [29] zuerst vorgestellt. Sie sind eine sehr recheneffiziente Variante, bestimmte Dinge, wie z.B. Augen in Bildern zu erkennen.

Wie in Abbildung 25 zu erkennen ist, wurden hier oft falsch positive Ergebnisse geliefert, und machten damit diesen Ansatz für Experimente unbrauchbar.

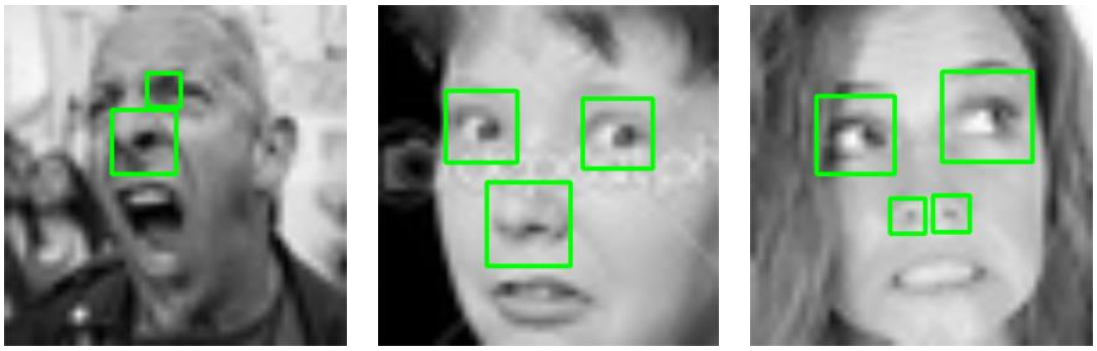

Abbildung 24: Haarcascade Beispiele für Augenerkennung

Ein Vorteil des FER2013 Datensatzes ist jedoch, dass Augen und Mund bei den meisten Bildern auf einer ähnlichen Höhe sind. Das zufriedenstellenste Ergebnis wurde damit erreicht, einen schwarzen Balken über eine manuell ausgewählte Region im Bild zu legen.

In Abbildung 26 wurden per Zufallsgenerator 6 Bilder des Testdatensatzes ausgewählt. Es ist zusehen, dass bei den meisten Bildern die Augen gut verdeckt sind. Nur beim 5ten Bild der Reihe (von links beginnend) ist der Balken ein wenig zu hoch um die Augen zu verdecken. Das Ergebnis dieses Ansatzes ist jedoch trotzdem genauer als alle anderen Versuche eine Verdeckung der Augen anzuwenden.

Abbildung 25: Beispiele für die Augenverdeckung

Bei der Verdeckung des Mundes wurde auf die gleiche Weise verfahren. Hier einige Bespielbilder (Abbildung 27):

Abbildung 26: Beispiele für die Mundverdeckung

Als letzte Version wurde dem Datensatz ein Rauschen vom Typ Salt and Pepper Noise hinzugefügt.

Abbildung 27: Beispiele für SnP Noise

Bei dieser Art der Manipulation werden zufällig eine bestimmte Menge an Pixeln schwarz oder weiß eingefärbt (in Abbildung 28 farbig zur besseren Sichtbarkeit). In diesen Versuchen lag die Menge geänderter Pixel bei 5%.

4.3 Testen des Basismodells auf manipulierten Testdaten

Als erster Test zur Stabilität wurde das Basismodell auf den manipulierten Versionen getestet. Zum Vergleich: das Basismodell hatte beim Test auf FER2013 eine Test-Accuracy von 65,65%. In der folgenden Confusion Matrix wurde der Test vom Basismodell auf dem manipulierten Datensatz mit verdeckten Augen analysiert, der mit einer Testgenauigkeit von 42,37% durchlief, (Abbildung 29).

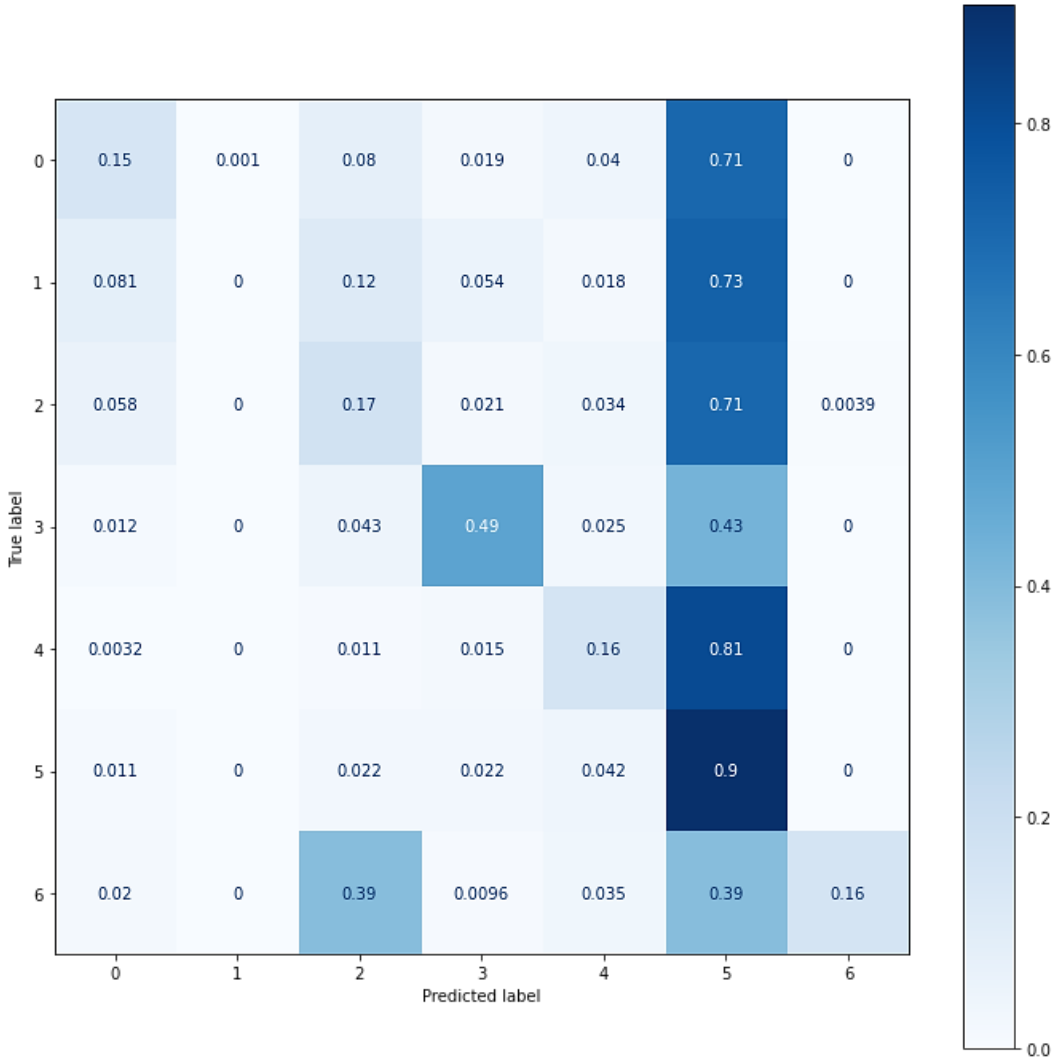

Abbildung 28: Confusion Matrix Basismodell Eye-Occlusion-Test

Auch in diesem Test wird Klasse 3 (Glücklichkeit) mit 49% oft richtig klassifiziert. Deutlich besser schneidet jedoch Klasse 5 (Trauer) mit 90, ab. Dieser Wert ist mit Vorsicht zu betrachten, da Klasse 5 bei fast allen echten Klassen auffälligerweise am häufigsten als Ergebnis vorhergesagt wird. Wenn z.B. immer nur Klasse 5 vorhergesagt wird, ist natürlich das Ergebnis für die echte Klasse 5 immer richtig, allerdings wäre das Modell dann außerstande andere Klassen zu erkennen. Es fällt außerdem aus das Klasse 1 (Ekel) hier auch, wie in Abbildung 22, fast gar nicht als Ergebnis ausgegeben wird.

Als nächstes folgte der Test des Basismodells auf der FER2013 Version mit verdeckten Mündern. Die Test-Accuracy lag hier nur noch bei 28,5%. Das war zwar immer noch besser, als hätte das Modell nur „geraten“ (bei 7 Klassen würde bei reinem Raten eine Test-Accuracy von 1/7 ≈ 0,1429 oder 14,29% erwartet werden), allerdings war dieses schon ein starker Verlust von den ursprünglichen 65,65%. Eine mögliche Erklärung dafür, dass die Test-Accuracy stark unter der vom Test bei den verdeckten Augen liegt, wäre dass der Mund im Gesicht wichtiger für FER ist als die Augen. Dies stimmt auch mit den Ergebnissen aus [26, Seite 31] überein. Die entsprechende Confusion Matrix ist in Abbildung 30 dargestellt:

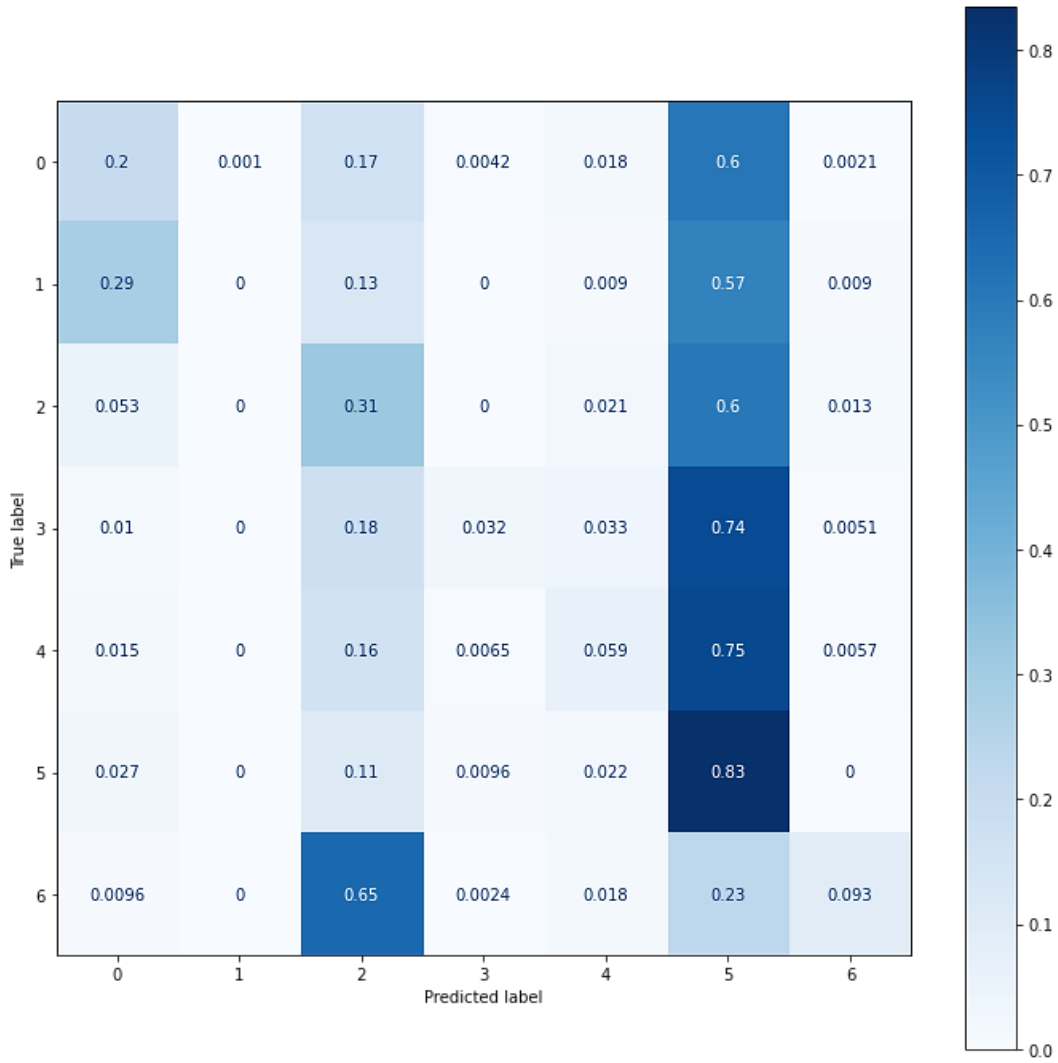

Abbildung 29: Confusion Matrix Basismodell Mouth-Occlusion-Test

Man erkennt ein ähnliches Klassifizierungsverhalten wie beim letzten Test. Nur wurde auch hier Klasse 3 (Glücklichkeit) mit 3,2% nicht mehr gut erkannt. Hier wäre ebenfalls eine mögliche Erklärung, dass der Mund für die Erkennung von Glücklichkeit nötig ist für das Modell. Auffällig ist, dass das Modell für Klasse 6 (Überraschung) mit 65% Wahrscheinlichkeit fälschlicherweise die Klasse 2 (Angst) vorhersagt.

