

Emotionsanalyse am Beispiel der Pokerfaceerkennung

Studienarbeit

des Studienganges **Angewandte Informatik / Betriebliches Informationsmanagement**
an der Dualen Hochschule Baden-Württemberg Mannheim

von

Fabian Brandmüller, Maximilian Ludwig, Kevin Wrona

8774695

1329241

2356667

26. April 2020

Bearbeitungszeitraum
Betreuer der DHBW

23.09.2019 - 24.04.2019
Prof. Dr. Eckhard Kruse

Erklärung

Wir versichern hiermit, dass die Studienarbeit mit dem Thema: **”Emotionsanalyse am Beispiel der Pokerfaceerkennung”** selbstständig verfasst und keine anderen als die angegeben Quellen und Hilfsmittel benutzt wurden.

Ort	Datum	Unterschrift Fabian Brandmüller
-----	-------	---------------------------------

Ort	Datum	Unterschrift Maximilian Ludwig
-----	-------	--------------------------------

Ort	Datum	Unterschrift Kevin Wrona
-----	-------	--------------------------

Abstract

Author — 1329241

Aufgabe dieser Studienarbeit ist es eine Emotionsanalyse durchführen zu können. Zur Veranschaulichung wird diese Thematik anhand des Anwendungsbeispiels der Pokerface-Erkennung realisiert. Ziel ist es dabei, dass eine möglichst genaue Vorhersage der Emotionen gemacht werden kann, die es wiederum ermöglicht eine Aussage darüber zu treffen, ob eine Person ein Pokerface aufgesetzt hat, oder nicht. Um dies umzusetzen wird eine künstliche Intelligenz entwickelt, genauer gesagt ein neuronales Netz. Die Trainings und Testdaten dieses Netzes sind dabei 35887 verschiedene Gesichtsbilder der FER (Facial Expression Recognition) Challenge von Kaggle. Analysiert werden die Emotionen "Happiness", "Sadness", "Fear", "Surprise", "Anger", "Neutral". Mittels dieser Daten kann das trainierte Modell eine Genauigkeit in der Emotionsvorhersage von 55,71% erzielen. die Rate der richtig erkannten Pokerfaces zu testen ist jedoch nicht so einfach, weshalb hierzu keine Daten in dieser Arbeit ermittelt wurden. Lediglich durch Ausprobieren ist es möglich die Genauigkeit der Pokerface-Erkennung festzustellen. Das Erkennen eines Pokerfaces ist mit der hier entwickelten Lösung jedoch möglich genauso wie eine Verwendung in der Praxis.

Inhaltsverzeichnis

1	Einleitung	1
2	Anforderungen	4
2.1	Aufgabenstellung	4
2.2	Usecase	4
2.3	MoSCoW Priorisierung	5
3	Stand der Technik	7
3.1	Gesichtserkennung	7
3.2	Emotionserkennung	8
3.2.1	Emotionen	9
3.2.2	Abgrenzung zur Gesichtserkennung	9
3.3	Wann ist ein Gesicht ein Pokerface?	10
3.4	Emotionserkennung mithilfe von Deep Learning	11
3.4.1	Machine Learning - Frameworks	11
3.4.1.1	Dlib	12
3.4.1.2	Keras	12
3.4.1.3	Unterschiede	13
3.4.2	Supervised vs. Unsupervised Learning	13
3.4.3	Aktivierungsfunktion	14
3.5	Gesichtserkennung mit OpenCV	16
3.5.1	Opencv Klassifizierer	16
3.5.1.1	HAAR Cascade Klassifizierer	16
3.5.1.2	Gesichtserkennungs Algorithmen	18
3.5.2	Eingabe Daten	19
4	Stand-Alone Lösung mit Schwerpunkt OpenCV	22
4.1	Konzept	22
4.1.1	Interaktionskonzept	22
4.1.2	Architektur	23
4.1.2.1	Programmierumgebung	23

4.1.2.2	Hardware	24
4.2	Umsetzung	24
4.2.1	Input GUI	24
4.2.2	Dataset	28
4.2.3	Training	30
4.2.4	Testing	31
4.2.5	Optimierung der Lösung	32
5	Server-Client Lösung	33
5.1	Konzept	33
5.1.1	Programmierumgebung	33
5.1.2	Hardware	34
5.2	Umsetzung	34
5.2.1	Dataset	35
5.2.2	Modell	37
5.2.3	Trainieren des Modells	38
5.2.4	Testen des Modells	40
5.2.5	Optimierung des Modells	41
5.2.6	Webserver	41
5.2.7	Client	42
5.2.7.1	Langzeit-orientierter Algorithmus	43
5.2.7.2	Kurzzeit-orientierter Algorithmus	46
6	Ergebnis	48
7	Diskussion	51
7.1	Reflexion der Ergebnisse	51
7.2	Reflexion Vorgehen	54
7.3	Reflexion der Literatur	56
7.4	Offene Implikationen	57
7.5	Alternative Ansätze zur Umsetzung von Emotionserkennung	57

Abbildungsverzeichnis

C.A Phasen der Gesichtserkennung	8
C.B ReLu Funktionsgraph	15
C.C haar features	17
C.D RGB Kanäle	20
D.A GUI mit Video Stream der Webcam als Output	25
D.B GUI mit Video Stream der Webcam und markierten Gesichtern als Output	26
D.C GUI mit Video Stream der Webcam und markierten Gesichtern und Augen als Output	27
D.D Emotionsverlauf von Neutral zu Disgust	28
D.E Struktur Emotionsverlauf	29
E.A Häufigkeitsverteilung der Emotionen	36
E.B Visualisierung der aufgezeichneten Daten des History-Objektes	39
E.C Programmablaufplan des Client-Programms	44
G.A Bearbeitetes Bild mittels Histogram Equalization	54
G.B Verzeichnisstrukturen des Cohn-Kanade Datasets	72
G.C Struktur überarbeitetes Dataset	73
G.D Model Summary with several Layers and I/O Shapes	74

Tabellenverzeichnis

IV.I Emotionen mit jeweiliger Anzahl an Bildern	28
V.I Struktur des CSV-Datasets der FER-Challenge	36
V.II Zuordnung und Häufigkeiten der Emotionen	36
VI.I Genauigkeit der Zuordnung der Testdaten zu den einzelnen Emotionen	48

Listings

4.1	Einbinden eines vortrainierten OpenCV-Modells zur Gesichtserkennung	25
4.2	Emotions-Array	30
7.1	Code für GUI mit Video Stream der Webcam als Output	62
7.2	Code für GUI mit Video Stream der Webcam und markierten Gesichtern als Output	63
7.3	Code für GUI mit Video Stream der Webcam und markierten Gesichtern und Augen als Output	64
7.4	Dataset sortieren und überarbeiten	66
7.5	Trainingsdaten als Arrays extrahieren	68
7.6	Skript zum Trainieren und Testen eines FisherFaceRecognizer	69
7.7	Klassifizierer mit der Input GUI testen	71

Kapitel 1

Einleitung

Author — 1329241

”Several researchers have stated that facial expression recognition appears to play one of the most important roles in human communication”¹ Dieses Zitat von Katherine B. Leeland gibt einen Einblick in die Relevanz der Emotionserkennung für den Menschen. Fragen zu dieser Thematik stellen sich allerdings nicht erst seit Beginn der Digitalisierung. Bereits der Evolutionsforscher Charles Darwin fragte sich, ob von den Gesichtsausdrücken einer Person nicht auch der emotionale Zustand abgeleitet werden kann.² Einen solchen Zustand von einem Mitmenschen mittels Software abzulesen, ist jedoch nicht leicht zu realisieren. Bereits durch kleine Änderungen in der Mimik werden verschiedene Emotionen ausgedrückt. Zum Beispiel, indem eine Person bei Wut die Lippen zusammenpresst und die Augen zusammenkniff oder bei Trauer die Mundwinkel nach unten zieht.³ Durch derartige Ausdrücke können Emotionen wie Wut oder Trauer Ausdruck gewinnen. Es gibt bereits Emotionserkennungssoftware, welche unter anderem in der Wirtschaft eingesetzt wird. Die Anwendungsgebiete reichen dabei von Jobinterviews, in denen analysiert wird, inwieweit die Bewerber für den jeweiligen Job geeignet sind,⁴ bis hin zur Automobilindustrie. Mittels geeigneter Sensorik wird dort versucht, die Emotion und somit den physiologischen Zustand des Autofahrers zu analysieren.⁵ Diese Daten legen den Grundbaustein für Warnsysteme, welche den Fahrer darauf hinweisen können, dass sein Zustand zum Betrieb eines Kraftfahrzeugs ungeeignet ist. Solche Einsatzszenarien werden jedoch auch durchaus kontrovers diskutiert. Auf Kritik stößt unter anderem, dass die sogenannten ”Basisemotionen” - z.B. Wut, Trauer, Ekel, Freude, Furcht, Überraschung - , die verwendet werden, um den KIs

¹Vgl. Leeland, *Face Recognition: New Research*, S. 1.

²Vgl. ebd., S. 2.

³Vgl. Li, *Handbook of Face Recognition*, S. 249.

⁴Vgl. Schreiner, *Künstliche Intelligenz: Emotionserkennung will gelernt sein*.

⁵Vgl. Diederichs, *KI-gestützte Emotionserkennung im Fahrzeug aus physiologischen Daten*, Herausforderung.

Emotionserkennung beizubringen, selbst umstritten sind.⁶ Vor allem ethische Bedenken werden im Bezug auf die Anwendungsgebiete zunehmend geäußert. Denn je nach Emotion, die erkannt werden soll, liegt die Fehlerrate sehr hoch. So hat das Fraunhofer Institut, welches am Einsatz von Emotionserkennung in Fahrzeugen arbeitet, festgestellt, dass eine Emotionserkennung je nach Zielemotion eine Vorhersagekraft zwischen 6% und 95% haben kann.⁷ Diese negativen Aspekte treffen jedoch nur teilweise auf das hier behandelte Forschungsprojekt zu, wie im Folgenden dargelegt werden soll: Basierend auf den zuvor genannten Basisemotionen Wut, Frucht, Trauer, Freude und Ekel soll in dieser Arbeit getestet werden, inwieweit eine technische Vorhersage der Emotionen mittels künstlicher Intelligenz möglich ist. Dies erfolgt am Anwendungsbeispiel der Pokerface-Erkennung. Mittels der hier entworfenen technischen Lösung soll daher getestet werden, inwiefern ein noch zu definierendes Pokerface erkannt werden kann. Diese Forschungsarbeit hat also das Ziel, Emotionen so genau wie möglich zu analysieren und durch geeignete Algorithmen zu ermitteln, ob ein Pokerface vorhanden ist oder nicht. Der aus dieser Arbeit hervorgehende Prototyp ist dabei jedoch theoretisch gesehen nicht an das hier verwendete Fallbeispiel des Pokerspielens gebunden. Die Grundidee dieses Prototypen lässt einige hypothetische Einsatzszenarien in der Praxis zu. Diese haben eine gewisse Schnittmenge mit denen von "normaler" Emotionserkennungssoftware, jedoch gibt es auch einige weitere. Im Folgenden werden einige denkbare Szenarien expliziert:

- Polizeiverhöre

Es ist denkbar, dass eine erweiterte Form des entwickelten Prototypen bei Polizeiverhören eingesetzt werden könnte. Hierdurch könnten Beamte die Anwendung eines Pokerfaces durch den Beschuldigten erkennen, was auf eine Lüge hinweisen würde. Neben der gebräuchlichen Verwendung eines Lügendetektors wäre der Einsatz, der in dieser Arbeit erstellten Lösung, eine kostengünstige Variante.

- Gerichtsverhandlungen

Das zweite Einsatzgebiet ist ähnlich zu dem Ersten. Bei Gerichtsverhandlungen gelten die gleichen Voraussetzungen wie bei einem Verhör der Polizei. Zwar müssen die Vorgeladenen eine eidesstattliche Erklärung abgeben, welche beinhaltet, nur die Wahrheit zu sagen, jedoch ist zu bezweifeln, ob dies auch immer der Fall ist. Nun soll nicht der Eindruck entstehen, dass das hier gebaute Werkzeug ein Lügendetektor ist. Es ist ebenfalls nicht möglich, dass von einem Pokerface immer auf eine Lüge geschlossen werden kann. Jedoch ist ein Pokerface ein Zeichen dafür, dass sich diese Person ihren

⁶Vgl. Kuhn, *Emotionen sind schwer definierbar*.

⁷Vgl. Diederichs, *KI-gestützte Emotionserkennung im Fahrzeug aus physiologischen Daten*, Ergebnis.

emotionalen Zustand nicht anmerken lassen möchte. Und dies wiederum deutet eher darauf hin, dass die Person nicht oder nur teilweise die Wahrheit sagt.

- Pokerspiel

Wie bereits erwähnt, ist dieses Einsatzszenario das Fallbeispiel dieser Arbeit. Dies liegt unter anderem daran, dass der erste Begriff, der mit dem Wort Pokerface - bzw. einem emotionslosen Gesichtsausdruck - in Verbindung gebracht wird, das Pokerspiel selber ist. Und auch in diesem kann es nützlich sein zu wissen, ob die Kontrahenten ein Pokerface aufsetzen oder nicht. Denkbar wäre es, dass ein Mitspieler zum Beispiel mittels einer Kamera das Gesicht des Gegenübers scannt und analysiert, ob ein Pokerface vorliegt oder nicht. Aufgrund der gewonnenen Erkenntnis kann er dann entsprechend reagieren.

Kapitel 2

Anforderungen

Das nun folgende Kapitel thematisiert die konkrete Aufgabenstellung der Arbeit sowie eine Anforderungsanalyse mittels MoSCoW-Priorisierung.

2.1 Aufgabenstellung

Author — 1329241

Das Projekt selber wird an der DHBW in Mannheim unter der Betreuung von Prof. Dr. Eckhard Kruse durchgeführt. Wie eingangs erwähnt, soll mittels künstlicher Intelligenz erkannt werden, welche Emotion vorliegt. Zur Umsetzung dieser Aufgabe wird eine Bilderkennungssoftware angefertigt, welche ein übermitteltes Bild anhand vorhandener Emotionen analysiert. Sollte durch die Software keine Emotionsänderung erkannt werden, wird dem Anwender das Vorhandensein eines Pokerfaces zurückgegeben. Wie bereits in der Einleitung beschrieben, sind diverse Einsatzmöglichkeiten vorhanden, an die das Werkzeug leicht angepasst werden kann.

2.2 Usecase

Author — 1329241

Wie bereits erwähnt, ist der hier behandelte Use Case das Pokerspiel selber. An diesem soll ermittelt werden, inwieweit sich Emotionen oder das Abhandensein von Emotionen vorhersagen lassen. Dieses Fallbeispiel wurde gewählt, da es unter anderem das Nahe liegendste ist und ein simples und leicht zu konstruierendes Szenario impliziert. Dabei soll mithilfe einer Kamera ein Bild von einem Pokerspieler aufgenommen werden und dann mittels dem hier entworfenen Prototypen verarbeitet und analysiert werden. Am

Ende soll dann eine Vorhersage der Emotionen getätigt werden, die dem Photographen zur Verfügung gestellt wird.

2.3 MoSCoW Priorisierung

Author — 1329241 - 8774695 - 2356667

Diese Arbeit soll methodisch mit der MoSCoW-Priorisierung bearbeitet werden. Diese Art der Priorisierung teilt die zu bearbeitenden Anforderungen in vier Kategorien ein:¹

- Must - Core Anforderungen, die unbedingt umgesetzt werden müssen
- Should - Anforderungen, die ebenfalls umgesetzt werden müssen, jedoch im Nachhinein noch durch Change Requests verändert werden können
- Could - Anforderungen, die nach den Must und Should Anforderungen umgesetzt werden sollen, sofern noch Ressourcen und Zeit vorhanden sind, um diese zu bearbeiten
- Won't - Anforderungen, die nicht in diesem Projekt bzw. Release erfolgen, jedoch in einer zukünftigen Version bearbeitet werden können

Im Zuge des Projektes wurden die vorhandenen Anforderungen anhand der MoSCoW-Priorisierung wie folgt eingeordnet:

- Must
 - Emotionen werden nicht zufällig erkannt
 - Es wird zwischen Pokerface und kein Pokerface unterschieden
 - Benutzerfreundliche Interaktionsmöglichkeit, um Input zu liefern und Output darzustellen
- Should
 - Die Genauigkeit zur Erkennung der richtigen Emotion soll über 50% liegen
 - Es sollen mindestens fünf verschiedene Emotionen erkannt werden
 - Die Erkennung einer Emotion und eines Pokerfaces soll innerhalb des Pokerspielzuges eines Spielers (ca. 20 Sekunden) erfolgen
- Could

¹vgl. Urs und Roger, *Handbuch Projektmanagement: Agil – Klassisch – Hybrid*, S. 90.

- Die Emotionen aus Bildern können nahezu in Echtzeit analysiert werden
 - Das Trainieren des Modells kann innerhalb eines Arbeitstages erfolgen
 - Eine Oberfläche zur intuitiven Bedienung der Lösung ohne technisches Verständnis muss vorhanden sein
- Won't
 - Neben dem Analysieren von Bildern ist auch die Analyse von Videos möglich
 - Das Modell soll bei jeder Belichtung valide Ergebnisse liefern

Dabei sollen die einzelnen Anforderungen entsprechend ihrer Priorität abgearbeitet werden. So kann am Ende der Erfolg der Arbeit deutlich besser eingeordnet werden.

Kapitel 3

Stand der Technik

Author — 1329241

In diesem Abschnitt soll der aktuelle Forschungs- und Entwicklungsstand im Bereich Emotionserkennung thematisiert werden. Jedoch muss dafür erst einmal eine Unterscheidung der Begrifflichkeiten Emotionserkennung und Gesichtserkennung erfolgen, da beide Gebiete Überschneidungen haben, jedoch inhaltlich und von ihren Zielen verschieden sind. Innerhalb dieser Arbeit wird sich beiden Themenbereichen bedient, weshalb zumindest eine grobe Erläuterung der Vorgehensweise dieser Ansätze sinnvoll ist.

3.1 Gesichtserkennung

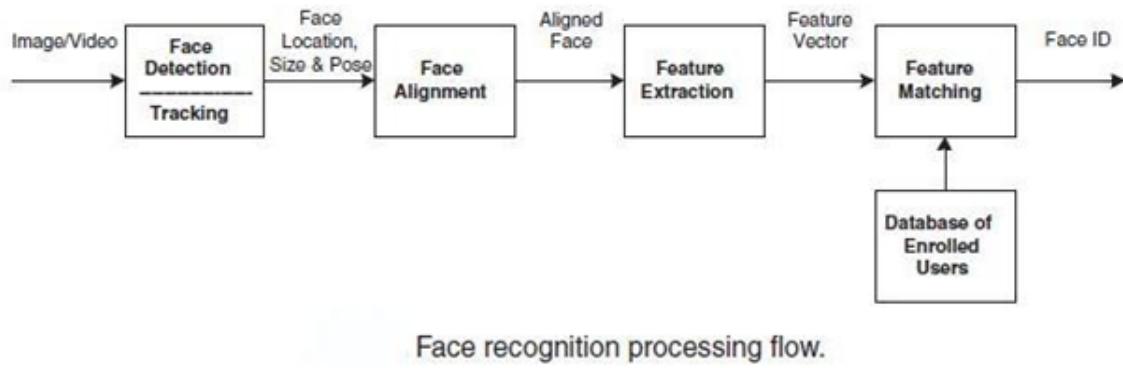
Author — 1329241

Gesichtserkennung ist eine Disziplin der Informatik in der es darum geht Gesichter wieder zu erkennen, und gegebenenfalls verschiedenen Personen zuzuordnen. Dabei lässt sich der Prozess der Gesichtserkennung in vier Phasen einteilen, Face "detection", "alignment", "feature extraction" und "matching", wie anhand von Grafik C.A sichtbar ist.¹ Die Detection Phase ist dafür verantwortlich, zu erkennen, ob Gesichter vorhanden sind in einem Bild oder aber Video.² In der darauffolgenden Alignment-Phase hingegen wird die Lokalisierung der Gesichter genauer, indem Gesichtskomponenten wie Augen, Augenbrauen, oder die Nase genauer lokalisiert werden. Dabei wird das Bild oder Video ebenfalls normalisiert, indem z.B. die Bildbeleuchtung angepasst wird.³ In der Feature extraction hingegen werden die verschiedenen Gesichtskomponenten wie Augen, Nase, Mund, dem Bild oder Video entnommen. Dies ist ein wichtiger Schritt für weitere

¹Vgl. Li, *Handbook of Face Recognition*, S. 2.

²Vgl. ebd., S. 2.

³Vgl. ebd., S. 2.



Quelle: <https://alitarhini.files.wordpress.com/2010/12/untitled1.png>
 Abbildung C.A: Phasen der Gesichtserkennung

Prozesse wie Eye Tracking oder Face Tracking. Alternativ kann sogar eine bestimmte Person anhand der extrahierten Merkmale erkannt werden.⁴ In der letzten Phase, dem Matching, geht es darum, die gewonnenen Daten mit den in der Datenbank vorhandenen Gesichtern abzugleichen. Wenn eine genügende Übereinstimmung gefunden wurde, wird ein Match mit einer Person ausgegeben.⁵ Die Anwendungsbereiche von Software, die Gesichtserkennung ermöglicht, sind mannigfaltig. Sie reichen von Applikationen die ein Gerät wie ein Smartphone entsperren, wenn das Gesicht des Besitzers als Treffer ausgegeben wurde, bis hin zur Anwendung in Verbrechensbekämpfung. In jedem dieser Szenarien wird dabei der oben beschriebene Ablauf durchgegangen und abhängig vom zu liefernden Ergebnis eine Abschlussaktion vorgenommen.