Für den letzten Test des Basismodells wurde der Datensatz mit Salt and Pepper (SnP) Noise analysiert. Die Testgenauigkeit lag bei 44,42% und die Confusion Matrix sah wie folgt aus (Abbildung 31):

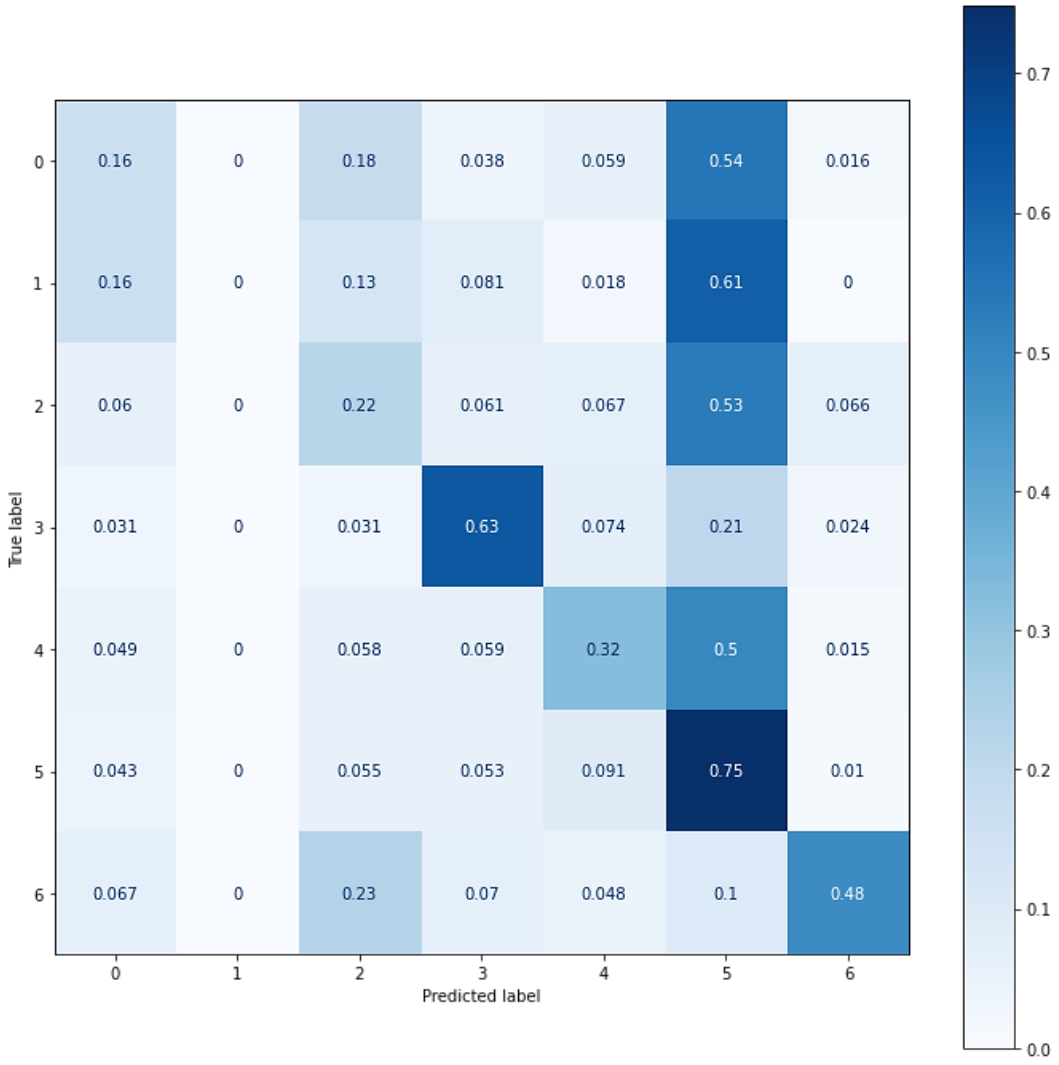

Abbildung 30: Confusion Matrix Basismodell SnP-Test

Es zeigte sich ein ähnliches Ergebnis zum Test mit verdeckten Augen. Der auffälligste Unterschied war, dass Klasse 6 (Überraschung) mit 48% besser klassifiziert wurde als bei den beiden letzten Tests. Anders beim ersten Test auf dem nicht manipulierten Datensatz (Abbildung 22), bei dem Klasse 6 mit 77% erkannt wurde. Eine mögliche Erklärung für das Versagen bei der Klassifikation von Überraschung bei den Tests mit Verdeckung wäre, dass für die Erkennung von dieser Emotion sowohl Augen als auch Mund entscheidend sind.

4.4 Trainieren mit manipulierten Daten

Als nächstes wurde das Netz auf den manipulierten Datensätzen neu trainiert. Zunächst wurde das Training auf dem „verdeckte Augen"-Datensatz betrachtet. Die Testgenauigkeit auf dem normalen FER2013 Datensatz betrug am Ende des Trainings 54,81% auf FER2013. Wie beim Training in 3.8.2 (Abbildungen 19 und 20), sieht man (Abbildungen 32 und 33) starkes Overfitting beim Fine Tuning.

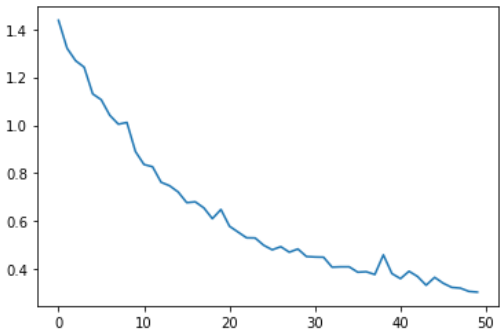

Abbildung 31: Training Loss Eye-Occlusion

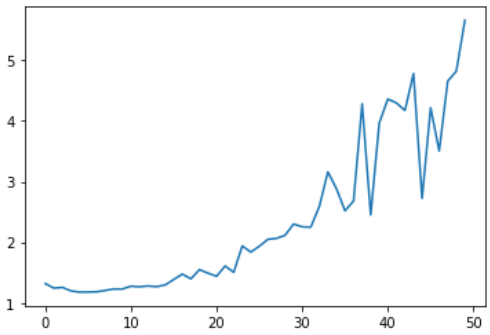

Abbildung 32: Validation Loss Eye-Occlusion

Als nächstes wurde das Netz auf dem „verdeckter Mund Datensatz" trainiert. Es ergab sich eine Testgenauigkeit von 50,5% auf FER2013. Das Overfitting des vorherigen Trainings ist auch hier zu erkennen. Das Training auf dem „SnP Datensatz" ergab eine Testgenauigkeit von 55,67% auf FER2013, und auch hier gab es Overfitting.

4.5 Testen von manipulierten Modellen

In diesem Teil wurden die auf manipulierten Datensätzen trainierten Modelle getestet. Es wurden sowohl die Test-Accuray als auch die beim Test entstehenden Confusion Matritzen ausgewertet.

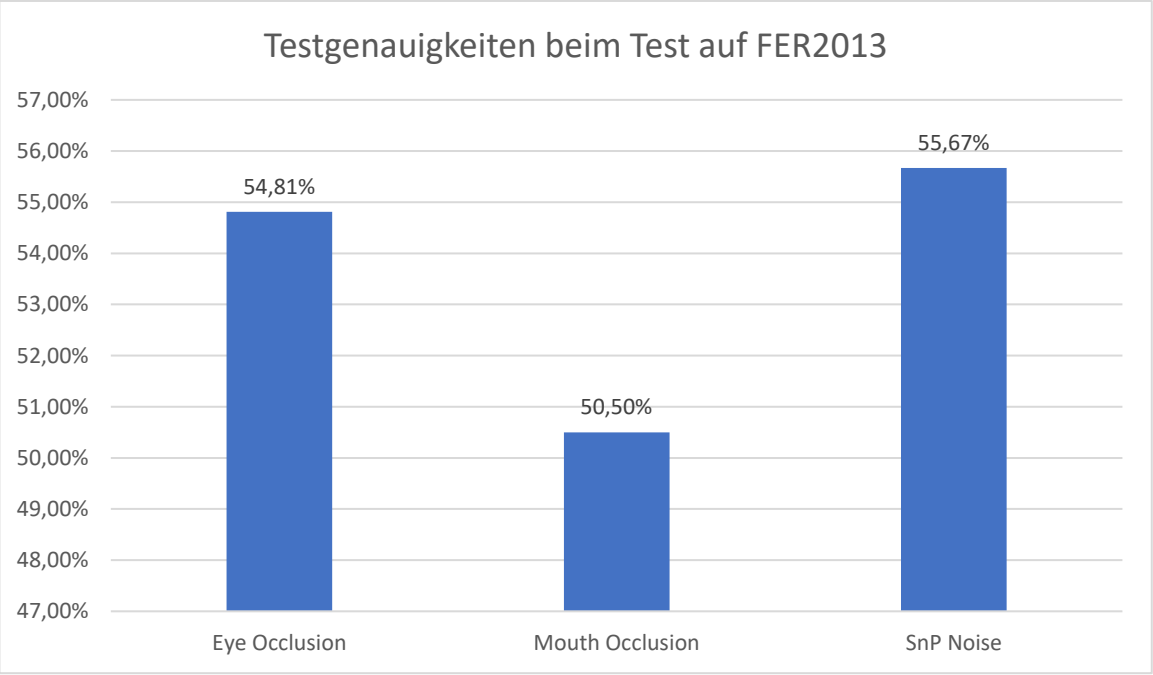

Abbildung 33: Vergleich von Testgenauigkeiten der „manipulierten Modelle" auf dem FER2013 Datensatz

Wie in Abbildung 34 zu sehen ist, wurde das Training von Salt and Pepper Noise am wenigsten gestört. Die größte Störung fand bei der Verdeckung des Mundes statt, und der Test mit den verdeckten Augen liegt zwischen diesen Werten. Eine mögliche Interpretation ist, dass das Netz aus dem Mund mehr lernt als aus der Augenregion, während Salt and Pepper das Netz im Allgemeinen stört.