3.2 Emotionserkennung

Author — 1329241

In diesem Unterkapitel nun sollen Emotionen an sich thematisiert werden, da diese maßgeblich sind für das zu entwickelnde Tool. Eine Definition von Emotionserkennung ist per se nicht schwer zu geben. Prinzipiell beschäftigt sich Emotionserkennung mit der Analyse von Gesichtern und den Emotionen, die diese Gesichter darstellen. Jedoch ist der Begriff der Emotionen nicht ganz so einfach zu definieren, wie im Folgenden erläutert wird:

⁴Vgl. Saranya und Kumar, *Geometric shaped facial feature extraction for face recognition*, Abstract.

⁵Vgl. Li, *Handbook of Face Recognition*, S. 3.

3.2.1 Emotionen

Author — 1329241

Grundsätzlich gibt es verschieden Ansätze Emotionen zu definieren und einzuteilen. Eine Variante ist dabei die eingangs erwähnte, nicht ganz unumstrittene Einteilung in Basisemotionen. Eine gängige Einteilung ist dabei die verschiedenen Emotionen in acht Bereiche einzuteilen. Diese Einteilung wurde 1984 von Plutchik postuliert und beinhaltet die Emotionskategorien Angst, Wut, Freude, Trauer, Akzeptanz, Ekel, Erwartung und Überraschung.⁶ Jedoch ist dies nicht die einzige mögliche Einteilung. Als weiteres Beispiel teilt MacLean die Emotionen in lediglich sechs Kategorien ein, welche da wären: Verlangen, Wut, Angst, Niedergeschlagenheit, Freude und Zuneigung.⁷ Wie sich bereits an den beiden Beispielen zeigt, geht die Meinungen der Forscher dabei stark auseinander, welche und wie viele Emotionen zu den sogenannten "Basis Emotionen" gehören. In dieser Arbeit werden die Emotionen in sechs Kategorien eingeteilt, in Wut, Verachtung, Angst, Freude, Trauer, Überraschung und Neutral. Diese Einteilung entspricht an sich keiner gängigen Einteilung, jedoch wurde diese aus den folgenden Gründen gewählt:

Die hier genannten Emotionen lassen sich gut anhand von Bildern erlernen, da diese zum Teil komplementär und somit eindeutig sind. Es ist aber auch einfacher Testdatensätze zu bekommen für ein freudiges Gesicht oder ein überraschtes, als ein Gesicht mit dem emotionalen Ausdruck Akzeptanz. Zudem sind die gewählten Emotionen häufig bei dem Test Usecase dieser Arbeit anzutreffen, dem Texas Hold'em Poker.

3.2.2 Abgrenzung zur Gesichtserkennung

Author — 1329241

Der grundlegende Unterschied zwischen Emotions- und Gesichtserkennung liegt nun darin, dass bei der Emotionserkennung selber nicht die agierende Person im Vordergrund steht, sondern die Aktion die sie ausführt. Bei der Gesichtserkennung hingegen spielt lediglich die Rolle wer eine Aktion ausführt und ob es einen Treffer in der Datenbank gibt, oder nicht. Wegen dieser Unterschiede ist auch die technische Realisierung eines Prototypen, vor allem im Bezug auf die Architektur, durchaus unterschiedlich. Dies ist jedoch ebenso von den unterschiedlichen Anwendungsszenarien der beiden Verfahren bedingt. Denn diese sind ebenso verschieden. Während Gesichtserkennung eher in den Bereich IT-Security oder aber Social Media (Snapchat Filter) eingesetzt wird, ist

⁶Vgl. Leeland, *Face Recognition: New Research*, S. 3.

⁷Vgl. ebd., S. 3.

Emotionserkennung eher Informationsgenerierend. Zum Beispiel können durch Emotionserkennung Informationen zugänglich werden, wie das Befinden eines Individuums ist, ob es emotional betroffen ist, oder aber nicht emotional betroffen wirken möchte und ein Pokerface aufsetzt. Wegen dieser signifikanten Unterschiede kann daher trotz der Gemeinsamkeiten nicht gesagt werden, dass Emotionserkennung eine Unterkategorie von Gesichtserkennung ist.

3.3 Wann ist ein Gesicht ein Pokerface?

Author — 1329241

Bis zu diesem Teil der Arbeit wurde schon immer wieder das Wort Pokerface verwendet ohne es formal zu definieren. Jedoch ist es für die Arbeit wichtig, eine konkrete Definition für diesen Begriff zu geben, da ansonsten die Anforderungen an die Arbeit nicht korrekt bewertet werden könnten. Da es mehrere Ansätze gibt ein Pokerface zu definieren, wird dies in diesem Abschnitt nachgeholt. Zum einen gibt es den trivialen Ansatz, dass ein Pokerface dann vorliegt, wenn ein Subjekt keinerlei Emotion in seinem Gesicht zu erkennen gibt. Dies würde einer neutralen Emotion entsprechen. Dieser Fall kann auch vergleichsweise einfach behandelt werden, da diese neutrale Emotion auch in die Basisemotionen mit aufgenommen wurde in dieser Arbeit. Das bedeutet, dass beispielsweise ein neuronales Netz auch diese Arten von Gesichtern klassifizieren könnte.

Der zweite Ansatz ist weniger intuitiv, jedoch ebenso wichtig. Es kann durchaus angenommen werden, dass z.B. ein sehr erfahrener Pokerspieler weiß, dass man ein Pokerface einfach identifizieren kann, und demnach alleine dadurch Informationen gewinnen kann. Um dem entgegenzuwirken könnte ein solcher Spieler eine andere Art Pokerface verwenden. Es wäre denkbar, dass er versucht sein Gesichtsausdruck permanent gleich zu halten, indem er eben dauerhaft lacht oder lächelt, dauerhaft betrübt, wütend, ängstlich, etc. aussieht. Dieser Ansatz ist weniger intuitiv als der erste und könnte demnach auch weniger schnell erkannt werden. Im Verlaufe der Arbeit wird mit der zweiten Definition gearbeitet.

3.4 Emotionserkennung mithilfe von Deep Learning

Author — 8774695

In dem nun folgenden Kapitel wird erörtert, wie das Ziel des Prototypen dieser Arbeit - das Erkennen eines Pokerfaces - mittels eines neuronalen Netzes umgesetzt werden kann. Dabei wird weniger auf die generellen Eigenschaften von neuronalen Netzen Bezug genommen, als auf die in dieser Arbeit spezifischen Aspekte. Diese sind vor allem verschiedene Ansätze und Möglichkeiten mittels Machine Learning eine Emotionserkennungssoftware zu erstellen.

3.4.1 Machine Learning - Frameworks

Author — 8774695

Um die gegebene Aufgabenstellung der Erkennung von Emotionen mittels der Analyse eines Gesichtes umsetzen zu können, musste ein entsprechendes Framework Anwendung finden, welches den Anforderungen gerecht wird. Maschinelles Lernen gehört in der heutigen Softwareentwicklung zu den beliebtesten Themen, wodurch dieses schnelle, regelmäßige Änderungen und Weiterentwicklungen erfährt. Dementsprechend werden auf dem Markt auch zahllose kostenlose wie auch kostenpflichtige Frameworks angeboten. Darunter fallen unter anderem OpenCV, Tensorflow oder Pandas. Möchte man nun das geeignete Framework für das eigene Projekt ausfindig machen, muss man das gegebene Angebot nach einigen Kriterien filtern. Als erstes stellt sich die Frage, was für eine Art von Applikation man umsetzen möchte. Soll das Projekt Texte analysieren oder wie im Falle dieses Projektes die Emotionen aus einem gegebenen Bild? Welche Programmiersprache wird innerhalb des Projektes eingesetzt? Zudem sind Informationen zur Lizenz und dem Support wichtig, sowie insbesondere die Community.

Nach entsprechender erster Selektion musste sich schlussendlich zwischen Dlib und Keras entschieden werden, welche für die Umsetzung der Anforderungen dieses Projektes am besten geeignet schienen. Beide Frameworks werden in den folgenden beiden Unterkapitel dem Leser kurz vorgestellt und anschließend wird ein Fazit gezogen, welches der beiden für dieses Projekt am geeignetsten ist.

3.4.1.1 Dlib

Author — 8774695

Dlib ist nach dessen Entwickler Davis King ein modernes C++ Toolkit, welches Machine-Learning Algorithmen und Tools zur Entwicklung komplexer Software enthält, um Probleme aus der echten Welt lösen zu können.⁸

Dlib selbst wurde in der Programmiersprache C++ entwickelt und kann durch eine Anbindung auch für Python Projekte eingesetzt werden. Die Software-Bibliothek kann unter den Bedingungen der Boost-Lizenz frei genutzt werden und ist durch entsprechende APIs des jeweiligen Betriebssystems plattformunabhängig. Ein weiterer Vorteil bietet die Unabhängigkeit der Bibliothek von anderen Bibliotheken. Dlib ist seit dem Jahr 2002 in Entwicklung und bietet dementsprechend unzählige Features an, welche für verschiedenste Einsatzgebiete verwendet werden können, wie numerische und graphische Modell-Algorithmen und vor allem Gesichtserkennung.⁹ Eines der größten Vorteile dieser Bibliothek besteht in der ausführlichen Dokumentation für sämtliche Klassen und Funktionen, wie es nicht häufig der Fall ist für vergleichbare Open-Source Projekte.¹⁰

3.4.1.2 Keras

Author — 8774695

Die Open-Source Bibliothek Keras wurde im Vergleich zu Dlib im Jahre 2015 von dem Google-Programmierer François Chollet in der Programmiersprache Python entwickelt.¹¹ Das ursprüngliche Ziel von Keras war es eine Abstraktionsschicht für das Framework Theano zu bilden, um die Handhabung zu vereinfachen. Mit der Veröffentlichung von TensorFlow im November 2015 wurde Keras im folgenden Monat weiter entwickelt um auch TensorFlow als Backend verwenden zu können. Schlussendlich konnte Keras die Frameworks TensorFlow, Theano und das Microsoft Cognitive Toolkit (kurz: CNTK) einbinden und beliebig austauschen.¹² Allerdings stoppte im September 2017 die Weiterentwicklung an Theano¹³ waraufhin Keras zunehmend an TensorFlow gebunden wurde.¹⁴ Schlussendlich wurde mit Keras 2.3, welches im September 2019

⁸Opala, *Top Machine Learning Frameworks Compared: Scikit-Learn, Dlib, MLlib, Tensorflow, and More*, Vgl.

⁹Vgl. ebd.

¹⁰*Dlib C++ Library*, Vgl.

¹¹Vgl. Stefan Luber, *Was ist Keras?*

¹²Vgl. *Keras backends*.

¹³Vgl. *Announcement of Theano's discontinuation*.

¹⁴Vgl. *TensorFlow 1.4.0*.

veröffentlicht wurde, die multi-backend Version eingestellt.¹⁵ Aus diesem Grund wird Keras überwiegend mit dem Framework TensorFlow verwendet.

Bei der Entwicklung von Keras stand die Benutzerfreundlichkeit als eines der wichtigsten Prinzipien im Mittelpunkt. Dem gefolgt bietet Keras eine schnelle und leichte Erstellung neuronaler Netze. Durch ein ausführliches Feedback können Fehler durch den Benutzer schnell ausfindig gemacht und behoben werden. Ein weiteres Designprinzip von Keras ist die Modularität, mithilfe derer man verschiedene Module beliebig konfigurieren und miteinander kombinieren kann. Zudem bietet die Open-Source Bibliothek die Unterstützung von rekurrenten und konvolutionalen Netzwerken, sowie die Unterstützung der GPUs und CPUs, was für die Umsetzung dieses Projektes von Vorteil ist.¹⁶ Zuletzt steht hinter Keras eine große aktive Community, welche zahlreiche Tutorials und Hilfestellungen für verschiedenste Problemstellungen liefert.

3.4.1.3 Unterschiede

Author — 8774695

Zusammenfassend kann gesagt werden, dass Dlib eine mächtige Bibliothek für die Umsetzung verschiedenster Aufgaben ist. Die ausführliche Dokumentation bietet eine große Hilfestellung. Jedoch folgt aus dem Angebot der zahlreichen Features auch eine längere Einarbeitung, bis die Anforderungen produktiv umgesetzt werden können.

Keras hingegen bietet durch dessen benutzerfreundliche Entwicklung und den zahlreichen Tutorials eine schnelle erste Umsetzung der Aufgaben. Durch die Modularität und der aktiven Community können viele öffentliche Lösungen genutzt und für die eigenen Bedürfnisse abgeändert werden. Aus diesem Grund und aufgrund der Unterstützung der rekurrenten und konvolutionalen Netzwerken wurde sich für die Verwendung von Keras als Open-Source Bibliothek in diesem Projekt entschieden.

3.4.2 Supervised vs. Unsupervised Learning

Author — 1329241

In diesem Abschnitt werden die beiden Ansätze des Supervised bzw. des Unsupervised Learnings evaluiert. Dabei sollen jedoch beide Begriffe nicht noch ausführlich beleuchtet werden. Es ist lediglich zu erwähnen, dass Supervised Learning Algorithmen im Gegensatz zu Unsupervised mit Datensätzen arbeiten, die kategorisiert sind. Damit ist es ihnen möglich Bilder die eingegeben werden anhand dieser Kategorien zu

¹⁵Vgl. *Keras 2.3.0*.

¹⁶Vgl. *Einführung in Keras*.

klassifizieren. Grundlegende Informationen zu den einzelnen Vorgehensweisen können aus den hier zitierten Büchern entnommen werden.¹⁷¹⁸ Nun gilt es zu klären, ob sich für die zugrundeliegende Aufgabe ein Supervised oder Unsupervised Ansatz eher anbietet. Ein Unsupervised Learning Algorithmus würde sich vor allem anbieten, wenn Zusammenhänge zwischen einzelnen Datensätzen gefunden werden sollen, die vielleicht nicht von Anfang an bekannt oder bewusst sind.¹⁹ Supervised Learning Algorithmen hingegen bieten sich vor allem an, wenn es darum geht einen Prozess zu automatisieren oder aus der Wirklichkeit zu replizieren.²⁰ Das in dieser Arbeit zugrundeliegende Problem ist demnach vor allem für Supervised Learning Algorithmen geeignet. Dies liegt zum einen daran, dass jedem Bild eines Menschen eine Basisemotion zugeordnet werden kann. Zum anderen ist auch der verwendete Datensatz, mit dem das Modell trainiert und getestet werden soll, ebenfalls gelabelt. Deshalb bietet sich dieses Verfahren am meisten an. Es wäre auch hypothetisch denkbar, einen Unsupervised Learning Algorithmus zu verwenden, aber dieser Ansatz wäre suboptimal verglichen mit einem Supervised Algorithmus.

3.4.3 Aktivierungsfunktion

Author — 8774695

Im Folgenden wird kurz auf die sogenannten Aktivierungsfunktionen eines neuronalen Netzwerkes eingegangen. Dabei wird dem Leser lediglich eine kurze Einführung gegeben, weshalb diese Funktionen für Machine Learning verwendet werden sollten und weniger auf die dahinterliegende Mathematische Problemstellung, welche den Komplexitätsrahmen dieser Arbeit überschreiten würde.

Aktivierungsfunktionen werden für ein kompliziertes nichtlineares Zuordnen der Eingangsdaten und deren abhängigen Ergebnisse verwendet. In der einfachsten Mathematischen Darstellung kann man sich also die Eingangsdaten \mathbf{X} und deren Gewichtung \mathbf{W} vorstellen, auf welchen die Aktivierungsfunktion $f(\mathbf{x})$ angewendet wird. Sollte bei der Berechnung eines neuronalen Netzes die Aktivierungsfunktion nicht explizit bestimmt werden, wird die einfachste Lineare Funktion $f(\mathbf{x})=\mathbf{x}$ verwendet, welche von ihrer Komplexität derart begrenzt ist, dass kein komplexes Funktionsmapping aus den Daten erlernbar ist. Daraus folgt, dass ein neuronales Netzwerk ohne Aktivierungsfunktion keine komplizierten Datenarten, wie zum Beispiel Bilder nutzen kann, welche für die Umsetzung der Aufgabe natürlich verwendet werden müssen.²¹

¹⁷Vgl. Johnston und Mathur, *Applied Supervised Learning with Python*.

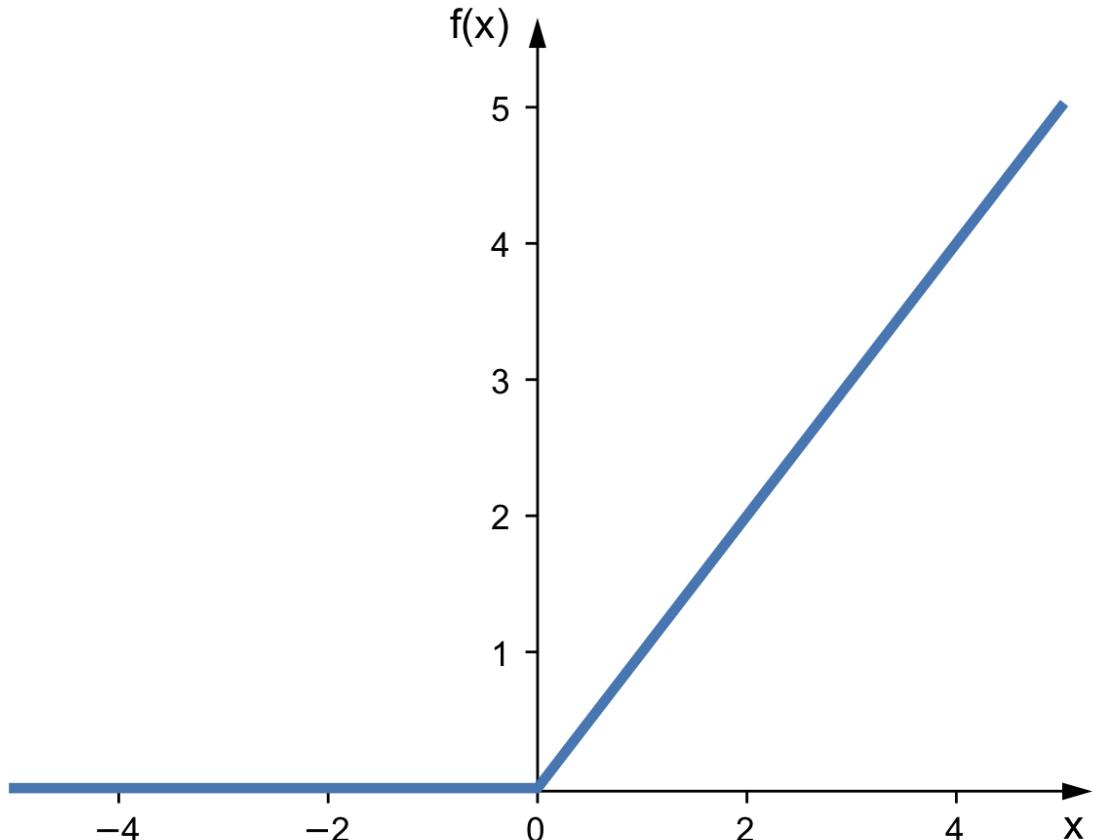
¹⁸Vgl. Giuseppe, *Hands-On Unsupervised Learning with Python*.

¹⁹Vgl. ebd., S. 21.

²⁰Vgl. Johnston und Mathur, *Applied Supervised Learning with Python*, S. 4.

²¹Vgl. *Aktivierungsfunktionen, ihre Arten und Verwendungsmöglichkeiten*.

Welche Aktivierungsfunktion soll also eingesetzt werden? Die beliebtesten und bekanntesten Aktivierungsfunktionen sind *Sigmoid*, *Logistik*, *Tanh (Hyperbolischer Tangens)*, sowie *ReLU (Rectified Linear Unit)*. Ein gemeinsames Problem der ersten drei genannten Funktionen liegt in dem sogenannten **Gradientenverschwinden**, welches bei der ReLU Funktion vermieden und korrigiert wird. Auf das Problem des Gradientenverschwindens wird aufgrund dessen Mathematischer Komplexität nicht weiter eingegangen. In der Grafik C.B ist der Funktionsgraph der ReLU Funktion dargestellt.



Quelle: https://sebastianraschka.com/images/faq/relu-derivative/relu_3.png
 Abbildung C.B: ReLU Funktionsgraph

Zu sehen ist die einfache mathematische Funktion der Form $\text{ReLU}(x) = \max(0, x)$, welche für alle Eingabewerte $x < 0 = 0$ ausgibt, und für $x > 0$ einen einfachen linear steigenden Graphen darstellt. Aufgrund der einfachen und effizienten mathematischen Grundform, sowie der Fähigkeit des Umgehens des Gradientenverschwindens, wurde diese Aktivierungsfunktion immer beliebter und wird neuerdings in fast allen neuronalen Netzwerken verwendet.²²

Aus diesen Gründen wurde sich auch dafür entschieden diese Aktivierungsfunktion für das hier entwickelte Modell zu verwenden.

²²Vgl. *Aktivierungsfunktionen: Neuronale Netze*.

3.5 Gesichtserkennung mit OpenCV

Author — 1329241

In dem nun folgenden Abschnitt wird der theoretische Aspekt der Gesichtserkennung mit dem Tool OpenCV erläutert. In der Arbeit findet die Gesichtserkennung an mehreren Stellen Anwendung, auch wenn es eigentlich um Emotionserkennung geht. So wird zum Beispiel, bevor eine Vorhersage über eine Emotion zu einem Bild gemacht wird, überhaupt erst geprüft, ob auf dem Bild auch ein Gesicht vorhanden ist. Dies ist auch notwendig, da dies die Fehlerbehandlung erleichtert. Sollte ein Bild eingegeben werden, auf dem kein Gesicht zu erkennen ist, darf auch kein Rückgabewert von Emotionsklassifizierungen erhalten werden, weshalb vorher überprüft werden sollte, ob ein menschliches Gesicht auf dem Bild vorhanden ist.

3.5.1 Opencv Klassifizierer

Author — 1329241

OpenCV stellt verschiedenste Klassifizierer für die Aufgabe der Gesichtserkennung zur Verfügung. Zum einen die sogenannten HAAR-Cascade Klassifizierer, zum anderen dedizierte Gesichtserkennungsalgorithmen. Beide Arten werden im Laufe dieser Arbeit verwendet werden, weshalb es zum besseren Verständnis der Arbeit dienlich ist, die Funktionsweise dieser zu beleuchten.

3.5.1.1 HAAR Cascade Klassifizierer

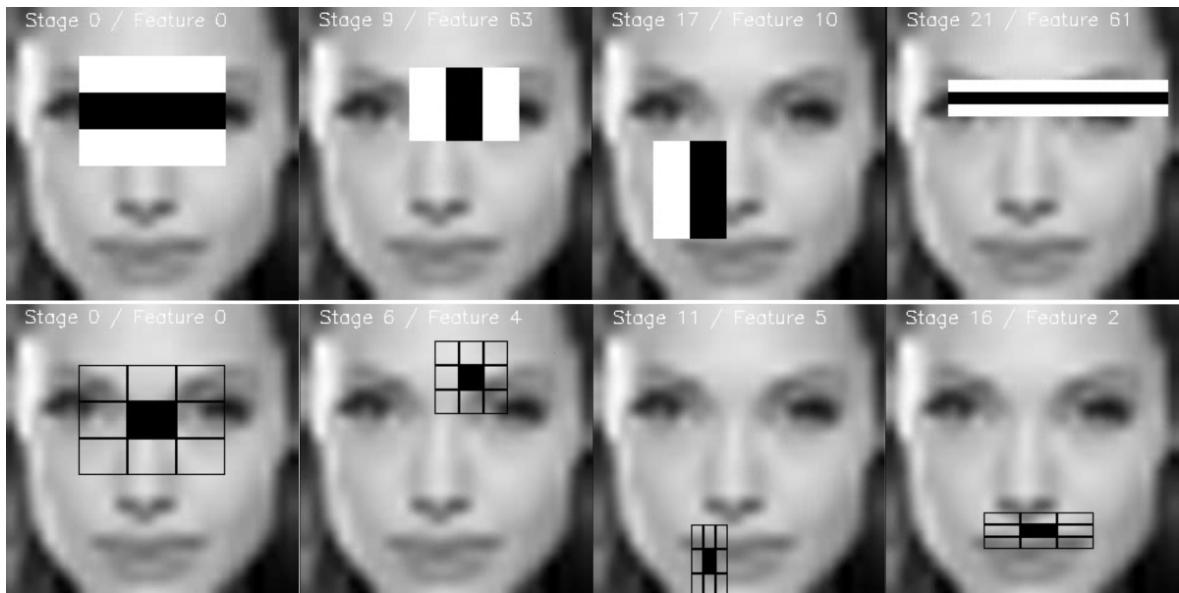
Author — 1329241

Als Datengrundlage für diese Art der Klassifizierer werden positive Bilder gebraucht - im Falle der Gesichtserkennung also Bilder mit Gesichtern - aber auch negative. Aus diesen werden dann Eigenschaften extrahiert und verarbeitet. Grundsätzlich ist diese Art der Klassifizierung deutlich flexibler als "reine" Gesichtserkennungsalgorithmen. Dies liegt daran, dass HAAR Klassifizierungen nicht auf Gesichter beschränkt sind, sondern auf zahlreiche Eigenschaften wie Augen und Münder oder gänzlich auf andere Objekte wie Mäuse und Häuser angewendet werden können. Ein HAAR-Cascade Klassifizierer lässt sich dabei in vier Bereiche einteilen.

- HAAR feature selection

- Creating Integral Images
- Adaboost Training
- Cascading Classifiers

²³ In dem ersten Schritt werden sogenannte HAAR-Features festgelegt. Diese sind im Wesentlichen verschiedenartige Ausschnitte aus den Bildern. Eine beispielhafte Selektion dieser kann der Grafik C.C entnommen werden. Wie in der Grafik ebenfalls zu sehen



Quelle: https://docs.opencv.org/master/dc/d88/tutorial_traincascade.html
 Abbildung C.C: haar features

ist, sind diese Features allesamt rechteckig. Dies bietet die Möglichkeit, die Berechnungen, die gemacht werden müssen, zu beschleunigen, indem Schritt zwei angewandt wird. die vorliegenden Bilder werden dabei in Integralbilder umgewandelt. Diese werden im Folgenden nicht genau expliziert, jedoch werden in Integralbildern, vereinfacht gesagt, in jedem Punkt des Bildes die Pixelsummen eines Rechteckes gehalten, das vom Ursprung bis zu dem jeweiligen Pixel aufgespannt wird. Bei einem Punkt (x,y) wird also das Rechteck $(0,0), (x,0), (0,y), (x,y)$ betrachtet.²⁴ Nun können verschiedene HAAR-Features selektiert werden. Das Problem dabei ist jedoch, dass diese für sich nicht sonderlich aussagekräftig sind. Beispielsweise ein Feature, dass die Nasenflügel betrachtet, ist nicht in der Lage ein Gesicht wirklich zu erkennen. Um deshalb ein aussagekräftiges Ergebnis zu erzielen, werden die besten dieser sogenannten schwachen Klassifizierer zu einem starken Klassifizierer zusammengefasst und trainiert. Dieser

²³Vgl. Berger, DEEP LEARNING HAAR CASCADE EXPLAINED.

²⁴Vgl. Wikipedia - Integralbild.

Schritt beschreibt das Adaboost-Training.²⁵ Diese Trainingsart benutzt nun den letzten Teil, das Kaskadieren von Klassifikatoren. Im Wesentlichen handelt es sich dabei um eine Methode, die so schnell wie möglich falsche Bilder, also solche die nicht der gewünschten Klasse entsprechen, aussortiert. Dabei wird in multiplen Stadien gearbeitet, in denen ein sich verschiebender Rahmen über das Bild gelegt wird, der wiederum die Größe des Zielbildes hat - zum Beispiel eines Gesichtes, Auges, Mundes, etc. . Wenn nun der Rückgabewert für einen Ausschnitt, den der Rahmen gerade umfasst, negativ ist, wird der Rahmen weiter verschoben. Wenn das Ergebnis jedoch positiv ist, wird das Bild dem nächsten Stadium übergeben.²⁶ Auf diese Weise können Gesichter oder andere Objekte mittels HAAR-Klassifizierung sehr schnell erkannt werden.

3.5.1.2 Gesichtserkennungs Algorithmen

Author — 1329241

Der Fisher Face Recognizer ist einer von drei Gesichtserkennungs Algorithmen die OpenCV mit sich bringt. Einer seiner Gegenstücke ist der Eigenface Recognizer, auf dem der Fisher Face Recognizer partiell aufbaut. Aus diesem Grund wird zuerst die grundsätzliche Funktionsweise des Eigenface Algoritihmus beleuchtet. Dieser Algoritihmus bekommt einen Datensatz von verschiedenen Gesichtsbildern als Eingabeparameter. Der Eigenface Algoritihmus stuft dabei manche Gesichtskomponenten wichtiger ein als andere. Zum Beispiel wenn mehr Varianz in den Trainingsdaten gegeben ist im Bereich der Nasen und Augen, und weniger im Bereich der Münder und Ohren, wird der Algorithmus die Nasen und auch Augen als die sinnvolleren oder wichtigeren Komponenten erachten. Anhand dieser wichtigen Komponenten können dann verschiedene Gesichter erkannt werden.²⁷ Das Problem dieses Algorithmus ist nun, dass viele Komponenten der Gesichter nicht mehr beachtet werden, weil sie insgesamt eine zu niedrige Varianz aufgewiesen haben. Ein anderes Szenario wäre, dass eine Komponente ohne Aussagekraft eine sehr große Varianz in den Trainingsdaten enthält. Zum Beispiel, wenn in den Datensätzen eine hohe Varianz der jeweiligen Beleuchtung vorhanden ist, also wenn keine einheitliche Beleuchtung der Bilder vorhanden ist. Die Beleuchtung sagt per se nicht viel über ein Gesicht aus, aber der Eigenface Recognizer würde diese Komponente als sehr wichtig einstufen, was zu Fehlern führen kann.²⁸ Im Gegensatz zum Eigenface Recognizer arbeitet der Fisherface Algorithmus mit klassifizierten Daten. Daher ist er im Vergleich zum bereits explizierten Eigenface-Ansatz ein supervised learning Algorithmus. Dieser supervised Algorithmus verfolgt ebenso

²⁵Vgl. Berger, *DEEP LEARNING HAAR CASCADE EXPLAINED*.

²⁶Vgl. ebd.

²⁷Vgl. *Face recognition using OpenCV and Python*.

²⁸Vgl. *Opencv Dokumentation*.

den Ansatz, dass aus dem Datensatz verschiedene Komponenten extrahiert werden, jedoch wird hierbei keine Gewichtung anhand der Varianz vorgenommen. Dadurch wird das zuvor beschriebene Problem teilweise aufgelöst.²⁹ Teilweise nur deshalb, weil ein Modell, welches mit dem Fisherface Algorithmus trainiert wird und beispielsweise nur Bilder als Eingabewerte mit einer hohen Beleuchtung bekommt, auch nur solche Bilder korrekt bewerten kann. Wenn nun ein schlecht beleuchtetes Bild bewertet werden sollte, wird der Algorithmus an seine Grenzen kommen. Dieses Problem kann unter anderem durch den dritten von OpenCV angebotenen Gesichtserkennungsalgorithmus ausgeglichen werden, dem sogenannten Local Binary Patterns Histograms Recognizer. Ein weiterer Lösungsansatz wäre hingegen noch die Eingabedaten anzupassen, mit denen das resultierenden Modell trainiert wird. Darauf wird im folgenden Unterpunkt 19 eingegangen. Da diese Vorbearbeitung das Problem der Beleuchtung ebenfalls löst und dabei sogar noch weitere positive Aspekte für die Arbeit mit sich bringt, wird an dieser Stelle darauf verzichtet, weiter auf den Local Binary Patterns Histograms Recognizer einzugehen und zum Nachlesen auf die Opencv Dokumentation verwiesen.³⁰

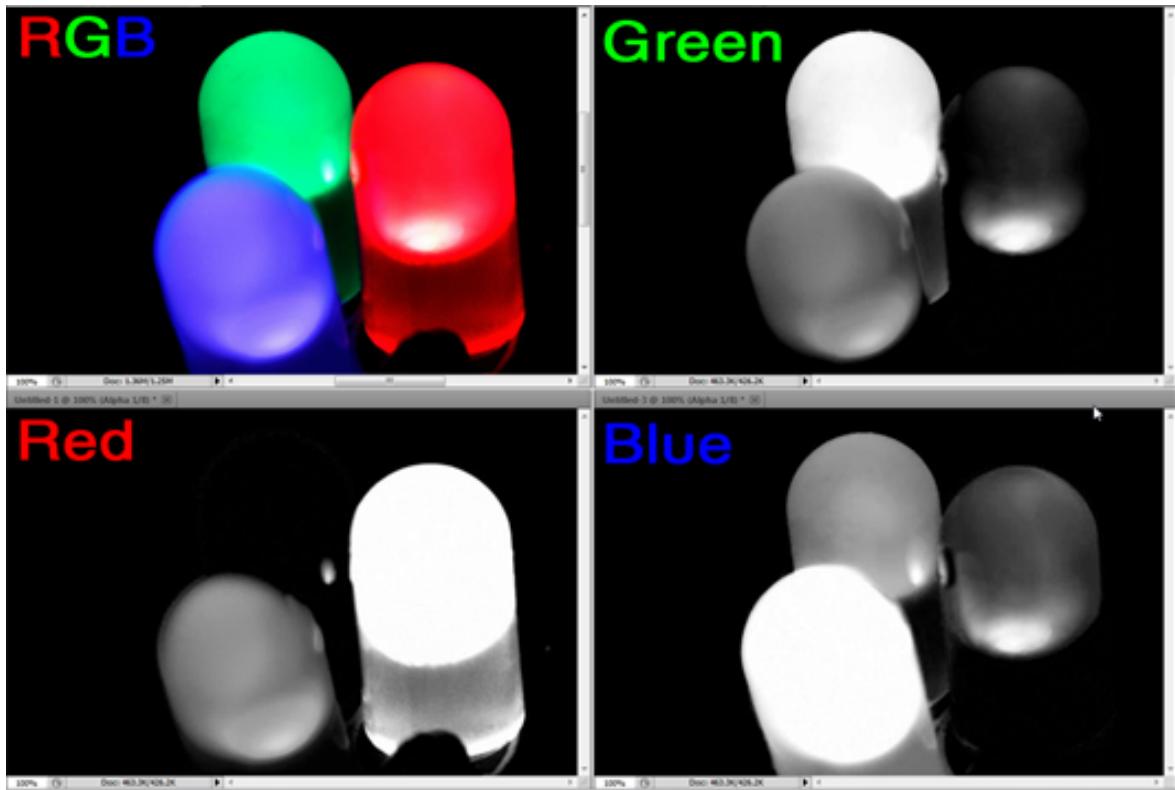
3.5.2 Eingabe Daten

Author — 1329241

Wie im vorherigen Unterpunkt erwähnt, kann das Modell ebenfalls dadurch beeinflusst werden, wie der Eingabedatensatz aufgebaut ist. Dabei wird im Folgenden nicht auf den konkreten Datensatz eingegangen, sondern viel mehr auf die Anforderungen an die Beschaffenheit von diesem. Zunächst ist dabei zu beachten, dass es grundsätzlich einige Möglichkeiten gibt, die Eingabedaten, in diesem Fall Bilder von Gesichtern, vorab zu bearbeiten. Ein wesentlicher Aspekt dabei ist neben dem Anpassen der Bilder auf eine bestimmte Größe auch die Wahl der Farbdarstellung. Dabei geht es weniger darum, ob beispielsweise die Bilder im RGB (Rot - Grün - Blau) oder HSV (Hue - Saturation - Value) Farbraum vorliegen, sondern um die generelle Zahl an Kanälen, die Informationen von einem Bild beinhalten. Beispielsweise werden bei einer klassischen RGB Darstellung Informationen über das Bild auf drei Kanälen gespeichert. Abbildung C.D zeigt wie die einzelnen Kanäle arbeiten. Informationen aus dem Grundbild die Rot erscheinen, werden im roten Kanal besonders hell dargestellt. Ausschnitte aus der originalen Abbildung die blau erscheinen, werden im blauen Kanal ebenso heller dargestellt. Analog zu den beiden Beispielen arbeitet ebenso der grüne Kanal. Werden die drei Kanäle wieder miteinander verbunden entsteht das Ausgangsbild. Der Vorteil solcher mehrkanaligen Formate ist, dass praktisch alle Informationen aus dem original Bild erhalten bleiben.

²⁹Vgl. *Face recognition using OpenCV and Python*, How to fix this issue.

³⁰Vgl. *Opencv Dokumentation*.



Quelle: <https://www.howtogeek.com/wp-content/uploads/2011/02/sshot-74.png>
 Abbildung C.D: RGB Kanäle

Der Nachteil jedoch wird deutlich, wenn die Farbe für den Anwendungsfall keine oder eine geringe Rolle spielt. Da in dieser Arbeit Emotionen anhand von Gesichtern erkannt werden sollen, spielt die Farbe der Bilder selber eine untergeordnete Rolle, da diese keine nützlichen Informationen für das zu trainierende Modell enthält. Es wäre sogar kontraproduktiv. Dadurch dass mehr Kanäle mit Informationen vorhanden sind, muss auch das Modell mehr Daten bekommen, um zu lernen, wie die Informationen aus diesen zu verstehen sind. Dies wird auch von dem sogenannten "Fluch der Dimensionalität" beschrieben. Auf den Anwendungsfall bezogen bedeutet dies nur, dass je mehr Eigenschaften ausgewertet werden sollen, desto mehr Daten zum Lernen vorhanden sein müssen. Um dem präventiv entgegenzuwirken, ohne in Verlegenheit zu geraten, die Testdatensätze später erweitern zu müssen, können auch Bilder verwendet werden, die auf weniger Kanäle an Informationen zurückgreifen. Sogenannte Grayscale- oder Graustufenbilder bieten hier den Vorteil, dass es nicht drei, sondern lediglich einen Farbkanal gibt. In diesem wird ein Wert pro Pixel gespeichert. Je höher dieser Wert, desto weißer der Pixel, je niedriger, desto schwärzer. Dieses Format eignet sich bestens für das hier behandelte Modell, da keine wichtigen Informationen entfallen, die in einer farbigen Version des Bildes vorhanden wären, während sich aber die Anzahl der Daten verringert, die benötigt werden um das Modell erfolgreich trainieren zu können.

Dies erhöht wiederum die Chance das zuvor definierte Should-Kriterium zu erfüllen, welches besagt, dass die Wahrscheinlichkeit zur Erkennung der richtigen Emotion über 50% liegen muss.

Kapitel 4

Stand-Alone Lösung mit Schwerpunkt OpenCV

Author — 2356667

Während der weiteren Ausarbeitung des Konzeptes und der darauf basierenden Umsetzung sind letztendlich zwei verschiedene Lösungen entstanden. Aufgrund von Problemen, die innerhalb des Entwicklungsprozesses im Zusammenhang mit dem Hardwarekonzept aufgetreten sind, konnte das erste Projekt, so wie es geplant war, nicht zu Ende gebracht werden. Da dieser Ansatz jedoch als Grundlage für die letztendlich finale Lösung diente, werden sowohl das Konzept als auch die Umsetzung der ersten Lösung nachfolgend genauer erläutert.

4.1 Konzept

Author — 2356667

Im Folgenden wird das in der Planungsphase entwickelte Konzept zur Emotionserkennung unter verschiedenen Aspekten, wie der Architektur und der Kommunikation, erläutert.

4.1.1 Interaktionskonzept

Author — 2356667

Von außen betrachtet steht zuerst die Planung eines Interaktionskonzeptes an, in welcher Form der Nutzer die Möglichkeit hat, Input zur Verarbeitung zu liefern und an Output zu gelangen. Dem Anwender soll es möglich sein, schnell und verständlich

mit dem Produkt zu interagieren, weshalb eine geeignete Benutzeroberfläche mit zwei wesentlichen Interaktionsoptionen erforderlich ist. Der aktuelle Video-Stream einer internen oder externen Kamera sollte als Input zur Emotionsanalyse verwendet werden und das Ergebnis der analysierten Sequenzen soll ausgegeben werden können. Um dies zu erreichen, soll die Bildverarbeitungsbibliothek OpenCV verwendet werden, mit der sowohl der Zugriff auf alle verfügbaren Kameraeingabegeräte als auch die Darstellung von Bildern und Text in einer grafischen Benutzeroberfläche (GUI) in Form eines Fensters möglich ist. Die vom Nutzer gelieferten Videosequenzen sollen entsprechend bearbeitet werden, um anschließend eine Emotion zu erkennen und den entstandenen Output an die durch OpenCV generierte Oberfläche weiterzugeben. Der Output wird dem Nutzer letztendlich so dargestellt, dass dieser als einfache Textausgabe auf dem Bildschirm erscheint.

Zusammenfassend kann man also sagen, dass der Nutzer aufgrund der Übersichtlichkeit lediglich mit der von OpenCV generierten Oberfläche interagieren kann und sich zu keinem Zeitpunkt mit den dahinterliegenden Komponenten der Emotionserkennung befassen muss.

4.1.2 Architektur

Author — 2356667

4.1.2.1 Programmierumgebung

Author — 2356667

Aufgrund der Vorkenntnisse und Relevanz innerhalb des Studiums soll die Programmiersprache Python in aktuellster Version (3.6.9) vorrangig verwendet werden, jedoch kann unter gegebenen Umständen womöglich auch die Programmiersprache C++ eingesetzt werden, was noch zu prüfen ist. Auf eine bestimmte IDE wie PyCharm, Emacs oder Visual Studio Code wird sich an dieser Stelle nicht festgelegt, da dies jedem Entwickler selbst zu überlassen ist. Diesbezüglich sind nur Einschränkungen aufgrund der gewählten Rechnerarchitektur und der Programmiersprache zu berücksichtigen.

Zur Entwicklung einer Emotionserkennung sollen als grundlegende Komponenten die Programmbibliothek OpenCV zur Bildverarbeitung und das Framework Tensorflow bzw. die Deep-Learning-Bibliothek Keras verwendet werden. Da das Einsatzgebiet von OpenCV in der Bildverarbeitung liegt, soll die Bibliothek dazu genutzt werden, den Input so zu verändern, dass dieser vom Modell zum Trainieren oder Vorhersagen ei-

ner oder mehrerer Emotionen verwendet werden kann. Für den wesentlichen Teil der Arbeit, das Entwickeln eines Modells, welches menschliche Emotionen anhand eines Bildausschnittes von einem Gesicht erkennen kann, ist die Deep-Learning-Bibliothek Keras zu verwenden.

4.1.2.2 Hardware

Author — 2356667

Der Einfachheit halber wird zum Entwickeln als grundlegende Komponente ein handelsüblicher Laptop genutzt. Dabei wird außerdem auf das Open-Source Betriebssystem Ubuntu 18.04.4 LTS in der 64-bit Variante zurückgegriffen. Als zugrundeliegende Ressourcen stehen ein 8 Gigabyte großer Arbeitsspeicher sowie ein Intel Core i5-4210 Quadcore Prozessor mit einer Taktfrequenz von 4 x 2,60 GHz zur Verfügung. Des Weiteren kann die dedizierte Grafikkarte GeForce 820M mit einem Grafikkartenspeicher von 2046 MB verwendet werden. Der Festplattenspeicher von 500 GB kann im Umfang dieser Arbeit vernachlässigt werden, da es im Zusammenhang mit Gesichts- bzw. Emotionserkennung primär darauf ankommt, wie viel Rechenleistung zur Verfügung steht und nicht, wie groß die Speicherkapazität des Laufwerks ist. Um das rechenintensive Trainieren des Modells zur Emotionserkennung in akzeptabler Zeit zu garantieren, soll die Rechenleistung des Prozessors und der Grafikkarte vollständig genutzt werden können.