4.5.1 Testen des Eye-Occlusion-Modells

Es wurde mit dem „verdeckte Augen Modell“ (das Netzwerk wurde auf dem Datensatz mit den verdeckten Augen trainiert) begonnen, welches eine Testgenauigkeit von 54,81% beim Testen auf FER2013 erzeugte. Es folgt die Confusion Matrix des Tests:

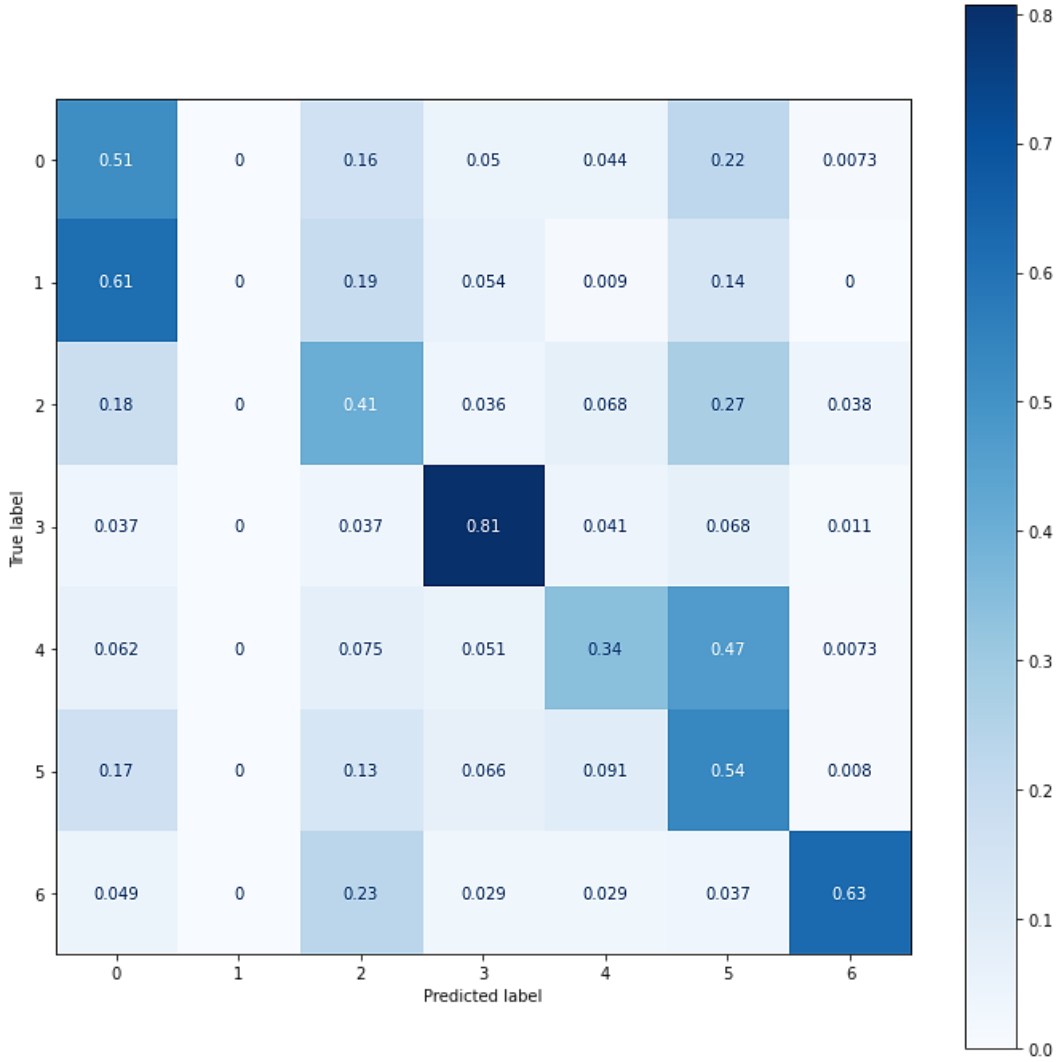

Abbildung 34: Confusion Matrix Eye-Occlusion-Modell FER2013-Test

An der dunklen Hauptdiagonalen in Abbildung 35 lässt sich erkennen, dass das Modell gut bei der Klassifikation von 5 von 7 Emotionen funktioniert. Nur bei Ekel (Klasse 1) erkannte das Modell mit 61% Wut (Klasse 0). Eine richtige Klassifikation von Ekel fand nicht statt, da auch hier Klasse 1 nie als Ergebnis des Modells ausgegeben wurde. Klasse 4 wurde zu 34% richtig klassifiziert, es wurde allerdings fälschlicherweise zu 47% Klasse 5 ausgegeben.

Der folgende Test, des „verdeckten Augen Modells“, auf dem „verdeckte Augen Datensatz“ erreichte eine Test-Accuracy von 59,75%. Das Modell ist demnach besser darin, Bilder mit verdeckten Augen als Bilder ohne Verdeckung zu klassifizieren. Man erkennt, dass das Modell besser im Klassifizieren eines Datensatzes ist, wenn es auf der gleichen Manipulation trainiert wurde.

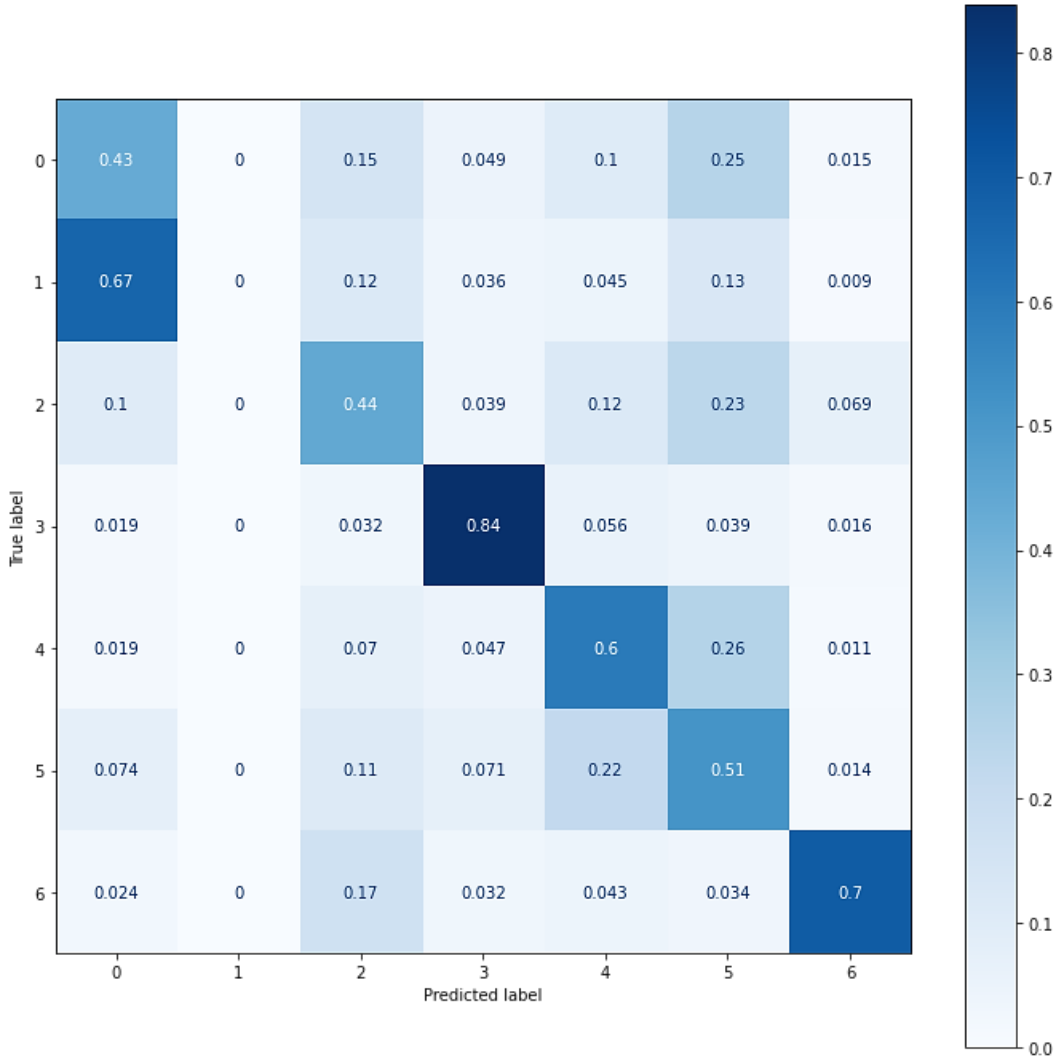

Abbildung 35: Confusion Matrix Eye-Occlusion-Modell Eye-Occlusion-Test

In der Confusion Matrix (Abbildung 36) ist eine sehr ähnliche Verteilung wie bei Abbildung 35 zu erkennen. Der größte Unterschied ist bei Klasse 4 zu erkennen, welche nun mit 60% deutlich besser klassifiziert wurde als zuvor mit 34%. Klasse 1 wurde auch hier nie vorhergesagt. Die Genauigkeiten von Klasse 3 und 6 waren mit 84% und 70% besonders hoch. Daraus kann man schließen, dass für die Erkennung von Angst und Überraschung die Augenregionen nicht notwendig sind. Da die Genauigkeit bei beiden dieser Klassen im Vergleich zum Basismodell-Test auf FER2013 steigt (siehe Abbildung 22), lässt sich dennoch folgern, dass in der Augenregion Informationen zu diesen beiden Emotionen zu finden sein müssen. Auch hier lässt sich jedoch im Vergleich zu Abbildung 35 erkennen, dass das Modell, welches mit Verdeckung der Augen trainiert wurde, durch die Anwesenheit dieser schlechter darin wird Angst und

Überraschung zu erkennen. Man könnte also sagen, das Modell wurde durch die Informationen der Augenregion verwirrt.

Der nächste Test verwendete den Datensatz mit verdeckter Mundregion und erzielte eine Testgenauigkeit von 19,26%. Dieses Ergebnis könnte vermutlich auch durch Raten erzielt werden, und war deutlich schlechter als die vorhergegangenen Tests. Dies ist auch in der Confusion Matrix zu erkennen:

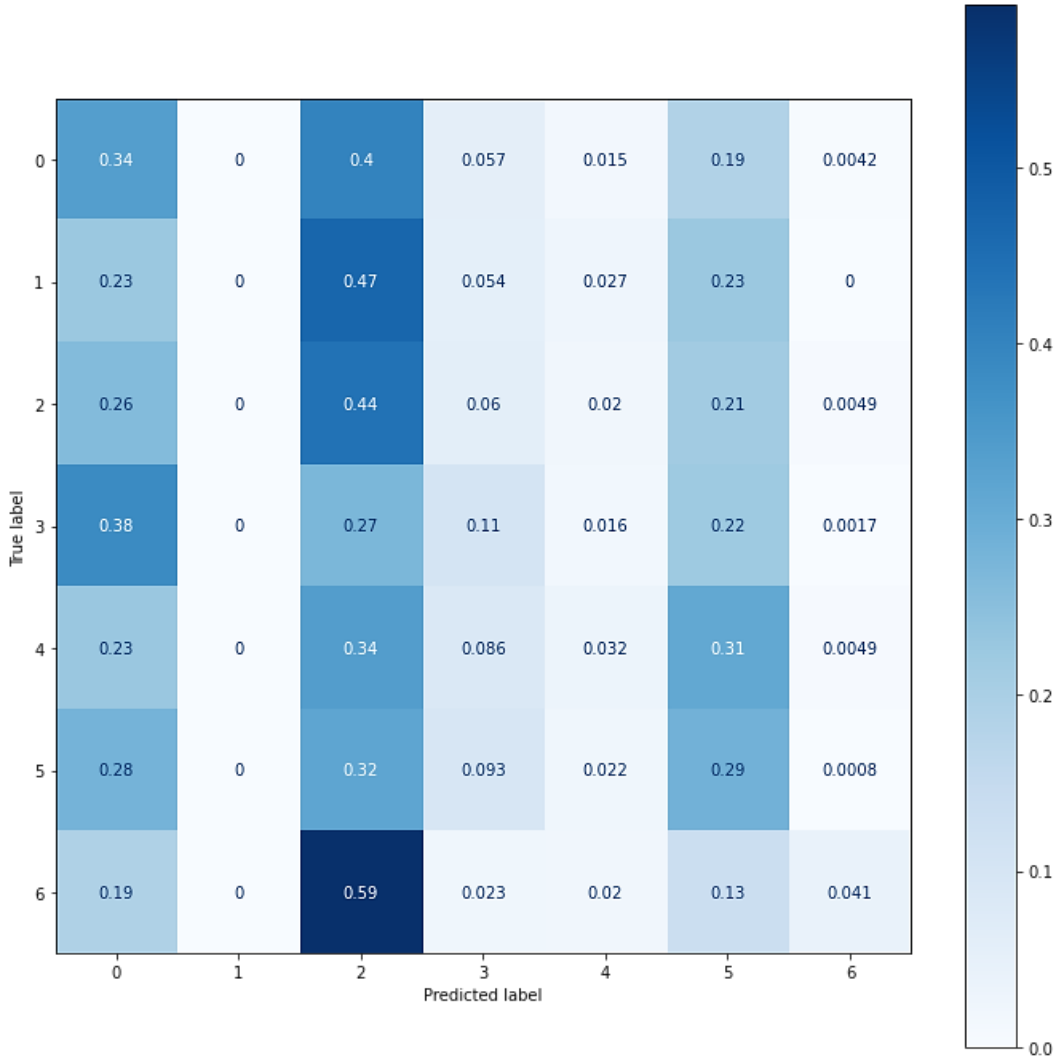

Abbildung 36: Confusion Matrix Eye-Occlusion-Modell Mouth Occlusion-Test

Man sieht an den dunklen vertikalen Linien in Abbildung 37, dass fast alle Vorhersagen des Netzes in Klasse 0, 2 oder 5 lagen. Klasse 1 wurde unverändert nicht erkannt oder vorhergesagt, allerdings wurden jetzt auch Klassen 3, 4 und 6 kaum noch als Output ausgegeben. Die richtigen Vorhersagen des Modells in Klasse 0, 2 und 5 lassen sich dadurch erklären, dass diese Klassen fast immer ausgegeben wurden. Dadurch und an der hellen Hauptdiagonalen lässt sich erkennen, dass das Modell fast nur noch „geraten hat“, zuvor bereits erkennbar an der Test-Accuracy . Das Modell war demnach kaum in der Lage diesen Datensatz richtig zu klassifizieren. Eine mögliche Erklärung hierfür ist, dass die wichtigsten Informationen zu FER in den Augen- und Mundregionen liegen. Da bei diesem Test ein Modell verwendet wurde, dem die