4.2 Umsetzung

Author — 2356667

In diesem Kapitel wird die Umsetzung der auf dem entwickelten Konzept basierenden Lösung näher erläutert. Es handelt sich dabei um den Stand-Alone Laptop mit installiertem Ubuntu 18.04.4 LTS, wobei der Input und der Output auf OpenCV GUIs basiert.

4.2.1 Input GUI

Author — 2356667

Als Einstieg in das Themengebiet Emotionserkennung bzw. Gesichtserkennung ergibt es Sinn, sich zuerst mit den Grundlagen der Bildverarbeitung und den Grundlagen

von OpenCV vertraut zu machen. Aufgrund der aufgestellten Anforderungen ist es außerdem notwendig, eine grafische Oberfläche zu implementieren, mit der der Nutzer interagieren kann. Wie bereits in dem Konzept beschrieben, soll es dem User daher möglich sein, mithilfe der Webcam und einer grafischen Oberfläche, welche mit OpenCV programmiert wird, ein Bild als Input zu liefern. Wie man in Abbildung D.A sehen kann, wird im ersten Schritt die grafische Oberfläche erstellt und der Video Stream der Webcam in der Oberfläche angezeigt. Der zugehörige Codeausschnitt kann dem Anhang 7.1 entnommen werden. Nachdem man nun auf den Video Stream der Webcam zugreift.



Abbildung D.A: GUI mit Video Stream der Webcam als Output

fen und diesen wiedergeben kann, gilt es den relevanten Bildausschnitt, die sogenannte Region Of Interest (ROI), zu identifizieren. Da wir uns im Umfeld der Gesichts- und Emotionserkennung befinden, sind für uns alle Bereiche relevant, die ein menschliches Gesicht enthalten. Standardmäßig stellt OpenCV einige Modelle zur Objekterkennung zur Verfügung, welche problemlos genutzt werden können. Um so ein vortrainiertes OpenCV-Modell zur Gesichtserkennung einzubinden, kann man folgende Zeile an den Anfang des Codes schreiben:

Listing 4.1: Einbinden eines vortrainierten OpenCV-Modells zur Gesichtserkennung

```
face_cascade = cv.CascadeClassifier('haarcascades/  
haarcascade_frontalface_default.xml')
```

Mithilfe der Methode `detectMultiScale` des eingebundenen Modells, können nun die Koordinaten sowie die Höhe und die Breite von jedem in dem Bild gefundenen Gesicht extrahiert werden. Aufgrund dieser Informationen kann man den bisherigen Code nun so erweitern, dass ein Rechteck um jedes im Video Stream der Webcam gefundene Gesicht gezeichnet wird und über die grafische Oberfläche sichtbar gemacht wird. Das Resultat des entstandenen Codes 7.2 kann der Abbildung D.B entnommen werden. Nun lässt sich noch darüber diskutieren, ob es weitere besonders zu

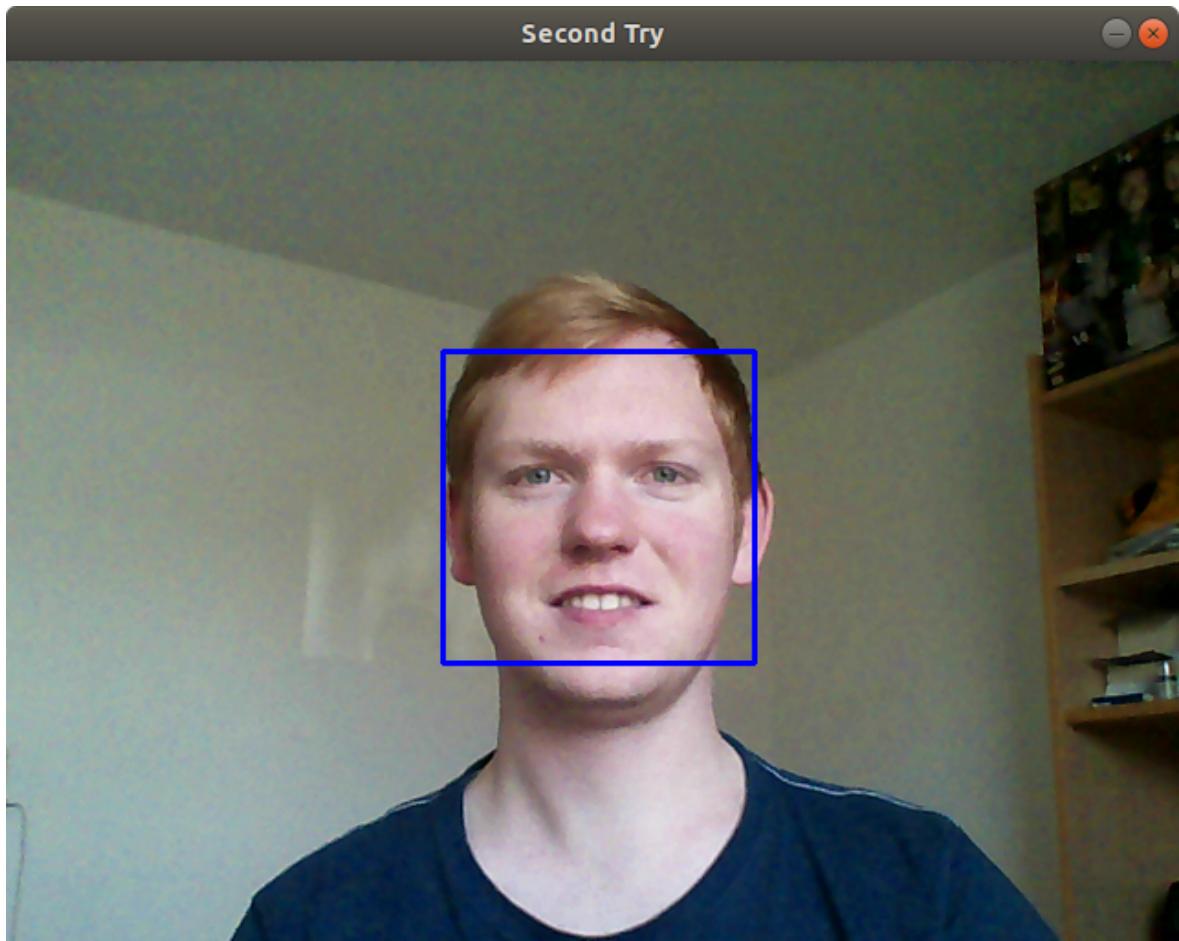


Abbildung D.B: GUI mit Video Stream der Webcam und markierten Gesichtern als Output

berücksichtigende ROIs gibt. Im Zusammenhang mit Emotionen können unter anderem die Augen und der Mund eine besondere Rolle spielen, sodass nachfolgend beispielhaft das Hinzufügen eines Modells zur Erkennung der Augen in jedem bereits entdeckten Gesicht gezeigt wird. Dazu muss zusätzlich zum OpenCV-CascadeClassifier `haarcascade_frontalface_default.xml` der CascadeClassifier `haarcascade_eye.xml` eingebunden werden. Um nun alle Augen in einem Bild zu erkennen, kann wieder die Methode `detectMultiScale` des Modells genutzt werden. Dem nachfolgenden Bild-

ausschnitt D.C kann man nun entnehmen, wie um alle Augen, die sich in dem Bildausschnitt befinden in dem auch ein Gesicht erkannt wurde, ein grünes Rechteck gezeichnet wird. Somit ist man nun in der Lage, alle relevanten Bereiche zu identifizieren und zu

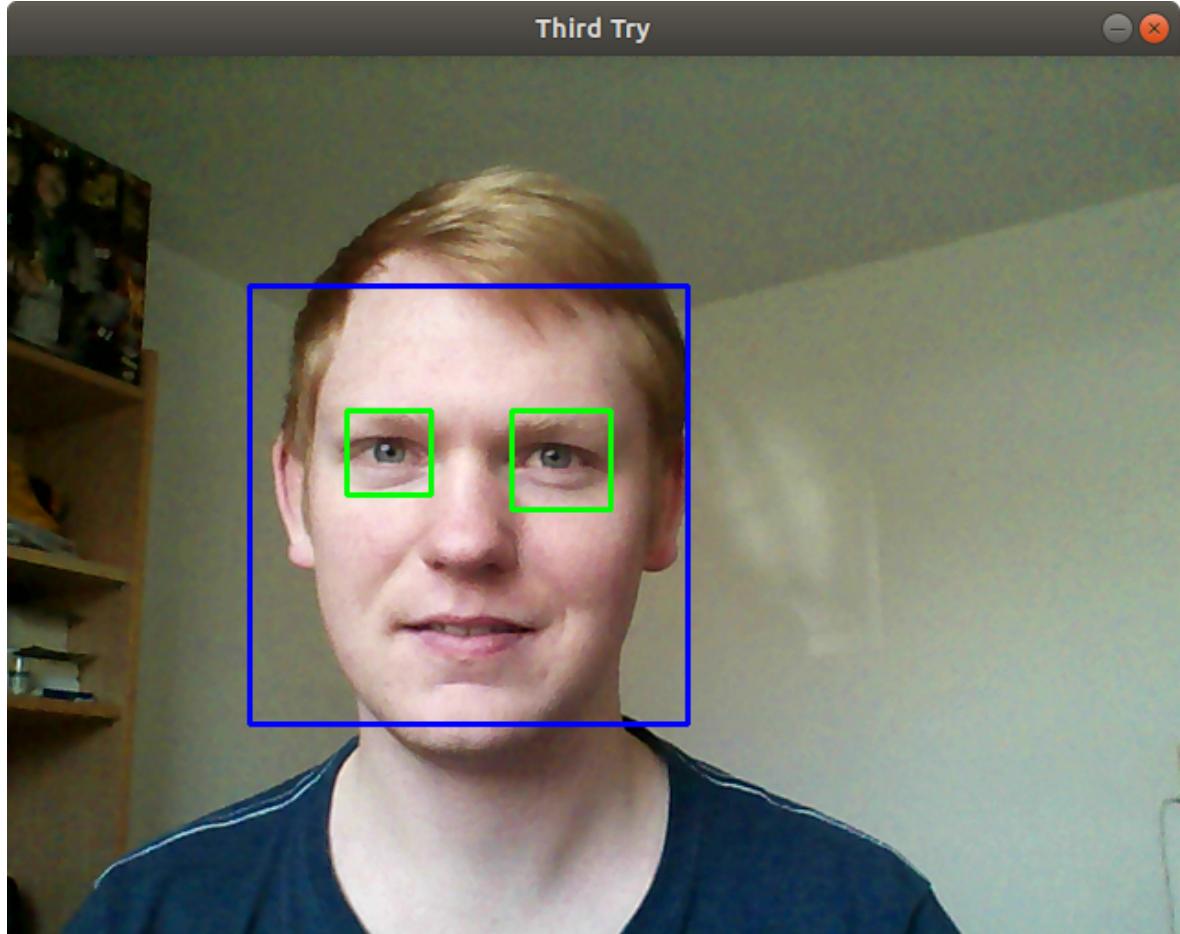


Abbildung D.C: GUI mit Video Stream der Webcam und markierten Gesichtern und Augen als Output

markieren. Um diese relevanten Regionen nun als Input für ein Modell zum Vorhersagen einer Emotion nutzen zu können, müssen die entsprechenden Bereiche jedoch erst einmal abgespeichert werden. Um die Vergleichbarkeit der Bilder untereinander und die Vergleichbarkeit des zu testenden Bildes zu den trainierten Bildern zu gewährleisten, sollten diese vom gleichen Format sein. Wie bereits in der Theorie erwähnt, ist es außerdem sinnvoll entsprechende Ausschnitte nicht im Farbmodus abzuspeichern, sondern lediglich als Grayscale-Grafik. Die Erweiterung des bereits entstandenen Codes um die erläuterten Aspekte kann man am Code 7.3 sehen. Somit ist es dem Nutzer möglich, schnell und unkompliziert Bilder mit der Webcam zu machen und dort enthaltene Gesichter so zu speichern, dass sie als Input zum Vorhersagen einer Emotion genutzt werden können.

4.2.2 Dataset

Author — 2356667

Zum Trainieren eines Modells zur Emotionserkennung wird das Cohn-Kanade Dataset zur Analyse von Emotionen verwendet.¹ In dem verwendeten Dataset sind ca. 10000 Grayscale-Bilder enthalten, welche die in IV.I zu sehenden Emotionen umfassen. Es

Emotion	Anzahl nutzbarer Bilder
Anger	45
Contempt	18
Disgust	59
Fear	25
Happy	69
Sadness	28
Surprise	83

Tabelle IV.I: Emotionen mit jeweiliger Anzahl an Bildern

fällt auf, dass sich die dargestellte Anzahl von ca. 325 nutzbaren Bildern stark von der bereits erwähnten Anzahl von 10000 Bildern unterscheidet. Das liegt daran, dass das Dataset ursprünglich für die Analyse von Emotionsverläufen und nicht direkt für die Emotionserkennung an sich entwickelt wurde. Was das genau heißt, wird durch die Abbildung D.D verdeutlicht.



Abbildung D.D: Emotionsverlauf von Neutral zu Disgust

Jeder Datensatz beinhaltet mehrere Bilder, die den Verlauf einer Emotion von neutral bis hin zur jeweiligen Emotion darstellen. Da dieser Verlauf für die Emotionserkennung

¹Vgl. Cohn, Kanade und Tian, *Comprehensive Database for Facial Expression Analysis*.

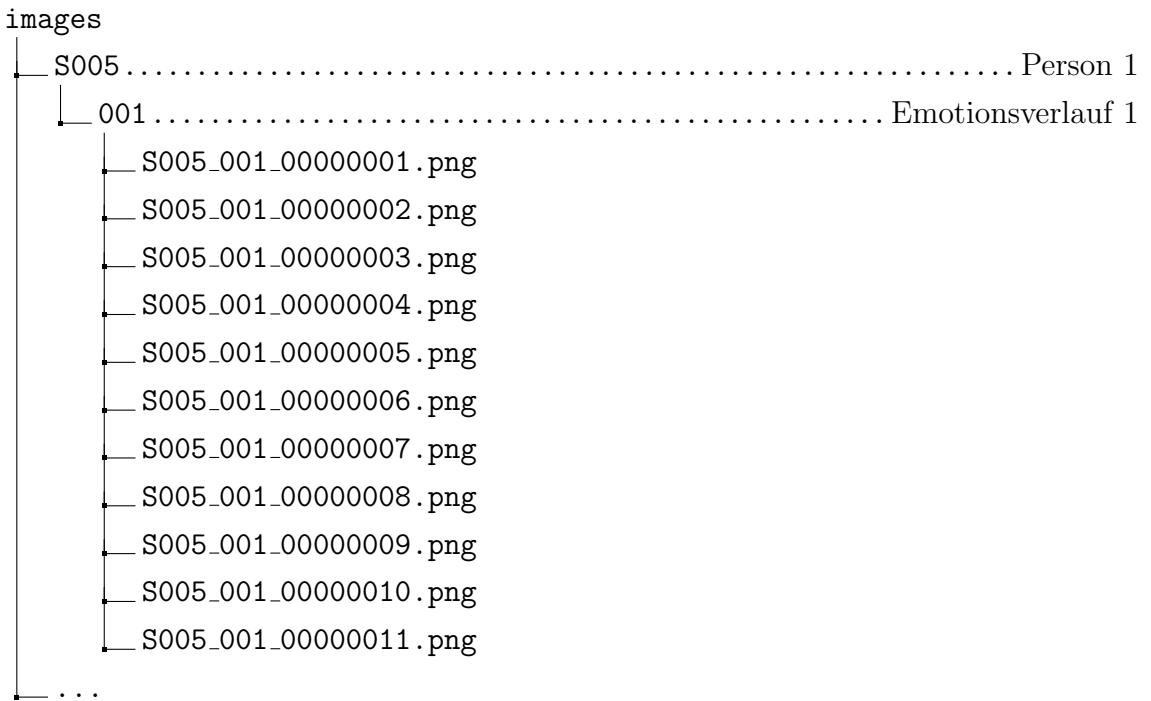


Abbildung D.E: Struktur Emotionsverlauf

allenfalls für die Klassifizierung in Ausprägungen innerhalb einer einzelnen Emotion relevant wäre und somit im Umfang dieser Arbeit nicht betrachtet wird, wird nur das jeweils letzte Bild dieses Verlaufs, also die vollständig ausgeprägte Emotion, zum Trainieren des Modells genutzt. Somit ergibt sich der Unterschied von der Anzahl der jeweils letzten Bilder der Verläufe, ca. 325, und der Anzahl der Gesamtbilder aller Verläufe, ca. 10000. Zur Verzeichnisstruktur des Datasets kann man sagen, dass es grundsätzlich zwei verschiedene Ordner gibt. In einem Ordner befinden sich alle Bilder, also die Verläufe der Emotionen und in dem anderen befinden sich mit der gleichen Struktur die dazugehörigen Emotionen in Textdateien. Die Darstellungen der beiden Strukturen als Verzeichnisbäume können der Anlage G.B entnommen werden. Wie man den Verzeichnisbäumen entnehmen kann, ist die Struktur in beiden Ordner identisch. In der ersten Ebene befinden sich Ordner jeweils mit 'S' aufgeteilt nach Personen. Innerhalb jeder dieser Personenordner gibt es einen oder mehrere weitere Ordner. Diese Ordner auf der 2. Ebene sind dreistellig aufsteigend nummeriert und beinhalten jeweils einen Emotionsverlauf. Im Sachzusammenhang bedeutet dies, dass jede Person eine oder mehrere verschiedene Emotionsverläufe darstellt. Die Syntax der Dateinamen in den aufsteigend nummerierten Ordner, die letztendlich den Emotionsverlauf beinhalten, ist <Personenordner>_<Emotionsverlaufsordner>_<Pos. im Verlauf>.png wobei die Position im Verlauf 8-stellig ist. Nimmt man die in Abbildung D.D gezeigten Bilder als Grundlage, könnten die einzelnen Dateinamen wie im Verzeichnisbaum D.E aussehen. Aus der dargestellten Struktur des Ordners, in dem sich sämtliche Bilder befinden

und den jeweiligen Dateinamen lassen sich keinerlei Rückschlüsse auf die dargestellten Emotionen ziehen. Um herauszufinden, welche Emotion durch welchen Emotionsverlauf dargestellt wird, muss man in das entsprechende Verzeichnis innerhalb des Ordners mit den gelabelten Emotionen schauen. Dort befindet sich dann eine Textdatei, die den selben Dateinamen hat, wie das letzte Bild aus dem dazugehörigen Emotionsverlauf. In dieser Datei befindet sich dann eine Fließkommazahl, die für die jeweilige Emotion in Tabelle IV.I steht. Um nun herauszufinden, welche Emotion durch den in Abbildung D.E gezeigten Emotionsverlauf dargestellt wird, schaut man in das Verzeichnis `emotions/S005/001/`, in dem sich die Datei `S005_001_00000011.txt` befindet. In dieser Datei würde dann `3.000000e+00` stehen, was sich gemäß der Tabelle mit `Disgust` gleichsetzen lässt.

Um später ein vernünftiges supervised Learning auf Grundlage des Datasets durchführen zu können, wird eine eigene Verzeichnisstruktur erstellt, die sortiert nach Emotionen jedes letzten Bild eines Emotionsverlaufs beinhaltet. Zusätzlich wird die Emotion neutral hinzugefügt, da jedes erste Bild eines Emotionsverlaufs als neutrale Ausgangslage dient. Der entwickelte Algorithmus zum Überarbeiten und Sortieren des Datasets und die daraus resultierende neue Verzeichnisstruktur können dem Code 7.4 und der Abbildung G.C im Anhang entnommen werden.

4.2.3 Training

Author — 2356667

Zum Trainieren eines Modells stellt OpenCV verschiedene Klassifizierer zur Verfügung, jedoch wird sich in diesem Abschnitt immer auf das Supervised Learning des FisherFaceRecognizer bezogen, welcher mit dem im vorherigen Schritt präparierten Dataset trainiert wird. Da das Dataset schon sortiert und vorbereitet wurde, ist es nun sehr leicht, eine Methode zu schreiben, die ein Array mit Trainingsdaten und ein Array mit den zugehörigen Labels liefert. Aufgrund der Verzeichnisstruktur des Datasets kann man über alle Emotionsordner iterieren, dabei alle Grayscale-Bilder in das Array mit den Trainingsdaten einlesen und die jeweilige Emotion in das Array mit den Labels einlesen. Dabei ist jedoch zu beachten, dass die Labels bzw. die Emotionen als Integer-Werte beginnend mit 0 und aufsteigend dargestellt werden müssen. Das heißt, die Ordnernamen, welche den Emotionen als Text entsprechen, müssen in Integer-Werte konvertiert werden, indem man z.B. den jeweiligen Index der Emotion in dem globalen Emotions-Array 4.2 nimmt.

Listing 4.2: Emotions-Array

```
emotions = [ "neutral" , "anger" , "contempt" , "disgust" , "fear" , "  
happy" , "sadness" , "surprise" ]
```

Die vollständige Methode zum Extrahieren der Trainigsdaten ist im Anhang unter 7.5 gelistet. Da nun die erforderlichen Arrays mit Trainingsdaten und Labeln erzeugt werden können, beschäftigt man sich als nächstes mit dem zu trainierenden Klassifizierer, dem sogenannten FisherFaceRecognizer. Die `FisherFaceRecognizer`-Klasse stellt die statische Methode `create()` und die Methode `train(InputArray data, InputArray labels)` zur Verfügung welche im Code 7.6 zum Trainieren benutzt werden. Es fällt somit deutlich auf, dass die Komplexität des eigentlichen Trainierens im Vergleich zur Vorbereitung des Datasets wesentlich geringer ist.

4.2.4 Testing

Author — 2356667

Das Testen des im vorherigen Schritt trainierten Modells kann aufgrund der Vorarbeit ebenfalls mit sehr geringem Zeitaufwand implementiert werden. Aufgrund der, wie sich später herausstellte, sehr geringen Datengrundlage und der Should-Anforderung gemäß MoSCoW Priorisierung, dass die Wahrscheinlichkeit zur Erkennung der richtigen Emotion über 50% liegen soll, wird anstatt der üblichen Aufteilung des Datasets in Trainings- und Testdaten das komplette Dataset als Trainingsdaten genutzt. Daraus ergibt sich dann, dass ein anderes Verfahren zum Testen des Klassifiziers gefunden werden muss. Somit wird zum Testen des Modells eine grafische Oberfläche erstellt, die im speziellen als Echtzeitanalyse der Emotionen fungiert. Hierzu kann ein Großteil des Codes genutzt werden, der als Vorarbeit bei der Entwicklung der Input GUI implementiert wurde. Der Ablauf eines konkreten Tests ist dann das Starten der Oberfläche, welche den Video Stream der Webcam ausgibt und gleichzeitig jedes Einzelbild vom Modell analysieren lässt. Dies geschieht, wie ebenfalls bereits beschrieben, durch das Ausschneiden des Gesichtes und das anschließende Konvertieren in das Grayscale-Format sowie daraufhin das Vorhersagen einer Emotion. Das Gesicht wird dann mit einem Rechteck markiert und darüber wird die vorhergesagte Emotion ausgegeben. Verändert man den Code der Input GUI nur geringfügig und erweitert ihn um das Vorhersagen der Emotion und das Schreiben der vorhergesagten Emotion auf den Output Video Stream, dann erhält man den im Anhang 7.7 gezeigten Code.

4.2.5 Optimierung der Lösung

Author — 2356667

Durch das Testen des Modells konnte festgestellt werden, dass trotz der vergrößerten Menge an Daten zum Trainieren die Genauigkeit der Vorhersagen mit ca. 20% deutlich unter den in den Anforderungen definierten 50% liegt. Aufgrund dieser Tatsache wird versucht, den Trainingsprozess zu optimieren. Die Möglichkeiten zum komplexen Konfigurieren eines Modells in OpenCV sind begrenzt, sodass eine Konfiguration, die wesentlich mehr Features der Emotion als zuvor analysiert, erschwert wird. Außerdem würde sich dann die Tatsache bemerkbar machen, dass OpenCV keine Möglichkeit bietet, die Rechenleistungen der CPU und der GPU zu kombinieren. Daraus würde sich ein sehr langwieriger Trainingsprozess ergeben, was nicht mehr den definierten Anforderungen entsprechen würde. Daher wird zunächst versucht, den Prozess in eine extra dafür vorgesehene Bibliothek auszulagern, wobei die Entscheidung auf die Keras-API gefallen ist. Keras selbst dient in unserem Fall als Schnittstelle zum Tensorflow-Framework. Auf die genaue Funktionsweise von Keras wird in einem späteren Kapitel näher eingegangen. An dieser Stelle bleibt nur noch zu bemerken, dass die Installation von Keras und das Einbinden benötigter Komponenten zwar erfolgreich durchgeführt werden konnten, jedoch die Ausführen der benötigten Komponenten aufgrund einer nicht unterstützten Prozessorarchitektur bzw. eines nicht unterstützten Chipsets nicht möglich war. Aufgrund der benötigten Ressourcen und der mit Tensorflow kompatiblen Architektur wurde sich deshalb für eine Server-Client-Architektur entschieden.

Kapitel 5

Server-Client Lösung

Author — 2356667

Basierend auf den Erfahrungen, die während des Entwicklungsprozesses der Stand-Alone Lösung gewonnen werden konnten, werden das Konzept und dementsprechend auch die Umsetzung überarbeitet und optimiert. Nachfolgend wird die damit finale Lösung genauer erläutert.

5.1 Konzept

Author — 2356667

Die Interaktionsmöglichkeiten, die der Nutzer haben soll, bleiben unverändert. Der Übersichtlichkeit halber soll ausschließliche mit einer mithilfe von OpenCV generierten grafischen Oberfläche interagiert werden. Die wesentlichen Änderungen beschränken sich auf die Systemarchitektur, insbesondere auf die Programmierumgebung.

5.1.1 Programmierumgebung

Zusätzlich zu den bereits erwähnten Gründen für die Entscheidung der Programmiersprache Python ist noch die Wiederverwendbarkeit der bisher entwickelten Lösung zu nennen. Um den Anforderungen bezüglich der dem Projekt zur Verfügung stehenden Zeit gerecht zu werden, wird weiterhin die Programmiersprache Python verwendet, da so der bisher implementierte Code als Grundlage dienen kann und die neue Lösung darauf aufbauend entwickelt werden kann. Außerdem stellt sich nun die Frage, mit welcher Entwicklungsumgebung gearbeitet wird. Da nun auf einer Cloud-Infrastruktur entwickelt wird, sind die reinen grafischen Umgebungen wie Visual Studio Code oder PyCharm nicht zu empfehlen. Aufgrund vieler Vorteile, die dem entsprechenden Kapitel zu entnehmen sind und der ebenfalls bereits erwähnten Relevanz innerhalb des

Studiums, wird zum weiteren Entwickeln die Open-Source Webapplikation Jupyter Notebook verwendet. Da über das Webinterface insbesondere das Debugging vereinfacht wird, kann vor allem der Entwicklungsprozess signifikant beschleunigt werden. Der Einheitlichkeit halber wird die Clientanwendung ebenfalls mithilfe von Jupyter Notebook entwickelt, wobei diese in der finalen Version durch minimale Anpassungen auch problemlos ohne Jupyter Notebook ausführbar gemacht werden kann. OpenCV, die Programmabibliothek für Bildverarbeitung und Keras, die Deep-Learning-Bibliothek, werden weiterhin für die bereits beschriebenen Zwecke verwendet.

5.1.2 Hardware

Author — 2356667

Der Server wird in Cloudumgebung gehostet, die Infrastructure as a Service (IaaS) für Wissenschaft und Bildung bereitstellt. Hier ist besonders hervorzuheben, dass durch die Cloudumgebung eine hohe Flexibilität bezüglich der Skalierbarkeit gewährleistet ist. Hinsichtlich der verfügbaren Ressourcen wird ein Paket gewählt, das 4 virtuelle CPUs sowie 16 Gigabyte Hauptspeicher beinhaltet. Sollte im Verlauf der Umsetzung auffallen, dass die standardmäßig zur Verfügung gestellten Ressourcen nicht ausreichen, können diese bis zu 16 vCPUs und bis zu 32 Gigabyte RAM erweitert werden. Die vordefinierte Größe der Festplatte von 12 GB sollte für den Umfang des Projektes ausreichen, jedoch kann auch hier die Kapazität um bis zu 50 Gibibytes erhöht werden. Weiterhin wird auf das Betriebssystem Ubuntu 18.04 gesetzt, da so ein Großteil der zuvor gewonnenen Erfahrung genutzt werden kann.

5.2 Umsetzung

Author — 2356667

Trotz der aufgetretenen Komplikationen bei der Entwicklung einer Stand-Alone Lösung muss an dieser Stelle erwähnt werden, dass große Teile, gerade im Bereich des Inputs und Outputs wiederverwendet werden. Die auf Grundlage der Stand-Alone Version entwickelte Server-Client Lösung wird nachfolgend genauer erläutert.

5.2.1 Dataset

Author — 2356667

Im Zuge der vertieften Recherche zu Keras und der direkten Verwendung im Bereich Gesichts- und Emotionserkennung konnte ein besseres Dataset als zuvor gefunden werden. Das Dataset gehört zu einer Open-Source Facial Expression Recognition auf der Plattform Kaggle.¹ Kaggle ist eine Online-Community deren Hauptzweck es ist, Data-Science-Wettbewerbe auszuschreiben, an denen jeder interessierte teilnehmen kann. Außerdem wird durch die Plattform ermöglicht, Datensätze zu finden und zu veröffentlichen und ggf. Modelle auszuprobieren oder ebenfalls zu veröffentlichen. Vereinfacht kann man sagen, dass Kaggle ein virtueller Treffpunkt für Data-Scientists in den Bereichen maschinelles Lernen und KI-Entwicklung ist.

Nun wird das im Projekt verwendete Dataset betrachtet. Das Dataset umfasst 35887 Datensätze, die allesamt gelabelt sind. Dabei wird das Set in Trainingsdaten, Public-Testdaten und Private-Testdaten unterteilt, wobei 80% aller Werte auf das Trainingsset entfallen und jeweils 10% auf die beiden Testsets. Die Unterteilung der Testdaten in Public und Private ist dabei nur im direkten Zusammenhang mit dem Kaggle-Wettbewerb von Relevanz, da ein Set zum Testen des Modells für die Entwickler selbst vorgesehen ist und das Andere, um einen Sieger der Challenge auszumachen. Da wir mit unserem Modell nicht an der Facial Emotion Recognition Challenge teilnehmen, kann man die beiden Testsets zu einem großen zusammenfassen. Damit erhält man 28709 Datensätze zum Trainieren des Modells und 7178 Datensätze zum Testen. Aufgrund der Menge an Daten werden diese nicht wie beim anderen Dataset in einer Ordnerstruktur mit entsprechend vielen Bilddateien zur Verfügung gestellt, sondern als Comma-Separated-Values in einer CSV-Datei. Dabei handelt es sich um eine Art Tabelle, deren Struktur der nachfolgenden Tabelle V.I zu entnehmen ist. Die Spalte `emotion` kann dabei Werte von 0 bis 6 annehmen, wobei sich insgesamt 7 verschiedene Werte ergeben, die den Emotionen entsprechend der Tabelle V.II zugeordnet werden können. Außerdem werden in der Tabelle die absoluten und relativen Häufigkeiten der Emotionswerte dargestellt, da diese für spätere Optimierungen relevant sind. Die Grafik E.A stellt eine visualisierte Häufigkeitsverteilung der angegebenen absoluten Häufigkeiten dar.

¹Vgl. Goodfellow u. a., *Challenges in Representation Learning: A report on three machine learning contests*.

emotion	pixels	Usage
0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121 119 115 110 98 ...	Training
0	151 150 147 155 148 133 111 140 170 174 182 154 153 164 173 178 ...	Training
2	231 212 156 164 174 138 161 173 182 200 106 38 39 74 138 161 ...	Training
4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 19 43 52 13 26 40 ...	Training
6	4 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84 115 127 137 142 ...	Training
2	55 55 55 55 55 54 60 68 54 85 151 163 170 179 181 185 188 188 ...	Training
4	20 17 19 21 25 38 42 42 46 54 56 62 63 66 82 108 118 130 139 ...	Training
3	77 78 79 79 78 75 60 55 47 48 58 73 77 79 57 50 37 44 56 70 80 ...	Training
0	254 254 254 254 254 249 255 160 2 58 53 70 77 76 75 78 68 18 32 ...	PublicTest
0	170 118 101 88 88 75 78 82 66 74 68 59 63 64 65 90 89 73 80 80 ...	PrivateTest

Tabelle V.I: Struktur des CSV-Datasets der FER-Challenge

Wert	Emotion	H_n	h_n
0	Angry	4953	13,80%
1	Disgust	547	1,52%
2	Fear	5121	14,27%
3	Happy	8989	25,05%
4	Sad	6077	16,93%
5	Surprise	4002	11,15%
6	Neutral	6198	17,27%

Tabelle V.II: Zuordnung und Häufigkeiten der Emotionen

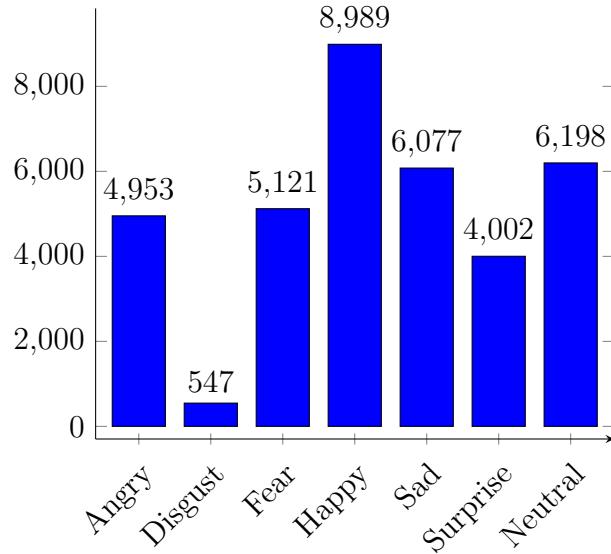


Abbildung E.A: Häufigkeitsverteilung der Emotionen

Nun werden die anderen Spalten betrachtet. In der Spalte `pixels` befinden sich die eigentlichen Bilddaten als Pixelwerte, die lediglich mit einem Leerzeichen getrennt in einem String dargestellt werden. In Tabelle V.I wird nur ein kleiner Ausschnitt der insgesamt 2304 Pixelwerte pro Datensatz angezeigt. Aus der Darstellung der Bilder im Grayscale-Format ergeben sich die in der Theorie bereits erläuterten Pixelwerte von 0

bis 255. Die Anzahl von 2304 Pixeln ergibt sich durch die Länge und die Breite der Bilder von jeweils 48 Pixel mal 48 Pixel. Bevor man die Bilddaten in Form des Strings aus Pixeln für das Modell nutzen kann, müssen diese erst entsprechend formatiert und umgeformt werden, aber dazu später mehr. Die letzte der Tabelle zu entnehmende Information befindet sich in der Spalte `Usage`. Dort wird jedem Datensatz zugeordnet, ob er im Zuge des Wettbewerbes zum Trainieren, zum privaten Testen oder zum öffentlichen Testen verwendet wird.

5.2.2 Modell

Author — 1329241

Nachfolgend wird das für die Emotionserkennung erstellte Modell mit den jeweils einzelnen Layern näher erläutert. Die komplette Struktur des Modells kann der Abbildung G.D im Anhang entnommen werden. Wie zu sehen ist, ist das Modell ein einfaches konvolutionales neuronales Netzwerk. Wie der Grafik zu entnehmen ist sind die Eingabedaten 48 x 48 Pixel große Bilder, die wiederum durch drei sequentielle Schichten von Convolutional Layern und Pooling Layern verarbeitet werden. Jede dieser drei Schichten weist dabei wiederum drei Conv2D-Layer auf. Diese sind in Keras Convolutional-Layer. Jeder von diesen hat dabei 32 Filter als Verarbeitungsparameter, die wiederum eine Größe von 3 x 3 Pixeln aufweisen. Die Aktivierungsfunktion jedes dieser Conv2D-Layer ist die Relu Funktion, die bereits in Kapitel 3.4.3 erläutert wurde. Die Padding-Strategie für diese Schichten ist "same". Das bedeutet, dass die Bilder über die die Filter der Conv2D-Layer laufen am Rand so aufgefüllt werden, dass die resultierenden Ausgabedaten ebenfalls 48 x 48 Pixel groß sind, und sich nicht nach jedem Convolutional-Layer verkleinern. Auf die jeweils drei Conv2D-Layer folgt ein Pooling Layer. Dieser verwendet die Max. Pooling Strategie und hat eine Größe von 2 x 2 Pixeln. Dadurch halbieren sich die Größen der Eingabebilder jeweils von 48 x 48 Pixeln auf 24 x 24 bzw. 12 x 12 und 6 x 6. Der anschließende Flatten-Layer formatiert die 6 x 6 Bilder und die jeweiligen 32 Filter um, sodass die darauffolgenden Dense-Layer (Vollständig verbundene Layer) die Bilder verarbeiten können. Die drei Dense-Layer Verdichten die Ausgabeneuronen so, dass am Ende für jede Emotion die vorhergesagt werden kann ein Neuron als Ausgabeknoten übrig ist. Aktivierungsfunktionen der ersten beiden der drei Dense-Layer ist die Relu Funktion. Die Letzte Schicht hingegen verwendet für die Ausgabe die Softmax Funktion, diese ermöglicht es die Wahrscheinlichkeiten für die jeweilige Emotion auszugeben.

5.2.3 Trainieren des Modells

Author — 2356667

Sobald das beschriebene Modell kreiert wurde, kann mit dem wesentlichen Teil, dem Trainieren, fortgefahren werden. Ein definiertes Modell besitzt in Keras die Methode `fit`, deren Verwendung im einfachsten Fall folgendermaßen aussieht: `fit(x=Trainingsdaten, y=Trainingslabels)`, wobei `x` und `y` in diesem Fall Numpy-Arrays sind. Die Trainingsdaten und die entsprechenden Labels erhält man aus dem Dataset, jedoch müssen diese noch in das erforderliche Format, eine Matrix bzw. ein Numpy-Array konvertiert werden. Dazu müssen die bereits erwähnten Pixel in der Zeichenkette in eine Matrix der Form 48 Pixel x 48 Pixel gebracht werden, wobei wiederum jedes Pixel als eindimensionales Array mit einem Element dargestellt wird. Sollten an dieser Stelle Farbbilder mit dem Format 48 Pixel x 48 Pixel als Input genutzt werden, hätte man anstatt der Struktur (48, 48, 1) entsprechend der drei RGB-Werte Daten der Struktur (48, 48, 3). Optional können der Methode `fit` Validierungsdaten und die zugehörigen Labels als Tupel übergeben werden. Dabei ist zu beachten, dass beide Komponenten als Numpy-Array vorliegen müssen und es analog zu den Trainingsdaten einer Konvertierung in das passende Format bedarf. Der Methode `fit` zum Trainieren des Modells können außerdem einige optionale Parameter mitgegeben werden, die im Zusammenhang mit den Validierungsdaten besonders für die Optimierung des Modells, dem Hyperparameter-Tuning, wichtig sind.

Als Rückgabewert liefert die angesprochene Methode ein History-Objekt, in welchem Daten wie Loss und Accuracy der Trainingsdaten und ggf. Loss und Accuracy der Validierungsdaten während des Trainingsprozesses aufgezeichnet wurden. Das History-Objekt existiert jedoch nur innerhalb der Laufzeit des Python-Skripts und wird nicht zusammen mit dem Modell über die API-Methode `save(path=modelToSave.model")` abgespeichert. Um die History eines Modells jederzeit anzeigen, ausgeben und ggf. visualisieren zu können, werden die aufgezeichneten Daten zusätzlich zum Modell im JSON-Format abgespeichert und können bei Bedarf über entsprechende Bibliotheken wieder geladen und in ein Dictionary konvertiert werden.

Die Visualisierungen der aufgezeichneten Daten inklusive entsprechender Testdaten mithilfe der Bibliothek `matplotlib` können der Abbildung E.B im Anhang entnommen werden.

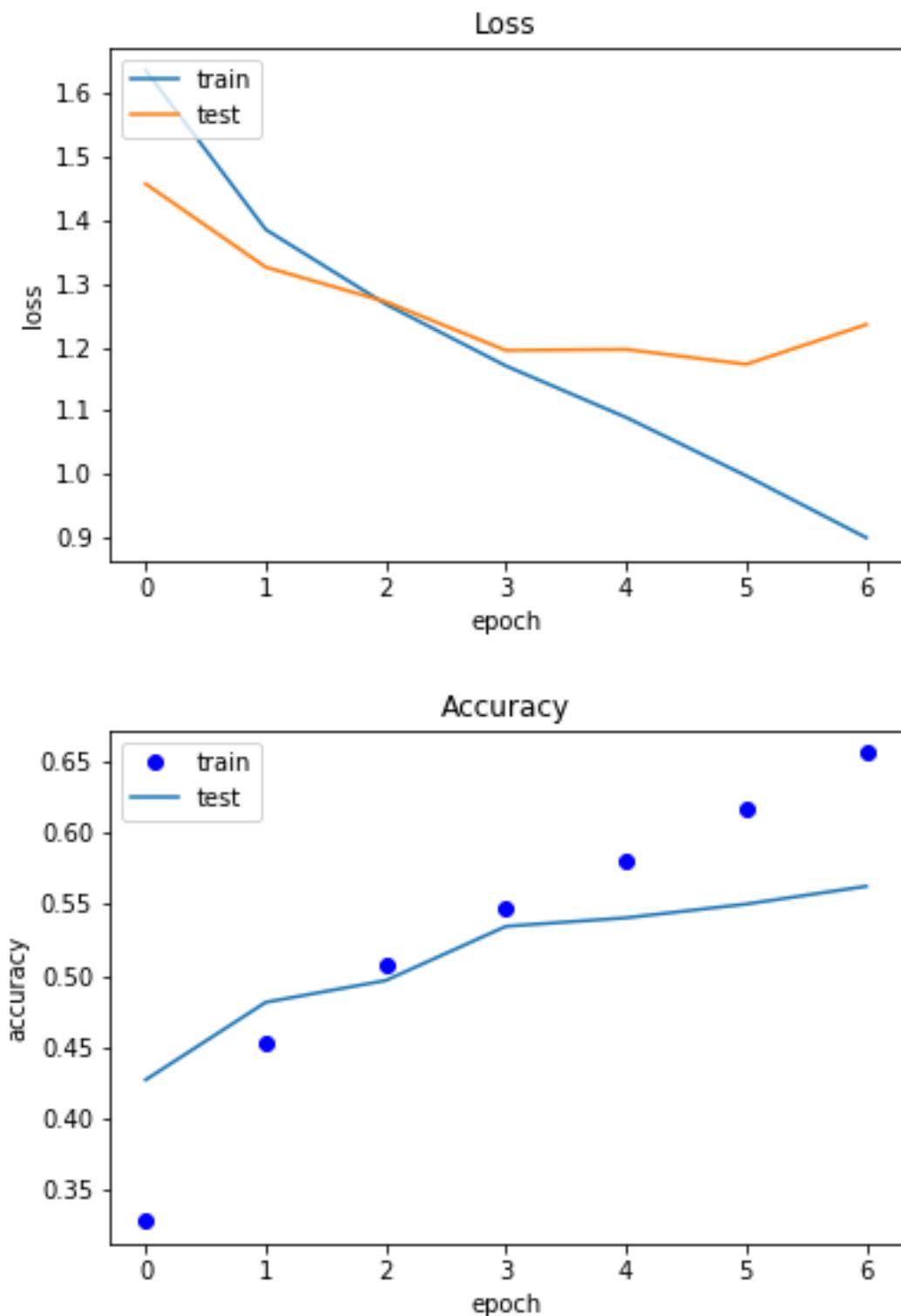


Abbildung E.B: Visualisierung der aufgezeichneten Daten des History-Objektes

5.2.4 Testen des Modells

Author — 2356667

Beim Testen des Modells benötigt man analog zum Trainieren eines Modells mehrere Datensätze und deren jeweilige Label. Um eine Genauigkeit zu ermitteln, lässt man sich die Label des Testsets vom trainierten Modell vorhersagen und gleicht diese mit den tatsächlichen Labeln ab. Voraussetzung dafür ist, dass die Testdaten dem selben Format wie die Trainingsdaten, also einer $(48, 48, 1)$ -Matrix, entsprechen. Mithilfe der Keras API-Methode `predict`, der man die gesamten Trainingsdaten auf einmal übergeben kann, kann man sich auf Grundlage des Modells die Labels vorhersagen lassen. Die konkrete Rückgabe der Methode ist ein Array, dessen Anzahl der Elemente der Anzahl an Trainingsdaten entspricht. Dabei ist wiederum jedes Element selbst ein Array, dessen Länge der Anzahl an möglichen Emotionen entspricht. Übergibt man beispielsweise 10 Testdatensätze und das Modell wurde mit 4 Emotionen trainiert, erhält man ein zweidimensionales Array, welches 10 Arrays der Länge 4 enthält. In den inneren Feldern sind für jede Emotion Werte zwischen 0 und 1 enthalten. Da diese einer Art Wahrscheinlichkeit entsprechen, ergeben sie kumuliert den Wert 1. Vereinfacht gesagt wird für jede Emotion, die als Input gegeben wird, die prozentuale Übereinstimmung mit jeder möglichen Emotion zurückgegeben bzw. die Wahrscheinlichkeit zu jeder möglichen Emotion vorhergesagt. Dies könnte beispielhaft so aussehen: `[0.05690899, 0.09713534, 0.5027235, 0.04000191, 0.29613075, 0.00709949]`. Möchte man zu einem Input eine einzige Emotion vorhersagen, ist es sinnvoll, die dem Maximum des Arrays entsprechende Emotion zurückzugeben. Nimmt man die in Tabelle V.II definierte Zuordnung von Emotionen als Grundlage, entspricht das Maximum des Arrays dem Index 2, wobei dieser Wert wiederum der Emotion **Fear** zugeordnet ist.