Informationen der Augenregion beim Training fehlten und der zu evaluierende Datensatz die Mundregion ausließ, hatte das Modell keine Möglichkeit die Emotionen richtig zu erkennen.

Als letzter Test dieses Modells folgt der „Salt and Pepper Test“. Die Testgenauigkeit lag bei 40,69%. Abbildung 38 zeigt die entsprechende Confusion Matrix.

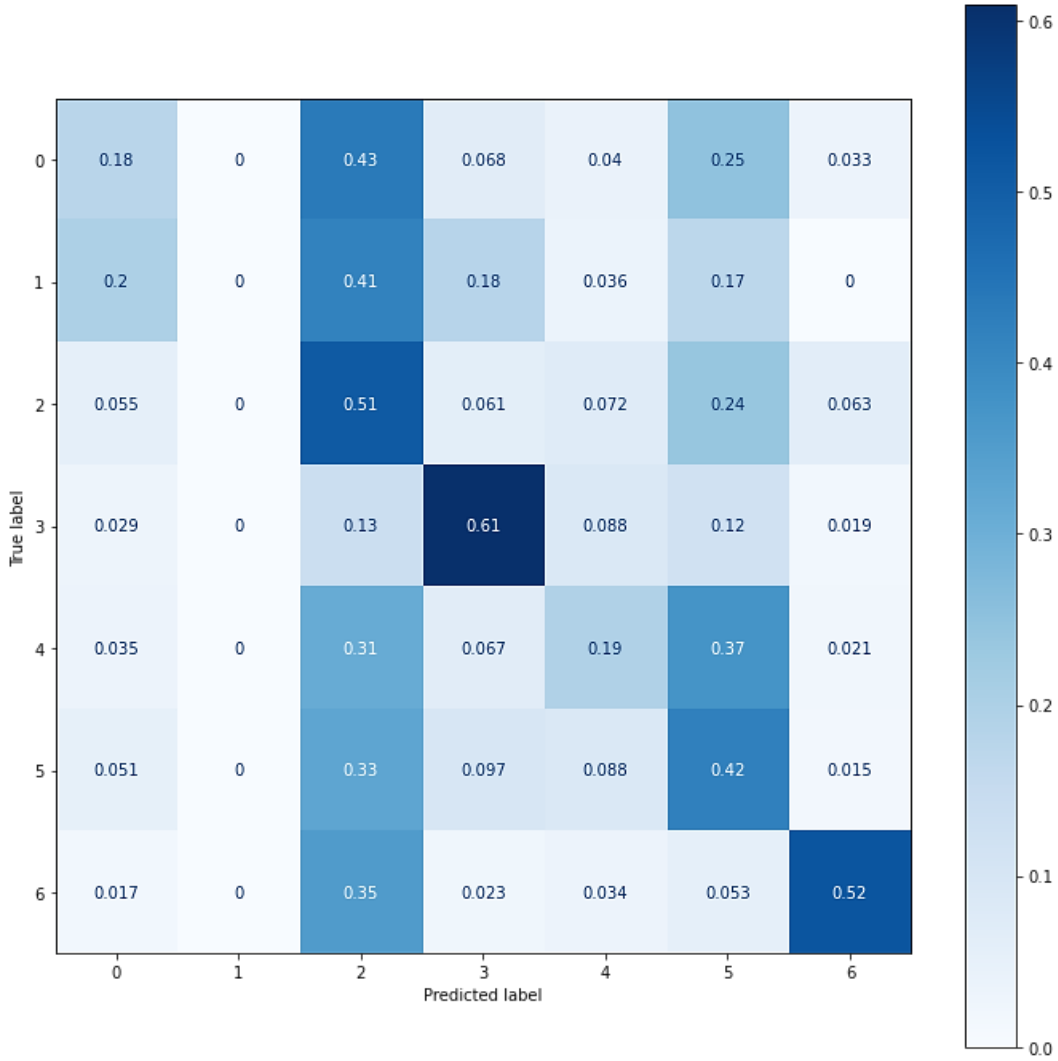

Abbildung 37: Confusion Matrix Eye-Occlusion-Modell SnP-Test

Man erkennt eine starke „Präferenz“ für Klasse 2 des Modells Dadurch sollte die Genauigkeit von 51% bei Klasse 2 mit Vorsicht betrachtet werden. Auch Klasse 5 wurde oft ausgegeben. Am besten wurde erneut Klasse 3 mit 61% Genauigkeit erkannt, und Klasse 6 wurde mit 52% ebenfalls oft richtig klassifiziert. Auch in diesem Test war es dem Modell nicht möglich Klasse 1 zu erkennen und gab dieses nie als Ergebnis aus. Im Vergleich zu Abbildung 36 nimmt die Genauigkeiten bei Angst und Überraschung deutlich ab. Wo es dem Modell beim Eye-Occlusion-Test noch möglich war Angst mit einer Genauigkeit von 84% zu klassifizieren, sind es beim SnP-Test nur noch 61%. Da die Modelle bis zu diesem Punkt immer besser darin waren Datensätze mit Manipulationen zu klassifizieren, welche sie im Training gelernt haben, liegt die Vermutung nahe, dass ein Modell welches sowohl Eye-Occlusion als auch SnP-Noise im Training

zur Verfügung hat, in diesem Test besser abgeschnitten hätte als es in Abbildung 38 der Fall ist.

4.5.2 Testen des Mouth-Occlusion-Modells

Als nächstes folgt das Modell, welches auf dem Datensatz mit der verdeckten Mundregion trainiert wurde. Die Testgenauigkeit auf FER2013 beträgt 50,5%.

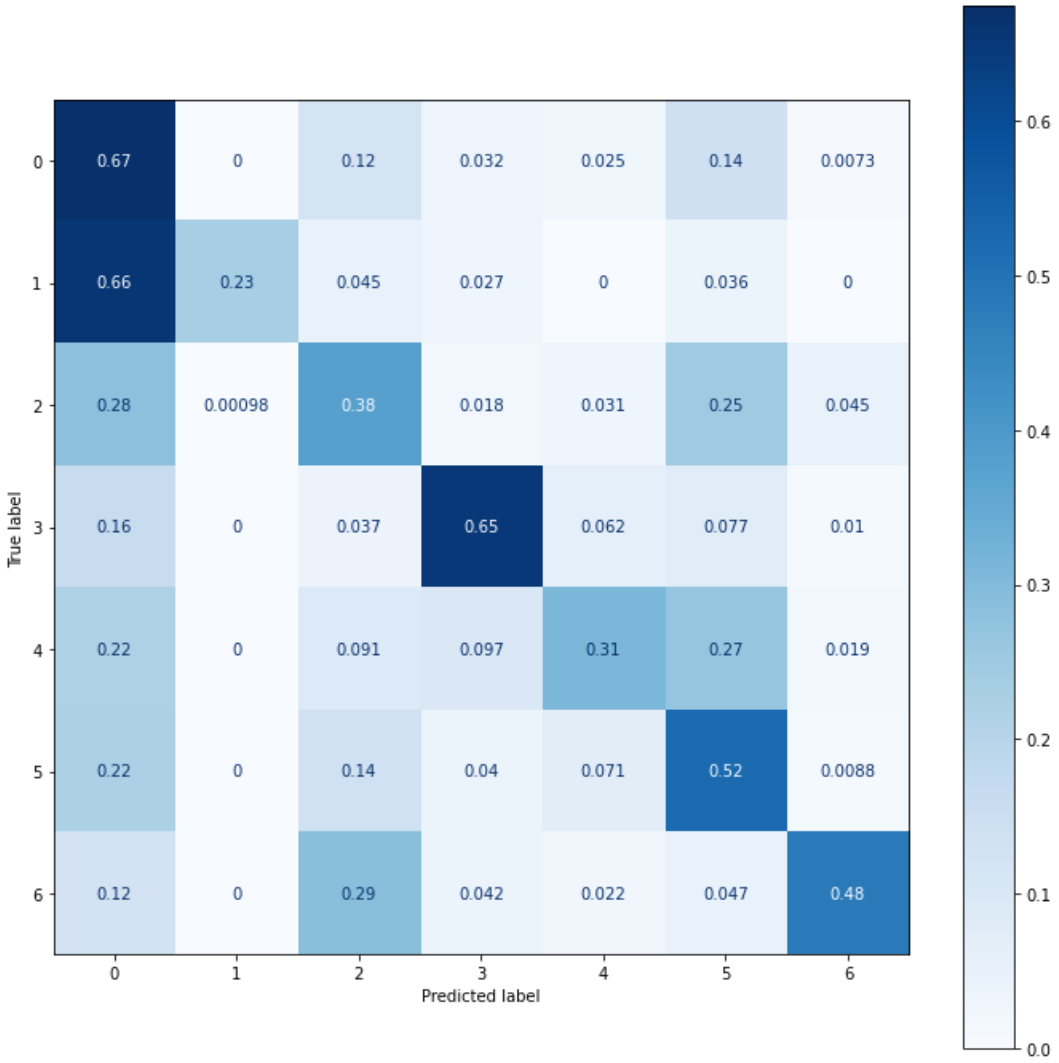

Abbildung 38: Confusion Matrix Mouth-Occlusion-Modell FER2013-Test

In Abbildung 39 fällt sofort auf, dass mit 23% zum ersten Mal Klasse 1 richtig klassifiziert wurde. Zwar wurde auch hier mit hoher Wahrscheinlichkeit Klasse 0 als Ergebnis ausgegeben, dennoch ist die Tatsache, dass Klasse 1 überhaupt in den Vorhersagen vorkam, bemerkenswert. Dies lässt sich möglicherweise darauf zurückführen, dass der Mund beim Lernen von Ekel stört.

Noch deutlicher wurde dies im nächsten Test auf dem „verdeckte Mundregion Datensatz“, welcher mit 61,92% Testgenauigkeit nur ungefähr 4,46% hinter dem ursprünglichen Test des Basismodells auf FER2013 liegt.