Gleicht man die vorhergesagten Emotionen mit den tatsächlichen Emotionen ab, kann man eine statistische Auswertung über der Gesamtmenge aller Daten vornehmen, z.B. dass die Wahrscheinlichkeit einer richtigen Vorhersage bei 40% liegt. Außerdem ist es möglich, den Anteil der richtigen Zuordnung jeder einzelnen Emotion, z.B. `[0.1, 0.2, 0.05, 0.15, 0.91, 0.34]` zu ermitteln. Dem letzteren Beispiel kann man entnehmen, dass das Modell sehr gut geeignet ist, die Emotion, die dem Wert 4 entspricht, zu erkennen, jedoch sehr schlecht darin ist, alle anderen Emotionen zu erkennen. Beide Daten werden während des Testprozesses berechnet und sind bei der späteren Optimierung des Modells hilfreich.

5.2.5 Optimierung des Modells

Author — 1329241

Um das Modell nun zu optimieren wurden einige Schritte unternommen. Zum einem wurde die Layer-Anordnung und auch Anzahl im Laufe der Arbeit verändert. Dabei konnten mit der vorliegenden Anordnung die aus der Grafik G.D zu entnehmen ist, die besten Ergebnisse erzielt werden. Die Konfiguration der einzelnen Layer wurde ebenfalls überarbeitet, vor allem hinsichtlich der Aktivierungsfunktion, Optimierungsfunktion, der Verlustfunktion, aber auch der allgemeinen Verarbeitungsparameter wie Padding-Strategie oder Filtergröße und Anzahl. Es wurden ebenso Optimierungen an der Epochen-Anzahl und Batch size vorgenommen. Letztlich wurden insgesamt 7 Epochen - Also sieben ganze Durchläufe zum Trainieren des Modells - vorgenommen, die jeweils eine Batch size von 64 aufweisen. Da zum Trainieren 80% des Gesamtdatensatzes verwendet wurden, also insgesamt 28709 Bilder, werden pro Epoche insgesamt 448 Durchläufe mit 64 Bildern, und ein Durchlauf mit 37 Bildern durchgeführt. Diese Anzahl kommt dadurch zu Stande, dass bei mehr Epochen ein Overfitting an den Trainingsdaten zu erkennen war. Die Batch size hingegen wurde so gewählt, um das Modell schnell und gleichzeitig weniger Ressourcen-intensiv zu gestalten. Die Optimierung des Modells generell folgt ein wenig dem trial and error Prinzip. Es gibt zwar einige Empfehlungen² - vor allem hinsichtlich der Hyperparameter wie Epochengröße und Batch size - jedoch gibt es keine generell beste Lösung die auf alle Anwendungsfälle passt. Deshalb müssen die besten Einstellungen für Hyperparameter zu dem jeweiligen Datensatz und Modellaufbau "gefunden" werden

5.2.6 Webserver

Author — 2356667

Einer der wesentlichen Unterschiede zu der zuvor entwickelten Lösung ist die Architektur. Es läuft nicht mehr alles lokal auf einem Rechner, sondern es wird zwischen Server und Client unterschieden. Alles was das Modell direkt betrifft, also das Trainieren, Testen und Vorhersagen, geschieht auf dem Server, wo auch das Modell selbst existiert. Um mit dem Client interagieren zu können, wird das Webframework Flask verwendet, welches konkret als Webserver dient. Da Flask in Python geschrieben ist, wird die Implementierung von Endpunkten in Verbindung mit den bereits implementierten Funktionalitäten sehr stark vereinfacht.

²Vgl. *A guide to an efficient way to build neural network architectures.*

Die Interaktion zwischen Server und Client ist sehr überschaubar und es wird lediglich ein einziger Endpunkt auf der Serverseite benötigt. Dieser Endpunkt nimmt mittels der POST-Methode ein vom Client geliefertes, grob vorbearbeitetes Bild entgegen. Dieses Bild wird dann auf dem Server so formatiert, dass mithilfe eines bereits trainierten Modells eine Emotionsvorhersage getroffen werden kann. Das Ergebnis wird dann an den Client zurückgeliefert und kann dort in einen entsprechenden Kontext, wie die reine Ausgabe oder die Analyse eines Pokerfaces, gesetzt und weiterverwendet werden.

Damit der Server mithilfe eines Modells Vohersagen machen und Bilder archivieren kann, werden zur Initialisierung das Modell geladen und entsprechende Ordnerstrukturen angelegt, sofern diese noch nicht vorhanden sind. Danach kann der Webserver gestartet werden, um Verbindungen vom Client über einen vordefinierten Port entgegenzunehmen. Sobald ein Client ein Bild an den Server übergibt, wird dieses gespeichert bzw. archiviert. Da nicht sichergestellt ist, dass der Client ausschließlich Bilder der Größe 48 Pixel x 48 Pixel mit einem Gesichtsausschnitt sendet, werden diese zunächst serverseitig geprüft und ggf. überarbeitet werden. Die allgemeine Vorgehensweise ist dabei, zuerst ein Gesicht zu erkennen und dieses danach auf 48 Pixel x 48 Pixel zuzuschneiden. Damit das Modell eine optimale Vorhersage geben kann, wird der Ausschnitt des extrahierten Gesichtes ggf. noch in ein Graustufenformat konvertiert. Um nun einen validen Input für das Modell zu generieren, wird das Image-Objekt in einen Pixel-String bzw. in eine Matrixrepräsentation der Form (48, 48, 1) gebracht. Jetzt kann auf dem bei der Initialisierung geladenen Modell die bereits erläuterte Methode `predict` aufgerufen werden. Der Rückgabewert dieser Funktion bzw. die Vorhersagewerte der einzelnen Emotionen werden dann an den Client zurückgegeben.

5.2.7 Client

Author — 2356667

Der Hauptzweck der Client-Anwendung ist die Darstellung des Output mithilfe einer grafischen Benutzerschnittstelle. Im Hintergrund interagiert der Client mit dem Server, indem er Webcam-Bilder liefert, die hinsichtlich der Emotion auf dem Server analysiert werden und erhält Vorhersage-Matrizen. Die Interpretation dieser Werte obliegt dem Client, der diese dem Nutzer auf bestimmte Weise zugänglich macht. Die Anwendung umfasst somit zwei wesentliche Funktionalitäten, eine grafische Oberfläche zur Interaktion mit dem Anwender und im Hintergrund die Netzwerkkommunikation mit dem Server. Da nach den definierten Anforderungen keine komplexen Interaktionen mit dem Nutzer notwendig sind, umfasst die GUI lediglich ein Fenster in dem der Video-Stream der Webcam ausgegeben wird. Zusätzlich enthält das Fenster, als Schriftzug in einer Ecke des Bildes, die Information, ob es sich laut Definition um ein Pokerface handelt

oder nicht. Somit kann der Nutzer seine aktuelle Emotion sehen und die Ausgabe, ob ein Pokerface vorhanden ist oder nicht, besser nachvollziehen. Da die Ausgabe des Video-Streams im Fenster und die Netzwerkkommunikation grundsätzlich voneinander unabhängig sind, können diese Aufgaben in zwei verschiedene Prozesse bzw. Threads aufgeteilt werden. Wird dies nicht gemacht, kann es je nach Auslastung des Netzwerkes oder Ressourcenknappheit auf der Serverseite zu einer ruckeligen und verzögerten Ausgabe des Video-Stream kommen. Somit werden im Hauptprozess die grafischen Aufgaben, wie die Ausgabe des Video Streams und des Schriftzuges, erledigt und in einem zweiten Thread wird die Kalkulation des Pokerfaces und der Informationsaus tausch mit dem Server geregelt. Um Daten zwischen den beiden Threads auszutauschen werden globale Variablen genutzt. Die Information, ob es sich um ein Pokerface handelt oder nicht, wird z.B. vom sekundären Thread in eine globale Variable geschrieben und vom GUI-Thread ausgelesen. In definierten Intervallen kann der Thread, der unter anderem mit dem Server kommuniziert, einzelne Bilder des Video-Streams abgreifen und zur Vorhersage an den Server schicken, ohne dass bei der grafischen Ausgabe auf die Antwort des Servers gewartet werden muss. Wie bereits erwähnt, werden die vom Server gelieferten Daten clientseitig interpretiert, was konkret bedeutet, dass ein Algorithmus implementiert werden muss, der anhand der Vorhersagedaten entscheidet, ob ein Pokerface vorhanden ist oder nicht. Um diese Auswertung eines Pokerfaces umzusetzen wurden zwei verschiedene Algorithmen entwickelt, welche im Folgenden genauer expliziert werden.

5.2.7.1 Langzeit-orientierter Algorithmus

Author — 8774695

Die erste Umsetzung der clientseitigen Auswertung betrachtete das Szenario, einen Pokerspieler während der gesamten Dauer eines Poker-Spieles zu analysieren und in Echtzeit dem Anwender mitzuteilen, wenn der Gesichtsausdruck des Spielers von seinem üblichen Pokerface abweicht. In der Theorie sollte demnach ein leichtes Lächeln oder ein trauriger Gesichtsausdruck direkt erkannt werden und somit dem Anwender ein Feedback über das derzeitige Blatt des betrachteten Spielers geben. Die Grundidee bei der Umsetzung des Algorithmus war es den Programmablauf in zwei Phasen einzuteilen. In der ersten Phase, der **Initialisierungsphase**, sollten Daten gesammelt und analysiert werden, um eine erste Vorstellung des Pokerfaces der zu betrachtenden Person zu bekommen. Um dies zu erreichen wird eine gewisse Anzahl an Bildern aufgenommen und die enthaltenen Emotionen validiert. Sollten die Bilder nahezu die selben Emotionen zeigen, könnte man davon ausgehen das diese das Pokerface der Person repräsentieren. Sollten die Bilder allerdings nicht ähnlich zueinander sein, also

verschieden Emotionen zeigen, wird die Initialisierungsphase wiederholt, solange bis der Datensatz an aufgenommenen Bildern genügend Ähnlichkeiten aufweist. Sobald die Daten genügend Aufschluss über ein mögliches Pokerface generiert haben, geht das Programm in die **Hauptphase** über. Hier wird anhand der gewonnenen Daten erkannt, ob ein Pokerface vorliegt oder die Person Emotionen zeigt. Die genaue Umsetzung des Algorithmus kann dem Programmablaufplan in Grafik E.C entnommen werden, welcher im Folgenden erläutert wird.

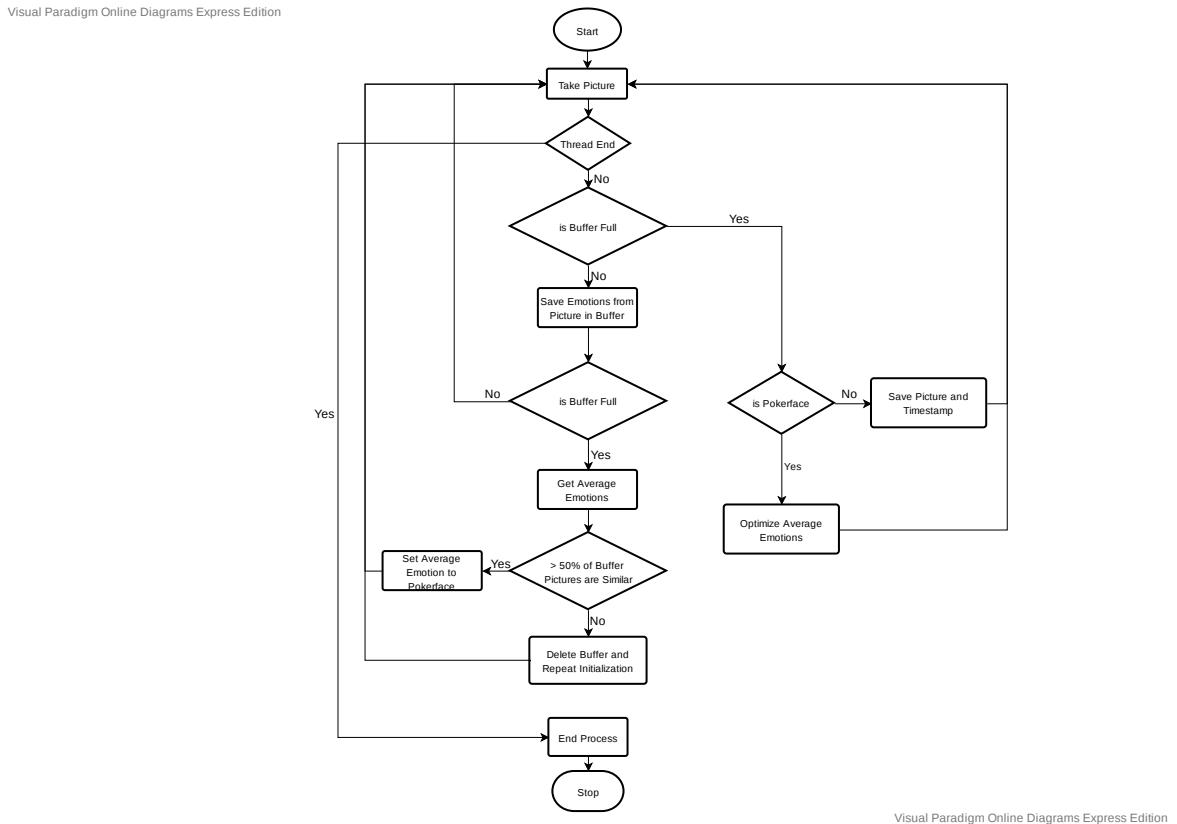


Abbildung E.C: Programmablaufplan des Client-Programms

Der dargestellte Ablauf wird solange durchgeführt, bis der entsprechende Thread beendet wird. Zu Beginn wird ein Bild aufgenommen und über die Verbindung zum Server analysiert. Während der Initialisierungsphase wird dieser Vorgang wiederholt bis ein definierter Buffer gefüllt wurde. Dieser kann beispielsweise mit zehn Plätzen definiert worden sein, demnach zehn Bilder der Person aufgenommen werden und die ausgewerteten Emotionen aller Bilder im Buffer abgespeichert werden. Eine Formalere Formulierung wäre die Buffer Größe der Länge n , wobei $n \in N$, welche die aggregierten Ergebnisse der Server-Evaluierung speichert. Jeder Eintrag innerhalb des Buffers ist wiederum eine Liste, welche die Emotionsvorhersagen beinhaltet. Eine mögliche Struktur eines Buffers der Länge zwei könnte wie folgt aussehen: $[[0.61, 0.30, 0.15, 0.01, 0.76, 0.95], [0.31, 0.22, 0.51, 0.41, 0.01, 0.55]]$ Sobald der Buffer mit den Emotionswerten gefüllt wurde, wird die Initialisierungsphase eingeläutet. Hier wird

zuerst die durchschnittliche Emotionswahrscheinlichkeit aller Emotionen über die gespeicherten Werte berechnet. Am Beispiel der zuletzt genannten Liste würde für die Emotion *Angry*, welche durch die erste Stelle der Liste repräsentiert wird, der Durchschnitt der Werte **0.61** und **0.31** berechnet werden. Diese Rechnung wird für jede Emotion durchgeführt.

Um mit Gewissheit sagen zu können, dass man einen guten Durchschnitt aller Emotionen erhalten hat, welche ein Pokerface repräsentieren, sollten die zugrunde liegenden Daten auch möglichst Ähnlich sein. Werden beispielsweise bei zehn aufgenommenen Bildern verschiedene Grimassen geschnitten, wodurch die Bilder mal ein glückliches Bild und mal ein trauriges Bild zeigen, kann man annehmen, dass der Durchschnitt der Bilder nicht das Pokerface der Person darstellt. Demnach wird geprüft ob mindestens 50% der Bilder innerhalb eines Toleranzbereichs des berechneten Durchschnitts liegen. Bei zehn Bildern sollten also fünf der Bilder möglichst gleiche Emotionen zeigen, wobei die anderen fünf Bilder natürlich keine starken Abweichungen darstellen dürfen. Sollte dies nicht gelingen, kann man keine klare Aussage über das Pokerface der Person treffen und die Initialisierungsphase wird durch das Löschen des Buffers wiederholt. Sollten allerdings die gewünschten Ergebnisse erzielt werden, kann mit den Durchschnittsemotionen ein Fazit über das Vorhandensein eines Pokerfaces getroffen werden und die Hauptphase wird eingeläutet. Hier werden die Analyseergebnisse der Bilder anhand der Durchschnittswerte und einem definierten Toleranzbereich verglichen und als Pokerface oder Nicht-Pokerface eingeordnet. Sollte das Bild als Pokerface eingeordnet werden, werden die Emotionen in die Durchschnittsemotionen mit einberechnet. Dadurch kann die Genauigkeit der Emotionswerte des Pokerfaces während des Spiels gesteigert werden. Sollte das Bild außerhalb des Toleranzbereichs liegen, wird der Zeitpunkt und das dazugehörige Bild gespeichert und der Anwender wird über das Vorliegen eines Nicht-Pokerfaces informiert.

Der beschriebene Algorithmus ist in der Theorie eine gute Umsetzung für das Erkennen eines Pokerfaces während eines Pokerspiel, allerdings sehr fehleranfällig in der Praxis. Das Problem liegt in der Bewegung der zu betrachtenden Person. Sollte die Person beispielsweise Reden, verändert sich der Gesichtsausdruck und somit auch die Emotionswerte, wodurch sich der Algorithmus für ein Nicht-Pokerface entscheidet. Auch wenn die Person sich zurück lehnt oder in eine andere Richtung blickt, werden, selbst wenn sich der Gesichtsausdruck nicht verändert, neue Werte durch das Modell berechnet und die Ergebnisse unbrauchbar. Im schlechtesten anzunehmenden Szenario würde die Person sich nach einer erfolgreichen Initialisierung zurück lehnen und somit könnte man über den gesamten Spielverlauf keine Aussagen mehr über ein Pokerface treffen. Zudem liegt ein weitere Problem in der Annahme, dass die ausgerechneten Emotionswerte auch tatsächlich das Pokerface des Spielers repräsentieren. In der Praxis wäre es wohl eher unüblich den Gegenspieler zu bitten während der Initialisierungsphase sein

Pokerface aufzusetzen. Bei einer Bufferlänge von 10 Bildern und einer Abtastrate des Streams von 0.2 Sekunden würde die Initialisierungsphase 2 Sekunden dauern. Gerade zum Beginn eines Spieles werden sich die Spieler noch miteinander unterhalten und nicht direkt ihr Pokerface aufsetzen. Aus diesem Grund ist es wahrscheinlicher, dass man ein 2 Sekunden dauerndes Lächeln des Spielers als Pokerface aufgreift und das wird kaum der Wahrheit entsprechen. Aus diesen Gründen wurde sich für den zweiten Algorithmus entschieden, welcher im Folgendem vorgestellt wird.

5.2.7.2 Kurzzeit-orientierter Algorithmus

Author — 1329241

Im Gegensatz zu dem letzten Algorithmus analysiert dieser die Kurzaufnahmen und nicht das gesamte Spiel. Wie in dem ersten Algorithmus wird auch hier der Videostream in einer gewissen Rate abgetastet. Diese Momentaufnahmen aus dem Videostream werden ebenfalls dem Server zum Evaluieren gegeben. Dieser Vorgang wird einige Male wiederholt, und die daraus resultierenden Daten aggregiert und zwischengespeichert. Das Ergebnis dieser Aggregation ist dann eine Liste der Länge n , wobei $n \in N$. Jedes Element dieser Liste ist wiederum eine Liste, die die Emotionsvorhersagen für ein bestimmtes Bild aus dem Videostream beinhaltet. So könnte z.B. eine solche Liste der Länge 2 folgende Struktur aufweisen:

$[[0.61, 0.30, 0.15, 0.01, 0.76, 0.95], [0.31, 0.22, 0.51, 0.41, 0.01, 0.55]]$ Wobei jeweils ein Wert für eine Emotion steht bzw. die Wahrscheinlichkeit einer Emotion für das jeweilige Bild. Anhand dieser Liste wird dann wiederum die Abweichung der Emotionen in den Bildern festgelegt. Dazu werden die einzelnen Emotionswerte zuerst extrahiert und zusammengefasst - also alle fröhlichen, ängstlichen, neutralen, etc. Danach werden dann die Unterschiede in den jeweiligen Werten ermittelt. Dies geschieht, indem zuerst der Mittelwert einer Emotion ermittelt wird, und dann die Werte gegen diesen geprüft werden. Weicht dabei eine Emotionsvorhersage zu weit von dem Mittelwert ab, wird zurückgegeben, dass ein Pokerface nicht vorliegen kann. Dies entspricht der Definition eines Pokerfaces in 3.3, da dann je nach Toleranzwert der wegen Messungenauigkeiten des Models eingeplant werden muss, eine zu hohe Abweichung einer Emotion in den Bildern gemessen wurde, und demnach der Gesichtsausdruck nicht beibehalten wurde. Formal beschrieben wertet der Algorithmus also folgende Formel aus:

Eingabe sei eine Liste der Länge 6, wobei jedes Element dieser Liste wiederum eine Liste ist. Ein Element dieser Unterliste heißt dabei $Emotion_e$, wobei $e \in 1 \dots 6$. $Emotion_e$ ist eine Liste der Länge n ist mit den Vorhersage Daten einer einzelnen

Emotion wie z.B. *fear* aus der Eingabeliste. $Emotion_e$ hat die Form

$$Emotion_e = [value_1, \dots, value_n]$$

und

$$average_e = \frac{1}{n} \cdot \sum_{i=1}^n value_i$$

ist der Durchschnitt von $Emotion_e$. Für diese Werte gilt:

$$\forall value_i \in Emotion_e : (average_e - value_i) \leq \epsilon \Rightarrow Pokerface = True$$

ϵ ist hierbei die Toleranz in der sich eine Abweichung der Emotion begeben darf.

Kapitel 6

Ergebnis

Author — 2356667

Nachfolgend werden die aufgrund des erstellten Konzeptes entwickelten Lösungsansätze und Lösungen erläutert. Dazu werden die definierten Anforderungen vor allem auch im Bezug auf das trainierte Modell zum Klassifizieren von den vordefinierten Emotionen und das Verifizieren und Testen dieses Modells auf Erfüllung geprüft.

Eine grundlegende Anforderung ist, dass die Emotionen nicht zufällig erkannt werden. Hier kann gesagt werden, dass diese Anforderung schon in frühen Projektphasen erfüllt werden konnte und die Genauigkeit der richtigen Zuordnung der Testdaten bei 55,71% liegt. Somit ist davon auszugehen, dass vorhandene Emotionen im Durchschnitt auf ca. jedem zweiten Bild richtig klassifiziert werden. Dies kann in der Praxis anhand von eigenen Tests nur so bestätigt werden. Ergänzt man die Tabelle V.II der Häufigkeitsverteilung der Emotionen mit der Genauigkeit der Zuordnung der Testdaten zu den einzelnen Emotionen ergibt sich folgende Tabelle VI.I.

Wert	Emotion	H_n	h_n	Genauigkeit
0	Angry	4953	13,80%	46,76%
1	Disgust	547	1,52%	—
2	Fear	5121	14,27%	33,4%
3	Happy	8989	25,05%	80,83%
4	Sad	6077	16,93%	39,45%
5	Surprise	4002	11,15%	73,65%
6	Neutral	6198	17,27%	49,39%

Tabelle VI.I: Genauigkeit der Zuordnung der Testdaten zu den einzelnen Emotionen

Bei Betrachtung der zwei Ansätze zur Erkennung eines Pokerfaces ist festzustellen, dass bei der Implementierung der Clientanwendung sowohl der langzeitorientierte als auch der kurzzeitorientierte Algorithmus zwischen Pokerface und kein Pokerface unterscheiden kann. Durch ausführliche Praxistests kann die grundsätzliche Funktionstüchtigkeit für beide Algorithmen validiert werden. Die letzte Must-Anforderung ist die Implemen-

tierung einer benutzerfreundlichen Interaktionsmöglichkeit, um Input zu liefern und Output darzustellen. Dabei stellt der im Projektumfang definierte Output lediglich die Information dar, ob ein Pokerface vorhanden ist oder nicht. Die erste Teilanforderung wurde schon in den ersten Entwürfen erfüllt, da ohne besondere Interaktion des Anwenders die Bilder der Webcam bzw. des Video-Streams automatisiert als Input genutzt wurden. Die am Anfang erstellte OpenCV-Anwendung dient auch als Grundlage für die finale Version, weshalb die Teilanforderung dort ebenfalls erfüllt ist. Durch die einfache Textausgabe auf dem gespiegelten Video-Stream in der Clientanwendung, ob ein Pokerface vorhanden ist oder nicht, wird auch die zweite Teilanforderung, dass ein Output darzustellen ist, erfüllt.

Dadurch dass die Genauigkeit zur Erkennung der richtigen Emotion bei über 50% liegt, gilt auch die erste Anforderung der Should-Klassifizierung als erfüllt. Die genannte Genauigkeit ergibt sich aus den in Tabelle VI.I dargestellten Daten. Hierbei ist zu bemerken, dass aufgrund der signifikant geringen Anzahl an Daten zu der Emotion **Disgust** entschieden wurde, im Umfang des Projektes lediglich die restlichen sechs Emotionen zu berücksichtigen. Somit wird zwischen mehr als fünf verschiedenen Emotionen differenziert und auch die zweite Anforderung der Should-Kriterien kann als erfüllt angesehen werden. Wie ebenfalls in den Anforderungen definiert, soll die Erkennung einer Emotion und eines Pokerfaces innerhalb eines Pokerspielzuges eines Spielers, also innerhalb von 20 Sekunden erfolgen. Nach ausgiebigen Praxistests der Clientanwendung kommt man zu dem Ergebnis, dass die Bilder bei einer guten Netzwerkverbindung zum Server in nahezu Echtzeit analysiert und die Ergebnisse an den Client zurückgesendet werden. Für zuverlässige Aussagen bezüglich des Pokerfaces müssen in beiden Algorithmen Referenzwerte gesammelt werden. Je nach Konfiguration der Anzahl der Referenzwerte und der Zeitdifferenz zwischen den Referenzwerten, vergehen bis zum ersten Erkennen eines Pokerfaces zwischen ca. fünf und zehn Sekunden. Beide Algorithmen laufen direkt auf dem Client und die Analyse der Emotionen erfolgt nahezu in Echtzeit, deshalb liegt die Dauer der Erkennung eines Pokerfaces nach Programmstart bei ca. 11 Sekunden und die reine Dauer zur Erkennung eines Pokerfaces liegt bei unter einer Sekunde. Somit gilt auch die letzte Should-Anforderung als erfüllt.

Wie bereits mehrfach erwähnt, werden Bilder in nahezu Echtzeit analysiert, sodass die erste Anforderung nach der Could-Klassifikation ebenfalls als erfüllt gilt. Eine weitere Anforderung in dieser Kategorie ist das Trainieren des Modells innerhalb eines Arbeitstages. Bei den ersten Modellen, welche auf dem Dataset der FER-Challenge über 50 bis 100 Epochen trainiert wurden, lag die Trainingsdauer mit ca. 14 Stunden deutlich über einem Arbeitstag. Bei der Optimierung des Modells und dem Feintuning der Hyperparameter Batchsize und Epochs konnte die Genauigkeit des Modells erheblich gesteigert werden, wobei sich als Nebeneffekt die Trainingsdauer auf ca. 30 Minuten signifikant verbessert hat. Somit wurde mit der Optimierung des Modells auch indirekt die zuvor

angesprochene Anforderung erfüllt. Da einzelne Bilder des Video-Streams der Webcam als Input zur Analyse dienen und diese automatisiert an den Server übermittelt werden sowie auch der Output mit der Information, ob ein Pokerface vorhanden ist oder nicht, automatisiert in der GUI ausgegeben wird, sind von Seiten des Nutzers keinerlei Interaktionen erforderlich. Dieser muss lediglich die Anwendung, also das Jupyter Notebook bzw. das Python-Skript ausführen. Dem Nutzer werden zur Interaktion keine Auswahlmöglichkeiten gegeben, weshalb für die reine Bedienung der Oberfläche kein technisches Wissen benötigt wird, jedoch sollte zum Starten des Skriptes ein geringes technisches Verständnis vorhanden sein. Aufgrund der MoSCoW-Klassifizierung dieser Anforderung als Could, kann diese nur zu einem Großteil erfüllt werden, was jedoch aufgrund der Klassifizierung kein Problem darstellt.

Das Ziel des Projektes ist nicht die Analyse von ganzen Videos. Da immer einzelne Emotionen auf einzelnen Bildern analysiert werden, kann man hier zwar von der eingeschränkten Erfüllung der Anforderung sprechen, jedoch ist es nicht möglich, direkt ein Video zu analysieren. So wie bei der Anforderung, dass das Modell bei jeder Belichtung valide Ergebnisse liefern soll, handelt es sich um Anforderungen der Won't-Klassifizierung. Daher stellt es kein Problem dar, dass zur Analyse von Emotionen auf Bildern zumindest eine mäßige Belichtung notwendig ist, wobei für zuverlässige Ergebnisse verschiedene Belichtungseinstellung getestet werden sollten.

Kapitel 7

Diskussion

Author — 8774695

In dem letzten Kapitel dieser Arbeit werden die Ergebnisse, das Vorgehen und die verwendete Literatur in diesem Projekt kritisch betrachtet und reflektiert. Zuletzt sollen dem Leser noch einige Ideen vorgestellt werden, in welche Richtung die entwickelte Lösung in Zukunft möglicherweise weiterentwickelt werden kann und welche Änderungen und Verbesserungen dafür notwendig wären. Ziel dieses Kapitels ist es dem Leser die positiven und negativen Aspekte des Projektablaufs aufzuzeigen und in letzterem Fall welche Hürden diese beeinflusst haben. Hierdurch sollen dem Leser Anregungen gegeben werden, auf welche Einflussfaktoren bei künftig gleichartigen Projekten im Bereich der Gesichts- / Emotionserkennung zu achten sind. Doch bevor die kritische Betrachtung im folgendem Vorgenommen wird, werden noch einmal kurz die Rahmenbedingungen des Projektes genannt. Das Projektteam besteht aus drei Studenten im sechsten Semester, welche vor dem Start des Projektes kaum nennenswerte Vorkenntnisse im Bereich von neuronalen Netzwerken und der Programmiersprache Python hatten. Für die Lösung der Aufgabenstellung standen insgesamt 32 Wochen zur Verfügung, welche innerhalb der Studien- und Arbeitszeit statt fand.

7.1 Reflexion der Ergebnisse

Author — 8774695

Wie in dem letzten Kapitel zu sehen ist, wurden die definierten Anforderungen erfolgreich Umgesetzt. Die Bewertung der Anforderungen selbst wird in dem folgendem Kapitel genauer diskutiert. Nachdem der Erfolg des Projektes größtenteils von dem entwickelten Modell abhängig war, wird auf dieses zuerst eingegangen.

Der größte Einflussfaktor auf ein durch neuronale Netze berechnetes Modell ist das ver-

wendete Datenset, und von diesem ist man auch sehr stark abhängig. Hier präsentiert sich schon das erste Hindernis vor welchem das Projektteam in der Entwicklung stand. zu Beginn der Entwicklung wurde viel Zeit investiert um ein passendes Datenset zu finden, welches für die Erstellung des Modells geeignet war. Die Daten sollten demnach im besten Fall gelabelt nach verschiedenen Emotionen sein und auch möglichst viele Daten enthalten um ein gutes Modell entwickeln zu können. Das erste gefundene Datenset wurde in Kapitel 4.2.2 vorgestellt und erfüllt auf den ersten Blick die Anforderungen an den Datensatz. Mit insgesamt 10.000 gelabelten Bildern schien der Datensatz optimal geeignet zu sein. Allerdings waren keine reinen Emotionen, sondern ein Emotionsverlauf durch die Daten dargestellt. Durch diese Einschränkung konnten lediglich das erste und das letzte Bild verwendet werden, zum einen für die neutrale Emotion und zum anderen für die “Reinform“ des dargestellten Emotionsverlaufs . Aus den anfangs 10.000 Bildern waren damit nur noch 325 Bilder zu gebrauchen und diese Anzahl war viel zu gering um ein brauchbares Modell erstellen zu können. Zu dieser Problematik kam zudem der Aufbau der Verzeichnisstruktur. Insgesamt wurde hierbei viel Zeit und Energie investiert zur Aussortierung der Bilder und zu der Zuweisung zu den entsprechenden Emotionen. Das erste Datenset war zwar sehr hilfreich um sich in die Programmiersprache Python einzuarbeiten, insbesondere im Bereich der Bild Bearbeitung, dem Verzeichniswechsel und dem Erstellen erster Modelle, aber das daraus resultierende Modell war nicht geeignet für die Erfüllung der Aufgabe.

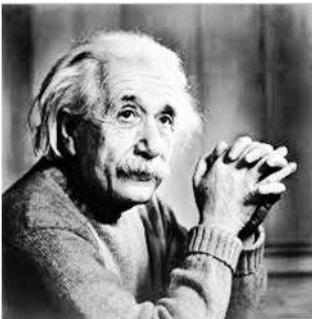
Glücklicherweise wurde im Laufe des Projektes ein zweiter Datensatz gefunden, welcher in Kapitel 5.2.1 vorgestellt wurde. Der Datensatz bestand aus insgesamt 35.887 Bildern, also mehr als drei mal so viele Bilder wie der erste Datensatz, von welchem auch alle Bilder verwendbar waren. Durch die Abspeicherung der Bilder in einer Tabelle, konnten die entsprechenden Labels auch schneller der dazugehörigen Zeile entnommen werden als in dem ersten. Auch waren die Bilder bereits eingeteilt für das Trainieren und Testen des Modells, was einen weitere Vereinfachung für die Verwendung der Daten darstellte. Das einzige Problem des Datensatzes stellte die Verteilung der einzelnen Bilder auf die Emotionen dar, welche der Abbildung E.A entnommen werden kann. Mit knapp 4.000 Bilder der Emotion überrascht und fast 9.000 glücklichen Bildern sind diese sehr ungleichmäßig auf den Emotionen verteilt. Dies hatte auch eine Auswirkung auf Genauigkeit der Emotionserkennung, welche in der Tabelle VII.1 dargestellt sind. Hätte die Aufgabe gelautet, nur die dargestellte Emotion in einem Bild zu erkennen, wäre das Datenset nicht besonders gut geeignet gewesen. Doch die Aufgabe war es ein Pokerface zu erkennen und somit nur eine Emotionsänderung feststellen zu können und daher hatte die Verteilung keine große negative Auswirkung auf das Modell. Auch wenn das Modell eine überwiegend glückliche Emotion errechnet, obwohl die Person einen neutralen Ausdruck angenommen hat, wirkt sich diese falsche Einschätzung nicht auf das Endresultat aus. Werden weitere ähnliche Bilder mit der falschen Einschätzung

des Modells berechnet, zeigt das Programm dennoch korrekt an ob eine Emotionsveränderung vorliegt oder nicht. Die entscheidende Botschaft aus diesem Absatz ist, dass der Erfolg des Projektes stark abhängig von dem Finden eines passenden Datensatzes ist. Ohne diesen Datensatz, hätte das Projektteam nicht die Möglichkeit gehabt mit den zur Verfügung stehenden Ressourcen und dem zeitlichen Rahmen die notwendige Menge an Daten selbst zusammen zu stellen, und die Anforderungen hätten nicht erfüllt werden können.

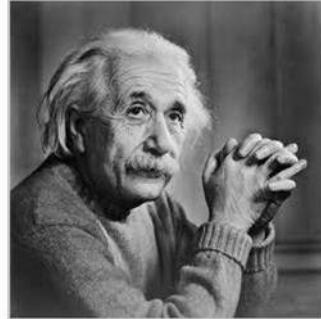
Ein weiterer wichtiger Punkt für die Genauigkeit des Modells ist der Prozess des Image Pre-Processing, also der Vorbearbeitung der Bilder, wodurch alle Bilder in der selben Form dargestellt werden. Dieser beinhaltete in dem Projekt lediglich die Größenanpassung der Bilder. Allerdings gibt es noch zahlreiche Möglichkeiten um die Bilder weiter anzupassen und somit die Ergebnisse des Modells zu verbessern. Eine Methode ist die sogenannte *Histogram Equalization*, durch welche das Histogramm eines Bildes derart geändert wird, dass dieses Konstant für jegliche Beleuchtungsstufen ist. Dies hätte das entstandene Modell erheblich verbessert, da genau dieser Punkt eine Schwachstelle des Produktes ausmacht. Die Genauigkeit der Emotionserkennung ist in der bisherigen Lösung stark abhängig von der Beleuchtung. Allerdings liegt die Entsprechende Anforderung auch im Bereich *Won't*, weshalb eine Umsetzung dieser nicht erforderlich war. Wie ein derart Bearbeitetes Bild aussehen kann ist in Abbildung G.A zu sehen. Für weitere Anregungen zur Image Pre-Processing Methoden, kann das Journal ***Preprocessing Technique for Face Recognition Applications under Varying Illumination Conditions*** von ***Global Journal of Computer Science and Technology Graphics & Vision*** studiert werden.

Weshalb wurden diese Methoden nicht verwendet um das Modell zu verbessern? Das Problem liegt in den Rahmenbedingungen. Die Projektmitglieder hatten nicht das Hintergrundwissen wie diese Methoden in dem Modell implementiert werden konnten und waren sich auch nicht der verschiedenen Methoden bewusst. Die Möglichkeiten zur Image Pre-Processing sind derart Vielzählig und komplex, dass sie in einer eigenen Studienarbeit bearbeitet werden können. Natürlich kann das Modell noch stark verbessert werden durch solche Methoden, doch die definierten Ziele konnten mit der jetzigen Lösung erreicht werden, ohne den zeitlich vorgegebenen Rahmen zu überschreiten. Doch für eine Weiterentwicklung dieses Modells rentiert es sich, solche Methoden weiter zu studieren und in das Modell mit aufzunehmen.

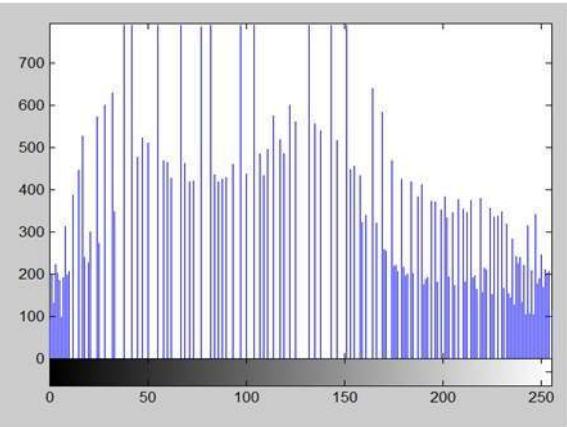
New Image



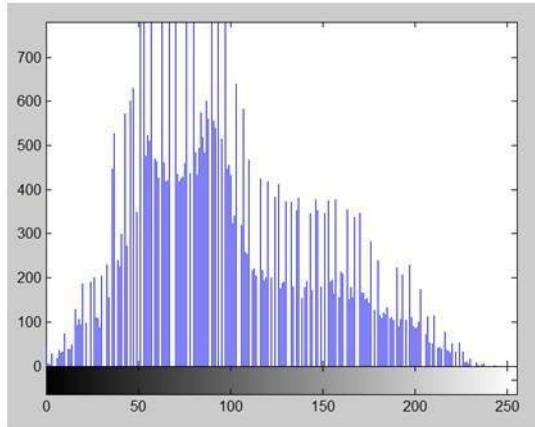
Old image



New Histogram



Old Histogram



Quelle: https://www.tutorialspoint.com/dip/histogram_equalization.htm
Abbildung G.A: Bearbeitetes Bild mittels Histogram Equalization

7.2 Reflexion Vorgehen

Author — 8774695

Grundsätzlich wurde das Projekt strukturiert durch die Erstellung, Priorisierung und Abarbeitung von Anforderungen durchgeführt. Auf die Verwendung eines Vorgehensmodells wurde verzichtet, da zum einen das Projekt aus nur drei Mitgliedern bestand und zum anderen das Umfeld für die Projektmitglieder sehr neu war. Daher war die Planung und Einteilung einzelner Phasen, wie bei dem Wasserfallmodell schwer umzusetzen und hätte keinen deutlichen Mehrwert gebracht. Die Verwendung eines Spiralmodells, wie es in technisch neuartigen Umfeld häufig verwendet wird, war zu schwerfällig und zeitaufwendig. Von daher wurde sich lediglich auf die Definition von Anforderungen gestützt, welche auch erfolgreich umgesetzt wurden. Allerdings stellt sich die Frage, ob die Anforderungen auch korrekt gewählt wurden. Dies soll im folgenden diskutiert werden.

Anforderungen sollten im Allgemeinen einige Rahmenbedingungen erfüllen, damit sie verständlich und bewertbar sind. Darunter fallen die folgenden Punkte:

- **Korrektheit:** Die Anforderung entspricht den Bedürfnissen der Stakeholder.
- **Eindeutigkeit:** Die Formulierung der Anforderung erlaubt nur eine gültige Interpretation.
- **Prüfbarkeit:** Es lassen sich Kriterien oder Tests angeben, um nachzuweisen, ob die Anforderung erfüllt ist oder nicht.
- **Nachverfolgbarkeit:** Der Ursprung der Anforderung und ihre Entwicklung lassen sich feststellen.