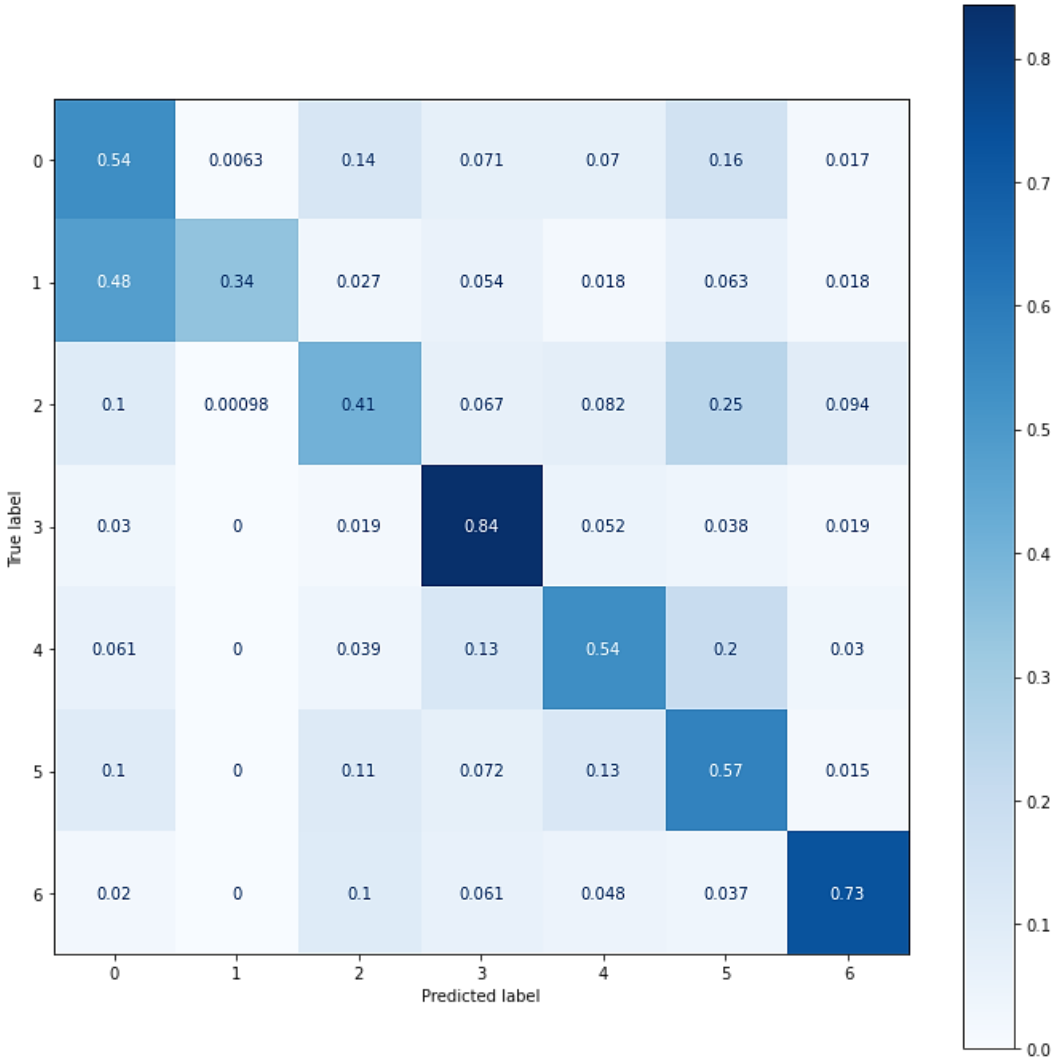

Abbildung 39: Confusion Matrix Mouth-Occlusion-Modell Mouth-Occlusion-Test

In Abbildung 40 erkennt man, dass Klasse 1 sogar mit 34% richtig klassifiziert wurde. Dies bestärkt die Vermutung, dass der Mund beim Klassifizieren von Ekel stört: wurde der Mund sowohl bei Trainings- und Testdaten verdeckt, wurde Ekel (Klasse 1) am besten erkannt. Auch fällt die ansonsten sehr dunkle Hauptdiagonale auf. Bei jeder Klasse, mit Ausnahme von Klasse 1, wurde am häufigsten die richtige Emotion klassifiziert.

Ein anderes Ergebnis lieferte der Test auf dem „Eye Occlusion Datensatz“. Mit einer Testgenauigkeit von 25,14% lag er deutlich hinter den vorherigen Tests dieses Modells (Abbildung 41).

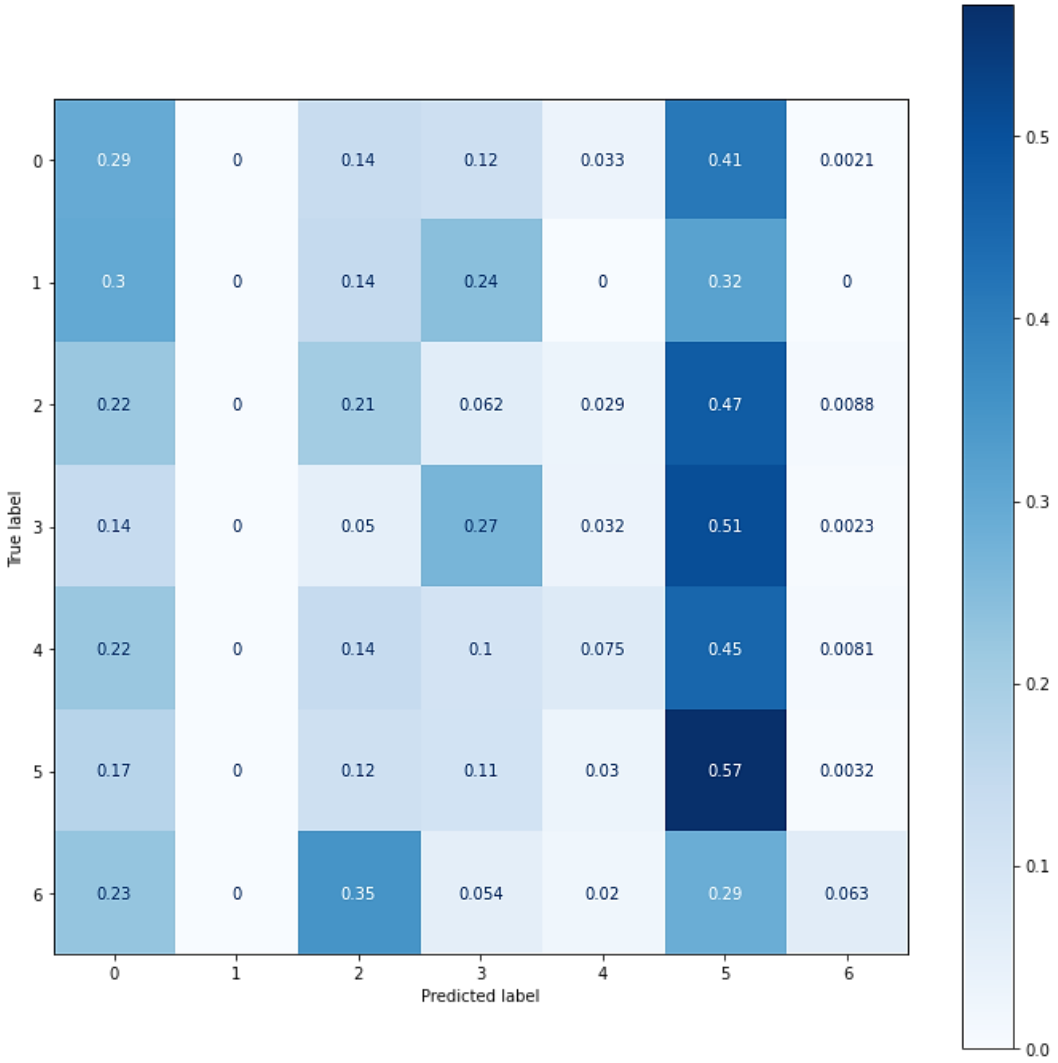

Abbildung 40: Confusion Matrix Mouth-Occlusion-Modell Eye-Occlusion-Test

Die sehr dunkle Vertikale von Klasse 5 lässt erkennen, dass das Modell mit einer Wahrscheinlichkeit von 29% - 57% bei jeder Klasse die, Klasse 5 vorhersagte. Ähnlich, allerdings weniger stark ausgeprägt, sieht die Vertikale von Klasse 0 aus. Klasse 6 wurde kaum noch als Ergebnis ausgegeben, und Klasse 1 wurde wieder nicht vorhergesagt. An der „hellen“ Hauptdiagonalen erkennt man wie schwer es dem Mouth-Occlusion-Modell fiel, den Eye-Occlusion-Datensatz zu klassifizieren.

Als letztes folgte noch der Test auf Salt and Pepper (Abbildung 45 im Anhang). Mit einer Testgenauigkeit von nur 23,77% lag er noch hinter dem letzten Test. In der Confusion Matrix sieht man, warum diese so niedrig war. Wie schon in Abbildung 38 sieht man hier sehr dunkle Vertikalen. Anders als bei Abbildung 38 ist jedoch Klasse 2 eine „präferierte“ Ausgabe des Modells.

4.5.3 Testen des Salt-and-Pepper-Noise-Modells

Als letztes Modell betrachteten wir das auf dem „Salt and Pepper Noise Datensatz“ trainierte Model. Mit einer Testgenauigkeit von 55,67% auf FER2013 ergab sich folgende Confusion Matrix:

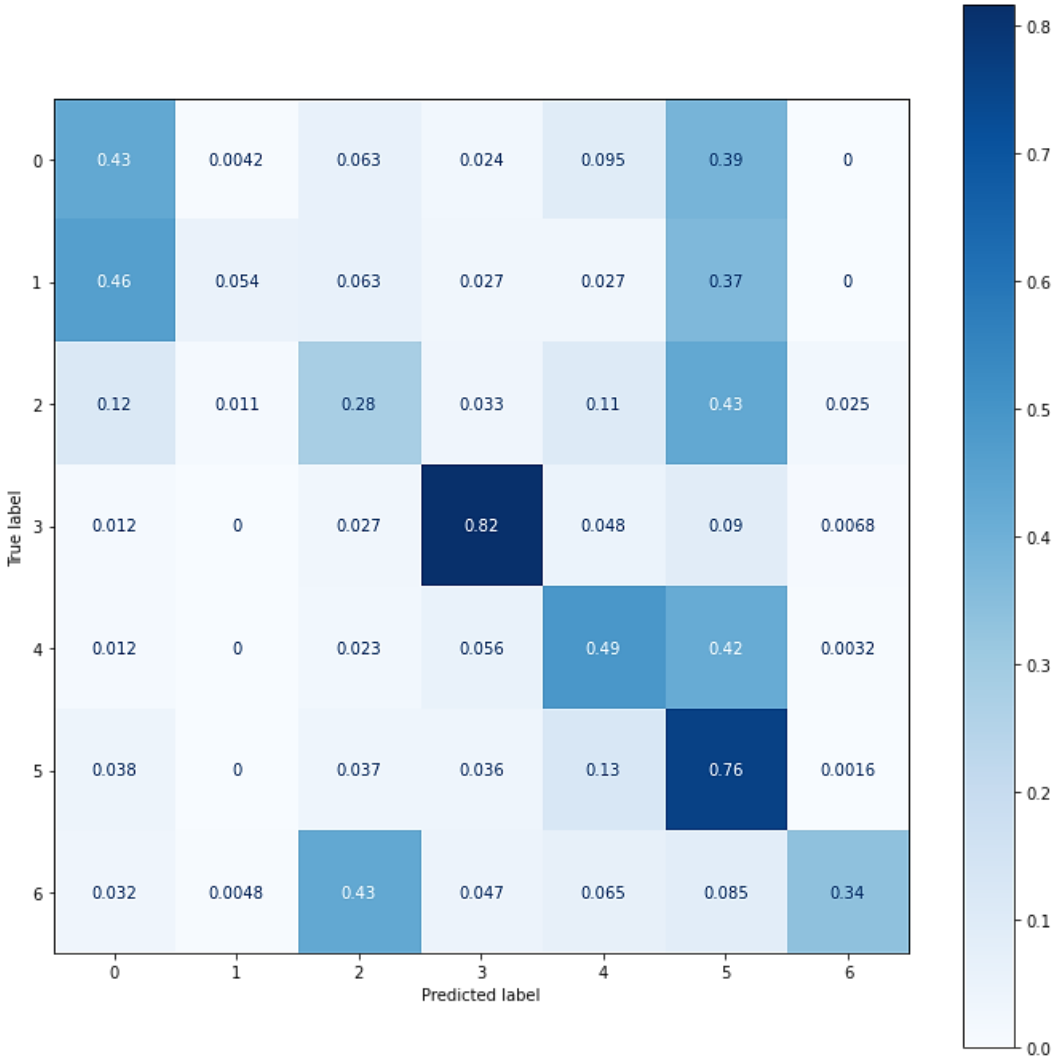

Abbildung 41: Confusion Matrix SnP-Modell FER2013-Test

Gut klassifiziert wurde Klasse 3 mit einer 82% Wahrscheinlichkeit, und Klasse 5 mit einer von 76% (Abbildung 42). Auch hier wurde bei Klasse 6 zu 43% fälschlicherweise Klasse 2 vorausgesagt. Wieder war zu beobachten, Klasse 1 nur zu 5,4%, richtig klassifiziert wurde, was beim Basismodell nicht der Fall war. Bei Klasse 5 fällt die dunkle Vertikale auf: irrte sich das Modell, wurde oft Klasse 5 als Ergebnis ausgegeben.

Zum Vergleich betrachten wir als nächstes den Test auf dem „SnP Testdatensatz“ (Abbildung 47 im Anhang). Mit einer Test Accuracy von 62,37% schnitt auch dieses Modell besser ab, wenn es auf einem Datensatz getestet wurde, welcher eine Manipulation aufwies, mit der es trainiert wurde. Es zeigte sich in ein ähnliches Bild wie in Abbildung 42. Die Hauptdiagonale ist allerdings „dunkler“ was bei einer höheren Test-Accuracy zu erwarten ist. Beispielsweise wurde Klasse 6 mit 72% deutlich besser klassifiziert als die 34% in Abbildung 42. Allerdings gab es auch Klassen, die beim nicht

manipulierten FER2013-Test besser erkannt wurden: Klasse 5 hatte im Vergleich zu den 76% in Abbildung 42 nur noch eine Wahrscheinlichkeit von 59% richtig erkannt zu werden. Es entstand also nicht bei jeder Klasse eine Verbesserung durch die Verwendung von SnP-Noise im Test.

Es folgte der Test auf dem „Eye Occlusion Datensatz“. Er schnitt mit einer Testgenauigkeit von 36,39% deutlich schlechter ab als die beiden vorangehenden Tests.

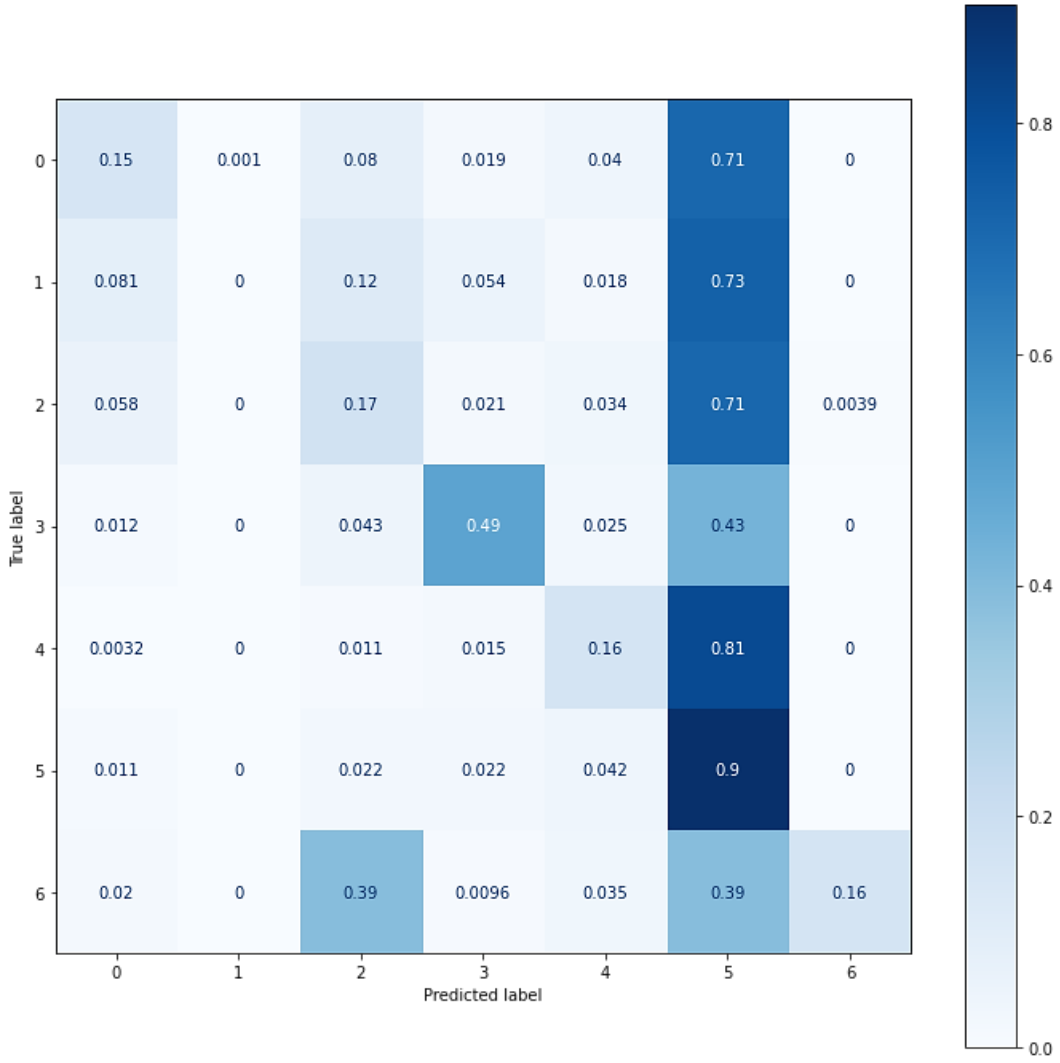

Abbildung 42: Confusion Matrix SnP-Modell Eye-Occlusion-Test

In der Confusion Matrix (Abbildung 43) ist zu erkennen, dass Klasse 5 zwar mit 90% sehr gut klassifiziert wurde, allerdings wurde sie auch bei allen anderen Klassen sehr häufig als Ergebnis ausgegeben. Daher sollten die 90% mit Vorsicht zu betrachten werden. Wenn eine Klasse immer als Ergebnis ausgegeben würde, wäre die Vorhersagegenauigkeit bei dieser Klasse natürlich auch 100%. Nur bei Klasse 3 war die Vorhersage mit 49% ein wenig höher bei der richtigen Ausgabe. Klasse 1, 2, 4 und 6 wurden kaum als Ergebnis ausgegeben.

Der letzte Test (Abbildung 46 im Anhang) des Modells verwendete den „Mouth Occlusion Datensatz“ und lag mit einer Testgenauigkeit von 24,47% nochmals 11,92% hinter dem letzten Test. Anders als beim letzten Test wurde hier nun auch Klasse 3 nicht mehr richtig erkannt. Dafür stieg die Genauigkeit von Klasse 2 ein wenig. Mit 65% wurde nun bei Klasse 6 die Klasse 2 als Ergebnis ausgegeben.

4.6 Auswertung zur Stabilität

In diesem Teil geht es um die Bewertung der Ergebnisse aus Teil 4.5 im Vergleich mit dem Basismodell. Zuerst ist festzustellen, dass alle Modelle unter der Verwendung der manipulierten Datensätze gelitten haben. Ob beim Training oder beim Test nahm die Genauigkeit der Modelle ab, wenn Datensätze mit Störungen verwendet wurden. Allerdings ist bei allen Modellen der Test am besten ausgefallen, der einen Datensatz verwendete, der dem entsprach auf dem trainiert wurde. Beim Basismodell war der beste Test der ohne Manipulationen, beim „Eye Occlusion Modell“ war der beste Test der mit „Eye Occlusion im Testdatensatz“ und so weiter.

In der folgenden Grafik (Abbildung 44) sieht man die Testgenauigkeiten aller Tests von jedem Modell.

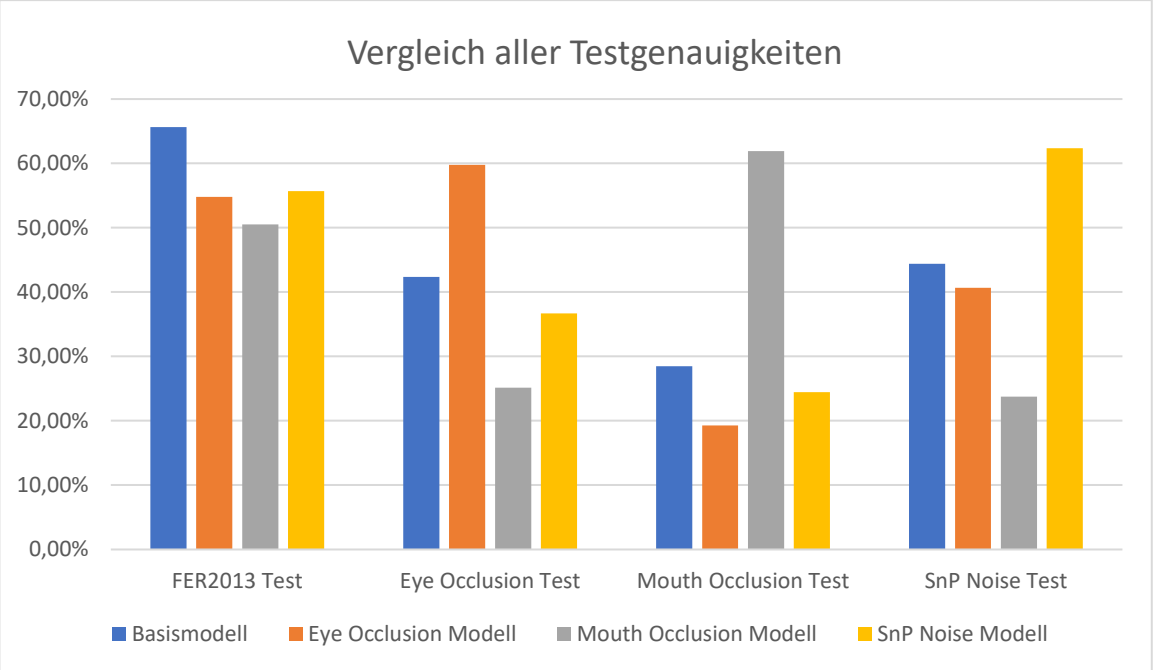

Abbildung 43: Vergleich der Testgenauigkeiten aller Modelle bei allen Tests

Man erkennt, dass jeder Test dann am besten verlief, wenn die Manipulation des Testdatensatze der des Trainingsdatensatzes entsprach. Von allen Modellen schnitt das SnP-Model mit einer durchschnittlichen Testgenauigkeit von 44,8% am besten ab, dicht gefolgt vom Basismodel mit 44,74%. Darauf folgte das „Eye Occlusion Model“ mit einer durchschnittlichen Test-Accuracy von 43,65%. Am schlechtesten schnitt das „Mouth Occlusion Model“ mit einer Genauigkeit von 40,33% ab. Bei den Versuchen dieser Arbeit störte die Verdeckung des Mundes beim Training des Netzes am meisten, während SnP-Noise das Modell sogar verbesserte, auch wenn nur marginal war.

Beim Testen hatten das beste Durchschnittsergebnis die Tests,auf dem nicht manipulierten FER2013-Datensatz mit einer Genauigkeit von 56,66%. Auf dem zweiten Platz lagen die Tests des SnP-Noise Testdatensatzes mit einer durchschnittlichen Testgenauigkeit von 42,81%. Darauf folgte der „Eye Occlusion Test“ mit einer durchschnittlichen Test-Accuracy von 40,99%, und auf dem letzten Platz lagen die Tests des „Mouth Occlusion Testdatensatzes“ mit einer Testgenauigkeit von durchschnittlich 33,54%.

Es lässt sich folgern, dass ein nicht manipulierter Datensatz am einfachsten zu klassifizieren ist. Dieses war zu erwarten, da durch die Manipulationen Informationen verloren gehen. Salt and Pepper Noise mag zwar beim Training nicht von Nachteil sein, beim Testen allerdings schon mit einem durchschnittlichen Testgenauigkeitsverlust von 13,85%. Beim Testen auf den Datensatz, bei dem die Augen verdeckt waren, ergibt sich zwar ein noch schlechteres Ergebnis als beim FER2013- und SnP-Test, allerdings liegt er deutlich höher, wenn der Mund im Testdatensatz verdeckt ist. Es ist also möglich, dass der Mund im Allgemeinen wichtiger für FER ist als die Augen. Diese Hypothese entspricht auch den Ergebnissen aus [26 Seite 31]. Allerdings ist anzumerken, dass das Mouth Occlusion Modell den Mouth Occlusion Datensatz mit einer Testgenauigkeit von 61,92% klassifizierte, was nur 3,73% hinter der Testgenauigkeit vom Basismodell auf dem Basisdatensatz liegt. Es ist dem Netz also möglich zu lernen, einen Datensatz mit Occlusion zu klassifizieren, wenn der Trainingsdatensatz eine ähnliche Störung aufweist.

Interessant wäre es zu testen, wie ein Modell abschneidet, welches in seinen Trainingsdaten alle hier verwendeten Manipulationen verwendet. Außerdem könnte es beim Klassifizieren von Daten hilfreich sein, vor dem Test eine Occlusion-Detection durchzuführen und dann beim Evaluieren der Daten ein Modell zu verwenden, welches auf dieser Occlusion trainiert wurde.

5 Fazit

Nach der Analyse aller Experimente ist das Fazit wie folgt: Es ist möglich mit vortrainierten CNNs Facial Expression Recognition durchzuführen. Selbst nach der Störung von Test und Trainingsdaten war es den Modellen oft möglich, Emotionen in Gesichtern zu erkennen. Es fiel auf, dass Modelle, die mit Trainingsdaten mit bestimmten Manipulationen trainiert wurden, gut darin abschnitten Testdaten mit ähnlichen Störungen zu klassifizieren. Würden in der realen Welt Anwendungen für FER implementiert, wäre es vorteilhaft den verwendeten CNNs Daten zum Trainieren zu geben, die bestimmten Verdeckungen oder Störungen aufweisen. Beispielsweise könnte man bei den Trainingsdaten Bilder verwenden, auf denen die Personen Masken oder Sonnenbrillen tragen. Des Weiteren lassen die Ergebnisse der Experimente in Kapitel 4 vermuten, dass die Region der Augen für die Erkennung von Emotionen weniger wichtig sein könnten als die Mundregion. Hierfür sprechen die Ergebnisse aus [26, Seite 31], denn auch hier stellten die Autoren fest, dass eine Verdeckung der Mundregion zu einem größeren Genauigkeitsverlust führt als die Verdeckung der Augenregion. Die Autoren weisen auf [26, Seite 34] darauf hin, dass die Augenregion nach der Mundregion die zweitwichtigste Region für FER ist. Da in der vorliegenden Arbeit allerdings nur Augen- und Mundregion getestet wurde, kann nur zwischen diesen beiden verglichen werden.

Außerdem war auffällig, dass es den Netzen in den Versuchen der vorliegenden Arbeit schwerer fiel, einige Emotionen zu erkennen als andere. Fast alle hier verwendeten Modelle hatten z.B. größere Schwierigkeiten Ekel in Gesichtern zu entdecken als Glücklichkeit. Auffällig war, dass Modelle, die nicht mit Daten trainiert wurden, welche eine gewissen Manipulation aufweisen, auch Schwierigkeiten hatten Daten mit dieser Manipulation zu klassifizieren.

Um FER in der realen Welt zu einer funktionierenden Technik zu machen, sind vor allem mehr und bessere Datensätze notwendig. Außerdem stellt sich immer noch das Problem, dass manche Emotionsausdrücke in Gesichtern nicht universal bei allen Menschen und Kulturen sind [1]. Solche Emotionen zu klassifizieren bleibt also weiter sehr schwierig.

Literaturverzeichnis:

[1] Thiago H.H. Zavaschi, Alceu S. Britto, Luiz E.S. Oliveira, Alessandro L. Koerich, Fusion of feature sets and classifiers for facial expression recognition, Expert Systems with Applications, Volume 40, Issue 2, 2013, Pages 646-655, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2012.07.074.

[2] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. 2016. Training deep networks for facial expression recognition with crowd-sourced label distribution. In Proceedings of the 18th ACM International Conference on Multimodal Interaction (ICMI '16). Association for Computing Machinery, New York, NY, USA, 279–283. https://doi.org/10.1145/2993148.2993165

[3] S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," in IEEE Transactions on Affective Computing, vol. 13, no. 3, pp. 1195-1215, 1 July-Sept. 2022, https://doi.org/10.48550/arXiv.1804.08348

[4] Khaireddin, Yousif & Chen, Zhuofa. (2021). Facial Emotion Recognition: State of the Art Performance on FER2013. https://doi.org/10.48550/arXiv.2105.03588

[5] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. https://doi.org/10.48550/arXiv.1511.08458

[6] D. Arora, M. Garg and M. Gupta, "Diving deep in Deep Convolutional Neural Network," 2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2020, pp. 749-751, doi: 10.1109/ICACCCN51052.2020.9362907.

[7] A. Özgür and F. Nar, "Effect of Dropout layer on Classical Regression Problems," 2020 28th Signal Processing and Communications Applications Conference (SIU), 2020, pp. 1-4, doi: 10.1109/SIU49456.2020.9302054.

[8] Shuai Tan, Zhi Tan, “Improved LeNet 5 Model Based on Handwritten Numerical Model”, The 31st Chinese Control and Decision Conference, 2019. DOI: 10.1109/CCDC.2019.8833112

[9] L. Pham, T. H. Vu and T. A. Tran, "Facial Expression Recognition Using Residual Masking Network," 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 4513-4519, doi: 10.1109/ICPR48806.2021.9411919

[10] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016. https://doi.org/10.48550/arXiv.1512.03385

[11] Abate, A.F., Cimmino, L., Mocanu, BC. et al. The limitations for expression recognition in computer vision introduced by facial masks. Multimed Tools Appl (2022). https://doi.org/10.1007/s11042-022-13559-8

[12] Simonyan, Karen and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” CoRR abs/1409.1556 (2015): n. pag. arXiv:1409.1556

[13] A. K. Jain, Jianchang Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial," in Computer, vol. 29, no. 3, pp. 31-44, March 1996, doi: 10.1109/2.485891

[14] Wennker, Phil: Künstliche Intelligenz in der Praxis. Springer Gabler, 2020. ISBN 978-3-658-30479-9

[15] SGDR: Stochastic Gradient Descent with Warm Restarts https://arxiv.org/abs/1608.03983

[16] Adam: A Method for Stochastic Optimization https://arxiv.org/abs/1412.6980

[17] https://www.image-net.org/about.php 12.09.2022

[18] Daoming She, Minping Jia, “Wear indicator construction of rolling bearings based on multi-channel deep convolutional neural network with exponentially decaying learning rate” ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2018.11.040

[19] Gu, Jiuxiang, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu et al. "Recent advances in convolutional neural networks." Pattern recognition 77 (2018): 354-377. https://doi.org/10.1016/j.patcog.2017.10.013

[20] M. Wang, S. Lu, D. Zhu, J. Lin and Z. Wang, "A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning," 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2018, pp. 223-226, doi: 10.1109/APCCAS.2018.8605654

[21] https://keras.io/guides/transfer_learning/ 16.09.2022

[22] Picard, Rosalind W. Affective computing. MIT press, 2000. ISBN-13: 978-0262281584

[23] Abraham, Ajith. "Artificial neural networks." Handbook of measuring system design (2005). https://doi.org/10.1002/0471497398.mm421

[24] Verdhan, V. (2021). Image Classification Using LeNet. In: Computer Vision Using Deep Learning. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6616-8_3

[25] E. Pranav, S. Kamal, C. Satheesh Chandran and M. H. Supriya, "Facial Emotion Recognition Using Deep Convolutional Neural Network," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 317-320, doi: 10.1109/ICACCS48705.2020.9074302.

[26] Ligang Zhang, Brijesh Verma, Dian Tjondronegoro, and Vinod Chandran. 2018. Facial Expression Analysis under Partial Occlusion: A Survey. ACM Comput. Surv. 51, 2, Article 25 (March 2019), 49 pages. https://doi.org/10.1145/3158369

[27] https://www.tensorflow.org/api_docs/python/tf/data/Dataset 10.11.2022

[28] Sebastian Raschka & Vahid Mirjalili 2019, “Python Machine Learning, 3rd Edition” ISBN-13: 978-1789955750

[29] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[30] Chollet, Francois. 2021. Deep Learning with Python, 2. Auflage. New York, NY: Manning Publications. ISBN 9781617296864

[31] https://www.tensorflow.org/overview 11.11.2022

[32] https://keras.io/api/preprocessing/image/#image_dataset_from_directory-function 10.11.2022

[33] Ekman, P., & Friesen, W. V. (1971). Constants across cultures in the face and emotion. Journal of Personality and Social Psychology, 17(2), 124–129. https://doi.org/10.1037/h0030377

Abbildungsverzeichnis:

Abbildung 1: Darstellung eines KNNs (adaptiert nach der Grafik in [5] Seite 2.) 2
Abbildung 2: Darstellung eines Neurons (eigene Grafik.) 3
Abbildung 3: Visuelle Repräsentation einer Konvolution, der eingerahmte Bereich entspricht des dem Input. 7
Abbildung 4: LeNet-1 [24, Seite 71] 9
Abbildung 5: VGG Architektur. Adaptiert nach einem Ausschnitt der Grafik 3 [10, Seite 4] 9
Abbildung 6: Residual Block. Adaptiert nach Grafik 2 in [10] 10
Abbildung 7: Vergleich von Labor- zu Real World Bedinungen [3] 11
Abbildung 8: Ablauf der Schritte in Kapitel 3 13
Abbildung 9: Darstellung der verwendeten Netzwerkarchitektur 15
Abbildung 10: Vergleich verschiedener LR Scheduler 16
Abbildung 11: Training Loss VGG16 17
Abbildung 12: Validation Loss VGG16 17
Abbildung 13: Trainings Loss VGG16 Fine Tuning 18
Abbildung 15: Validation Accuracy VGG16 Fine Tuning 18
Abbildung 16: Training Loss ResNet50 18
Abbildung 17: Validation Loss ResNet50 18
Abbildung 18: Validation Accuracy ResNet50 19
Abbildung 19: Training Loss ResNet50 Fine Tuning 19
Abbildung 20: Validation Loss ResNet50 Fine Tuning 19
Abbildung 21: Validation Accuracy ResNet 50 Fine Tuning 19
Abbildung 22: Confusion Matrix für Basismodell FER2013-Test 20
Abbildung 23: Vergleich zwischen CNNs 21
Abbildung 24: Ablauf von Kapitel 4 22
Abbildung 25: Haarcascade Beispiele für Augenerkennung 23
Abbildung 26: Beispiele für die Augenverdeckung 24
Abbildung 27: Beispiele für die Mundverdeckung 24
Abbildung 28: Beispiele für SnP Noise 24
Abbildung 29: Confusion Matrix Basismodell Eye-Occlusion-Test 25
Abbildung 30: Confusion Matrix Basismodell Mouth-Occlusion-Test 26
Abbildung 31: Confusion Matrix Basismodell SnP-Test 27
Abbildung 32: Training Loss Eye-Occlusion 28
Abbildung 33: Validation Loss Eye-Occlusion 28
Abbildung 34: Vergleich von Testgenauigkeiten der „manipulierten Modelle“ auf dem FER2013 Datensatz 28
Abbildung 35: Confusion Matrix Eye-Occlusion-Modell FER2013-Test 29
Abbildung 36: Confusion Matrix Eye-Occlusion-Modell Eye-Occlusion-Test 30
Abbildung 36: Confusion Matrix Eye-Occlusion-Modell Mouth Occlusion-Test 31
Abbildung 38: Confusion Matrix Eye-Occlusion-Modell SnP-Test 32
Abbildung 39: Confusion Matrix Mouth-Occlusion-Modell FER2013-Test 33
Abbildung 40: Confusion Matrix Mouth-Occlusion-Modell Mouth-Occlusion-Test 34
Abbildung 41: Confusion Matrix Mouth-Occlusion-Modell Eye-Occlusion-Test 35
Abbildung 42: Confusion Matrix SnP-Modell FER2013-Test 36
Abbildung 43: Confusion Matrix SnP-Modell Eye-Occlusion-Test 37
Abbildung 44: Vergleich der Testgenauigkeiten aller Modelle bei allen Tests 38
Abbildung 45: Confusion Matrix Mouth-Occlusion-Modell SnP-Test 45
Abbildung 46: Confusion Matrix SnP-Modell Mouth-Occlusion-Test 46
Abbildung 47: Confusion Matrix SnP-Modell SnP-Test 47

ANHANG:

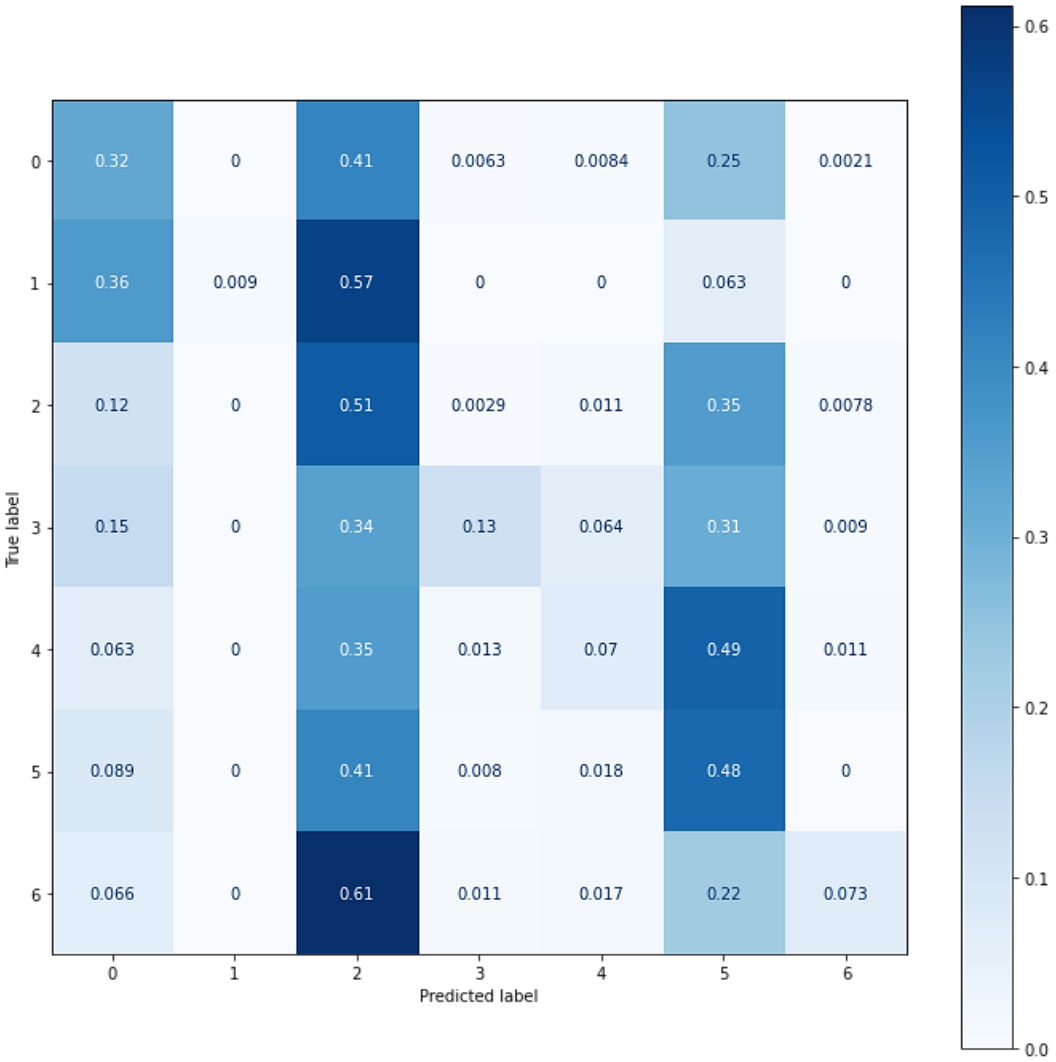

Abbildung 44: Confusion Matrix Mouth-Occlusion-Modell SnP-Test

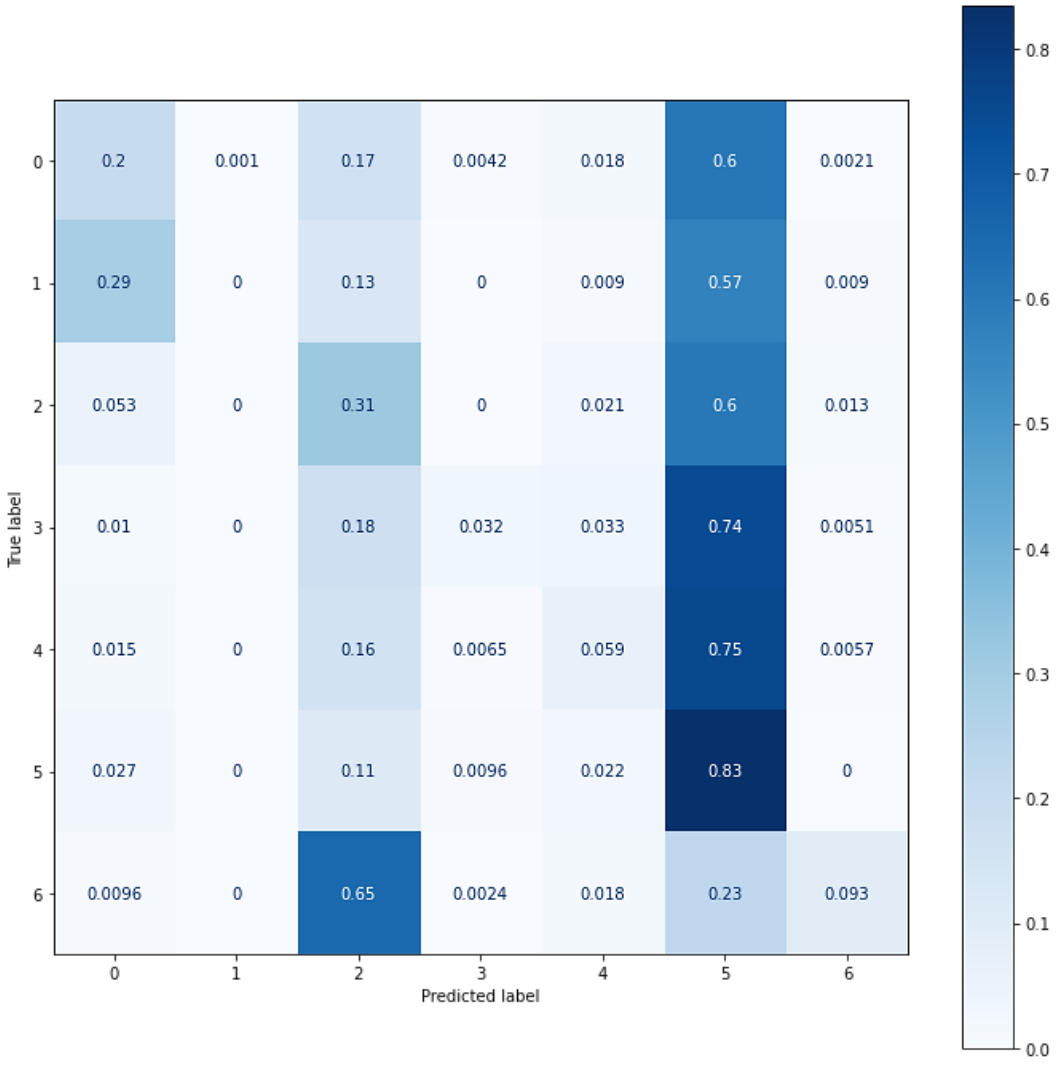

Abbildung 45: Confusion Matrix SnP-Modell Mouth-Occlusion-Test

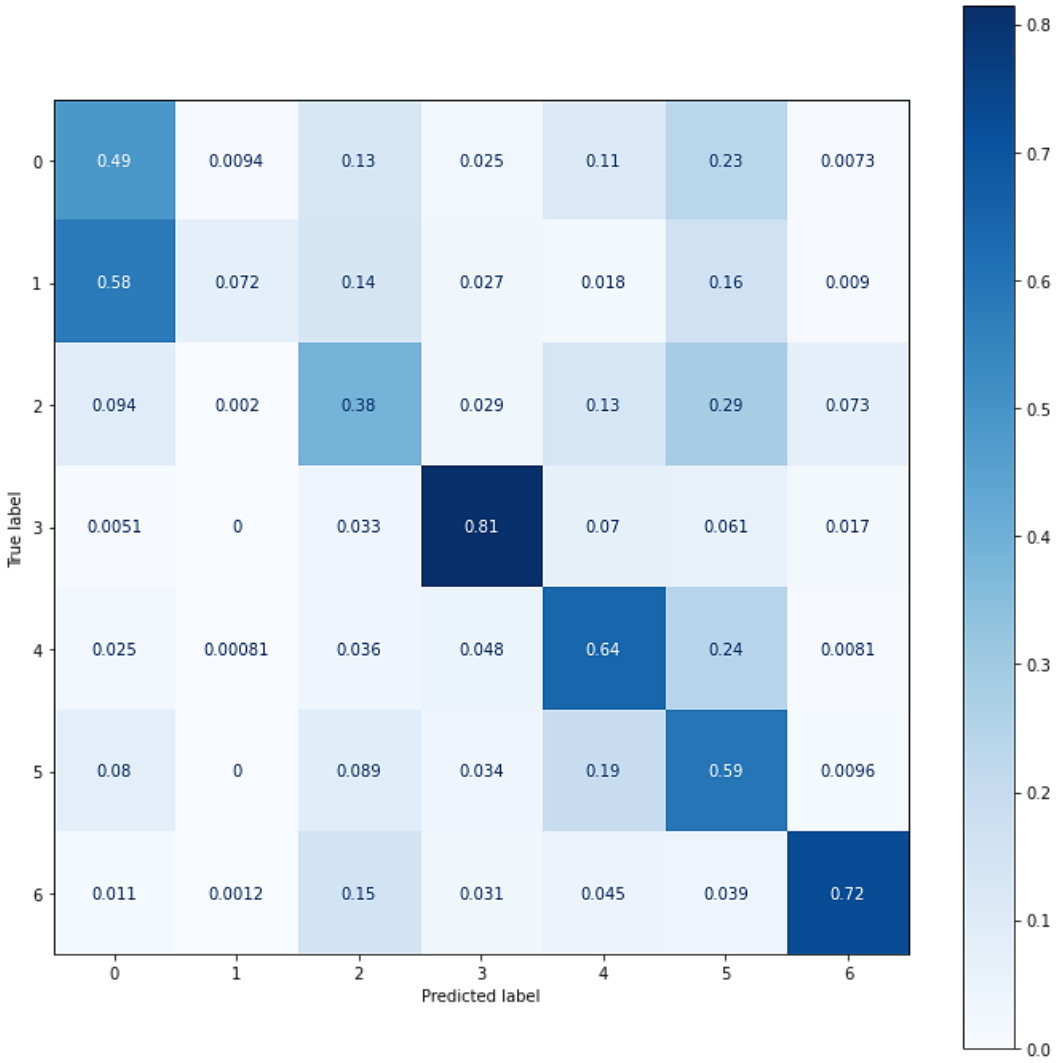

Abbildung 46: Confusion Matrix SnP-Modell SnP-Test

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel:

Facial Expression Recognition mittels verschiedener CNNs im Vergleich mit anschließender Robustheitsprüfung

_____

selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Datum	Unterschrift