Zur Bewertung des ersten Punktes, müssen zuerst die Stakeholder des Projektes identifiziert werden. Zu diesen zählen zum einen die Projektbeteiligten und zum anderen dem betreuenden Dozenten. Nachdem die Anforderungen durch alle drei Projektbeteiligten entworfen wurden, kann auf deren Seite eine Korrektheit der Anforderungen bestätigt werden. Mit dem betreuenden Dozenten wurde regelmäßig, vor allem zu Beginn des Projektes, Absprache gehalten. Hierbei wurden die Ziele und Aufgabenstellung des Projektes gemeinsam diskutiert aus welchen auch die Anforderungen hervor gingen. Von daher kann auch von dieser Seite eine Korrektheit der Anforderungen angenommen werden. Schlussfolgernd gilt der erste Punkt als erfüllt.

In Bezug auf die Eindeutigkeit der Anforderungen, kann man diese für die meisten bestätigen. Allerdings sind vor allem die Anforderungen einer benutzerfreundlichen Interaktionsmöglichkeit und einer Oberfläche zur intuitiven Bedienung stark individuell abhängig. Eine intuitive Bedienung ist für den Techniker schneller gegeben als für den Manager, dessen Erwartungen an solch eine meist höher gesteckt sind. Doch in der frühen Entwicklungsphase wurde sich dafür entschieden diese Anforderungen mit aufzunehmen zu müssen, auch wenn es schwierig war die entsprechende Anforderung eindeutig zu beschreiben. Das Problem wurde auch gewitzt umgangen, indem der Anwender gar nicht erst mit der Anwendung interagieren muss.

Auch für die Prüfbarkeit wurden für die meisten Anforderungen feste Werte oder eindeutige Zeitrahmen definiert um deren Erfüllung bewerten zu können. Aus dem Rahmen fallen auch hier die bereits genannten Anforderungen aus den gleichen Gründen. Bei der Nachverfolgbarkeit muss an dieser Stelle eingestanden werden, dass diese nicht vollständig erfüllt wurde. Der Ursprung der Anforderungen kann auf die ersten Wochen der Bearbeitungszeit festgelegt werden, doch deren genaue Entwicklung ist nicht nachverfolgbar. Der Grund liegt in der fehlenden Dokumentation während dem Entwicklungsverlaufs. Als Dokumentation dient diese Arbeit, jedoch existiert keine genauere Dokumentation, welche Anforderungen an welchen Bearbeitungstagen bearbeitet oder fertig gestellt wurden. Auf diese wurde gezielt verzichtet, da eine Dokumentation einen

zusätzlichen Arbeitsaufwand und damit verbunden einen erhöhten Zeitaufwand bedeutet hätte. In einem größeren Team müssen die Tätigkeiten nach verfolgbar sein, da sonst das Scheitern des Projekts möglich ist, doch in diesem Projekt waren lediglich drei Entwickler beteiligt, wodurch auf eine Dokumentation verzichtet werden konnte ohne den Erfolg des Projektes zu gefährden.

Zuletzt stellt sich die Frage, ob die gegebenen Anforderungen auch vollständig sind. Nach dem aktuellen Kenntnisstand hätten weitere Anforderungen an das Projekt gestellt werden können, wie detailliertere technische Anforderungen im Bezug auf die minimal benötigten Ressourcen der Hardware um nur ein Beispiel zu nennen. Doch zusammenfassend kann gesagt werden, dass die Anforderungen situationsbedingt gut gewählt wurden. Nachdem alle Projektbeteiligten neu in diesem Bereich waren, war es nicht möglich genauere Anforderungen zu stellen, da das entsprechende Wissen fehlte. Die Anforderungen mögen nicht exakt nach der Definition vollständig formuliert worden sein, doch es wurde sich dafür entschieden mit diesen Anforderungen zu arbeiten um ein strukturiertes Vorgehen zu ermöglichen. Bei keiner Vorgabe, wäre die Bewertung eines Erfolgs des Projektes nicht möglich gewesen.

7.3 Reflexion der Literatur

Author — 8774695

Bezüglich der Literatur ergeben sich nun einige Schwierigkeiten. Dies liegt unter anderem daran, dass das generelle Thema der Gesichts und Emotionserkennung immer noch vor allem aus psychologischer Sicht in der Literatur behandelt wurde. Zwar gibt es Fachbücher auch aus informationstechnischer Sicht, welche ebenfalls in dieser Arbeit verwendet wurden.

Für die Recherche dieser Arbeit wurden 30 Quellen verwendet, von denen 9 Quellen aus Büchern verwendet wurden und 21 Quellen aus online Recherchen hervorgingen. Die 30% der schriftlichen Quellen wurden aus Fachbüchern aus diesem Gebiet oder einem verwandten Gebiet des Themas dieser Arbeit verwendet, wodurch eine gute theoretische Einarbeitung in die Thematik ermöglicht wurde. Die Online Quellen wurden größtenteils für technische Lösungsansätze und Bereiche verwendet, welche durch die Fachbücher nicht gedeckt werden konnten. Dabei wurden offizielle Dokumentationen verwendet, die eine optimale Informationsquelle für technische Details der Software bietet, aber nicht für die Bewertung dieser. Diese Dokumentationen wurden jedoch lediglich für technische Hintergrundinformationen verwendet. Des Weiteren wurden auch Handbücher und Blogartikel verwendet, welche zwar keine wissenschaftliche Grundlage bieten, aber gute Lösungsansätze für die gestellten Aufgaben anboten. Es wurde im

laufe der Arbeit darauf geachtet seriöse und vertrauenswürdige Quellen für die Recherche zu nutzen und keine subjektiven und nicht nach verfolgbaren Verweise in die Arbeit mit einfließen zu lassen.

Wie eingangs erwähnt, stellt die Recherche von Fachliteratur in Bezug auf Emotionserkennung im technischen Umfeld eine Herausforderung dar. Die schriftlichen Quellen behandeln meist nur die übergeordneten Themen der Künstlichen Intelligenz im Allgemeinen oder mehr in Richtung der Gesichtserkennung. Allerdings waren die Ziele dieser Arbeit mehr praktisch Motiviert und weniger auf die Beantwortung wissenschaftlicher Fragestellungen ausgelegt. Aus diesem Grund gilt die verwendete Literatur als gute Basis für die Erfüllung dieser Arbeit.

7.4 Offene Implikationen

Author — 8774695

Dieses Kapitel soll einen kurzen Überblick über mögliche Weiterentwicklungen der erstellten Lösung bieten. Das entwickelte Modell, sowie die Client-Seitige Software bietet eine solide Grundlage um weitere Aufgaben erfüllen zu können, aber natürlich stellt es nur eine erste Umsetzungen mit einigen Schwachstellen dar. Eine erste Möglichkeit der Weiterentwicklung bieten die *Won't* Kriterien. Das Modell kann derzeit lediglich einzelne Bilder analysieren, welche aus dem Lifestream periodisch ausgelesen werden. Eine interessante Möglichkeit wäre das Analysieren ganzer Videostreams. Die zweite Anforderung, welche definitiv einen Mehrwert für das Modell erzielen würde, wäre die Vorbearbeitung der Bilder um in sämtlichen Belichtungen die Bilder analysieren zu können, wie es in der Reflexion der Ergebnisse bereits vorgestellt wurde. Dadurch könnte man die Lösung auch für Poker-Spiele in zwielichtigen Nebenräumen einsetzen. Des weiteren könnte das Client-Programm weiter entwickelt werden, um eine Nutzung zu vereinfachen. So könnte zum Beispiel automatisch durch Bilderkennung die Emotionsänderung der Spieler gezielt in dem Moment geprüft werden, wenn ein neues Blatt aufgedeckt wird. Durch weitere Modelle könnte auch eine Analyse der Beziehung zwischen dem Gesichtsausdruck der einzelnen Spieler und dem Blatt erfolgen, wodurch die Chancen auf einen Gewinn des Benutzers errechnet werden. Im Idealfall muss der Nutzer nur noch die Anzahl der Chips die ihm das Programm nennt zur Richtigen Zeit einsetzen. Des weiteren könnten auch alternative Einsatzszenarien neben dem typischen Pokerspiel diskutiert werden. Solche wurden bereits eingangs in der Einleitung vorgestellt, wie zum Beispiel der Einsatz in Verhören durch die Polizei oder der Verwendung in einem Gerichtsfall. Natürlich bietet die Lösung noch nicht die Genauigkeit um in einem solchen Umfeld gezielt eingesetzt werden zu können, doch für die Zukunft

wären Modelle dieser Art für die genannten Einsatzgebiete durchaus vorstellbar.

7.5 Alternative Ansätze zur Umsetzung von Emotionserkennung

Author — 1329241

In diesem Abschnitt nun werden verschiedene alternative Ansätze dargestellt und expliziert, die dazu verwendet werden können um Emotionen zu erkennen. Dieses Unterkapitel beschäftigt sich mit alternativen Ansätzen zu den bereits explizierten Basismotionen. Diese sind wie bereits erwähnt umstritten, was die Frage zulässt warum diese überhaupt verwendet werden sollten. Alternativen dazu bieten weitere kreative Ansätze, wie die Erkennung von Emotionen anhand der Stimmlage. Dieser Ansatz beruft sich darauf, dass das Sprachzentrum eines Menschen einer der wichtigsten Aspekte der Kommunikation und somit auch der Preisgabe von Informationen über den emotionalen Zustand eines Individuums ist.¹ Dieser Ansatz ist jedoch nicht zielführend, da hier hauptsächlich die Stimme analysiert wird. Von einer Stimme kann nun auf eine Emotion geschlossen werden. Für den Usecase ist dieser Ansatz allerdings ungeeignet, aus folgenden Gründen:

Es kann möglich sein eine Emotion anhand der Sprache zu erkennen. Das Äquivalent eines Pokerfaces wäre dementsprechend je nach Definition eine neutrale Stimmlage, welche keine Emotionen suggeriert. Nun kann aber keine Aussage getroffen werden aus welchen Gründen eine Person neutral spricht. Es könnte von einem Pokerface stammen, oder einer monotonen Sprechweise, oder einen gelangweilten Gemütszustand. Dies ist nicht eindeutig identifizierbar. Ein weiterer Ansatz wäre die Analyse der derzeit vernommenen Musik. Diese kann einem bestimmten Gemütszustand zugesprochen werden, welches auf eine aktuelle Emotion übertragbar ist.² Ziel dieses Forschungszweiges ist es daher die hinter Liedern oder Klängen stehenden Emotionen zu ermitteln und diese entsprechend zu kategorisieren. Dieser Ansatz erscheint zunächst durchaus interessant, hat jedoch genauso Nachteile wie die Analyse von Emotionen anhand der Stimmlage. Der Größte liegt hier unter anderem in der Genauigkeit der Analysen. So z.B. lieferte ein Testprojekt an der Russischen HSE (Higher School of Economics) das Ergebnis von einer maximalen Genauigkeit von 71%.³ In dem Versuchsaufbau wurden Spektrogramme von Klangfragmenten ausgewertet und versucht mittels Neuronalen Netzen eine

¹Vgl. Rao und Koolagudi, *Emotion Recognition using Speech Features*, Abstract.

²Vgl. Yang und Chen, *Music Emotion Recognition*, S. 1.

³Vgl. Popova, Rassadin und Ponomarenko, *Emotion Recognition in Sound*, Abstract.

Klassifikation der hinter dem Klang liegenden Emotion zu erreichen.⁴ Der generelle Ansatz anhand von Musik die Emotion eines Individuums abzulesen ist zwar praktikabel und von dem Versuchsaufbau auch vergleichbar zu dem Ansatz bereits gelabelte Bilder zu verwenden. Jedoch lässt sich auf diese Weise aus zwei Gründen nicht die eigentliche Zielaufgabenstellung ableiten - das Erkennen eines Pokerfaces. Zum einen handelt es sich in dieser Arbeit um eine visuelle Problemstellung, in welcher das Erkennen des Gemütszustandes anhand des Gesichtsausdruckes erkannt werden soll, also einem vorhandenen bzw. nicht vorhandenen Pokerface. Zum anderen würde die Analyse von Musik einen Rückschluss auf den allgemeinen Gemütszustand des Betroffenen folgern und nicht eine kurzzeitige Stimmungsschwankung aufgrund beispielsweise eines schlechten Kartenblattes.

Anhand der hier gezeigten Alternativen kann Emotionsanalyse auf verschiedenste Weise implementiert werden. Die Pokerface-Erkennung hingegen kann lediglich visuell erfolgen, weshalb die hier explizierten Ansätze dafür nicht valide sind.

⁴Vgl. ebd., Abstract.

Literatur

Literaturquellen

- [1] Katherine B. Leeland. *Face Recognition: New Research*. 1. Aufl. Nova Science Publishers INC, 2008. ISBN: 978-16045646625.
- [2] Anil K. Li Stan Z.and Jain. *Handbook of Face Recognition*. Springer Science und Business Media, 2005. ISBN: 978-0387405957.
- [6] Schneider Patrickand Witschi Urs und Wüst Roger. *Handbuch Projektmanagement: Agil – Klassisch – Hybrid*. Auflage 4. Sebastopol: SSpringer Gabler”, 2018. ISBN: 978-3662578773.
- [7] R Benedict Saranya und J. Satheesh Kumar. *Geometric shaped facial feature extraction for face recognition*. Coimbatore - India: IEEE, 2017. ISBN: 978-1509037698.
- [16] Benjamin Johnston und Ishita Mathur. *Applied Supervised Learning with Python*. Auflage 1. Birmingham: Packt Publishing Ltd, 2019. ISBN: 978-1789955835.
- [17] Bonaccorsor Giuseppe. *Hands-On Unsupervised Learning with Python*. Auflage 1. Birmingham: Packt Publishing Ltd, 2019. ISBN: 978-1789349276.
- [28] K. Sreenivasa Rao und Shashidhar G. Koolagudi. *Emotion Recognition using Speech Features*. 1. Auflage. Berlin, Deutschland: Springer Science und Business Media, 2012. ISBN: 978-1461451433.
- [29] Yi-Hsuan Yang und Homer H. Chen. *Music Emotion Recognition*. Boca Raton Florida: CRC Press, 2011. ISBN: 978-1466508927.
- [30] Anastasiya S. Popova, Alexandr G. Rassadin und Alexander A. Ponomarenko. *Emotion Recognition in Sound*. Cham, Deutschland: Springer Verlag, 2017. ISBN: 978-3319666037.

Sonstige Quellen

- [3] Maximilian Schreiner. *Künstliche Intelligenz: Emotionserkennung will gelernt sein.* <https://mixed.de/kuenstliche-intelligenz-emotionserkennung-will-gelernt-sein/>. Einsichtnahme:24.01.2020. 2019.
- [4] Dr. Frederik Diederichs. *KI-gestützte Emotionserkennung im Fahrzeug aus physiologischen Daten.* <https://www.hci.iao.fraunhofer.de/de/Human-Centered-AI/feinfuehligetechnik/KI-gestuetzte-Emotionserkennung.html>. Einsichtnahme:24.01.2020. 2020.
- [5] Johannes Kuhn. *Emotionen sind schwer definierbar.* <https://www.sueddeutsche.de/digital/smash/software-emotionen-simulation-ki-1.4377004-2>. Einsichtnahme:24.01.2020. 2019.
- [8] Mateusz Opala. *Top Machine Learning Frameworks Compared: Scikit-Learn, Dlib, MLlib, Tensorflow, and More.* <https://www.netguru.com/blog/top-machine-learning-frameworks-compared>. Einsichtnahme:07.03.2020. 2018.
- [9] *Dlib C++ Library.* <http://dlib.net/>. Einsichtnahme:07.03.2020. 2019.
- [10] Nico Litzel Stefan Luber. *Was ist Keras?* <https://www.bigdata-insider.de/was-ist-keras-a-726546>. Einsichtnahme:24.01.2020. 2018.
- [11] *Keras backends.* <https://keras.io/backend/>. Einsichtnahme:24.04.2020.
- [12] *Announcement of Theano's discontinuation.* <https://groups.google.com/forum/#!topic/theano-users/7Poq8BZutbY>. Einsichtnahme:24.04.2020.
- [13] *TensorFlow 1.4.0.* <https://github.com/tensorflow/tensorflow/releases/tag/v1.4.0>. Einsichtnahme:24.04.2020.
- [14] *Keras 2.3.0.* <https://github.com/keras-team/keras/releases/tag/2.3.0>. Einsichtnahme:24.04.2020.
- [15] *Einführung in Keras.* <https://datasolut.com/einfuehrung-in-keras/>. Einsichtnahme:15.03.2020.
- [18] *Aktivierungsfunktionen, ihre Arten und Verwendungsmöglichkeiten.* <https://www.ai-united.de/aktivierungsfunktionen-ihre-arten-und-verwendungsmaeglichkeiten/>. Einsichtnahme:12.03.2020.
- [19] *Aktivierungsfunktionen: Neuronale Netze.* <https://www.ai-united.de/aktivierungsfunktionen-neuronale-netze/>. Einsichtnahme:12.03.2020.
- [20] Will Berger. *DEEP LEARNING HAAR CASCADE EXPLAINED.* <http://www.willberger.org/haar-explained/>. Einsichtnahme: 22.03.2020. 2018.
- [21] *Wikipedia - Integralbild.* <https://de.wikipedia.org/wiki/Integralbild>. Einsichtnahme: 22.03.2020. 2017.
- [22] *Face recognition using OpenCV and Python.* <https://www.superdatascience.com/blogs/opencv-face-recognition>. Einsichtnahme: 23.03.2020. 2017.

- [23] *Opencv Dokumentation*. https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tut.html.
Einsichtnahme: 23.03.2020. 2019.
- [24] *Opencv Dokumentation*. https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tut_binary-patterns-histograms.html.
Einsichtnahme: 01.04.2020. 2019.
- [25] Jeffrey F. Cohn, Takeo Kanade und Yingli Tian. *Comprehensive Database for Facial Expression Analysis*. <http://www.consortium.ri.cmu.edu/ckagree/>.
Einsichtnahme: 07.03.2020. 2000.
- [26] Ian Goodfellow u. a. *Challenges in Representation Learning: A report on three machine learning contests*. 2013. URL: <http://arxiv.org/abs/1307.0414%5C%7D>.
- [27] *A guide to an efficient way to build neural network architectures*. <https://towardsdatascience.com/guide-to-an-efficient-way-to-build-neural-network-architectures-part-i-hyper-parameter-8129009f131b>.
Einsichtnahme: 21.04.2020.

Anhang

Verwendeter Code

Listing 7.1: Code für GUI mit Video Stream der Webcam als Output

```
1 import cv2 as cv
2
3 cam = cv.VideoCapture(0)
4 cv.namedWindow("InputGUI1")
5 img_counter = 1
6 while True:
7     ret, img = cam.read()
8     cv.imshow("InputGUI1", img)
9     if not ret:
10         break
11     k = cv.waitKey(1)
12     if k%256 == 27: # ESC pressed, closing the window
13         break
14     elif k%256 == 32: # SPACE pressed, save image as input
15         cv.imwrite("InputGUI{}.png".format(img_counter), img)
16         img_counter += 1
17
18 cam.release()
19 cv.destroyAllWindows()
```

Listing 7.2: Code für GUI mit Video Stream der Webcam und markierten Gesichtern als Output

```
1 import cv2 as cv
2
3 face_cascade = cv.CascadeClassifier('haarcascades/
4     haarcascade_frontalface_default.xml')
5 cam = cv.VideoCapture(0)
6 cv.namedWindow("InputGUI")
7 img_counter = 0
8 while True:
9     ret, img = cam.read()
10    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
11    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
12    for (x,y,w,h) in faces:
13        img = cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
14    cv.imshow("InputGUI", img)
15    if not ret:
16        break
17    k = cv.waitKey(1)
18    if k%256 == 27: # ESC pressed, closing the window
19        break
20    elif k%256 == 32: # SPACE pressed, save image as input
21        cv.imwrite("InputGUI{}.png".format(img_counter), img)
22        img_counter += 1
23
24 cam.release()
25 cv.destroyAllWindows()
```

Listing 7.3: Code für GUI mit Video Stream der Webcam und markierten Gesichtern und Augen als Output

```

1 import cv2 as cv
2
3 face_cascade = cv.CascadeClassifier('haarcascades/
4 haarcascade_frontalface_default.xml')
5 eye_cascade = cv.CascadeClassifier('haarcascades/
6 haarcascade_eye.xml')
7 cam = cv.VideoCapture(0)
8 cv.namedWindow("InputGUI")
9 img_counter = 0
10 while True:
11     ret, img = cam.read()
12     gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
13     faces = face_cascade.detectMultiScale(gray, 1.3, 5)
14     for (x,y,w,h) in faces:
15         img = cv.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
16         roi_gray = gray[y:y+h, x:x+w]
17         roi_color = img[y:y+h, x:x+w]
18         eyes = eye_cascade.detectMultiScale(roi_gray)
19         for (ex,ey,ew,eh) in eyes:
20             cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),
21                         (0,255,0),2)
22         cv.imshow("InputGUI", img)
23         if not ret:
24             break
25         k = cv.waitKey(1)
26         if k%256 == 27: # ESC pressed, closing the window
27             break
28         elif k%256 == 32: # SPACE pressed, save grayscale face as
29             input
30             gray = gray[y:y + h, x:x + w] # Cut the frame to size
31             try:
32                 cv.imwrite("InputGUI {}.png".format(img_counter),
33                             cv.resize(gray, (350, 350)))
34                 img_counter += 1
35             except:
36                 pass

```

```
33 cam.release()  
34 cv.destroyAllWindows()
```

Listing 7.4: Dataset sortieren und überarbeiten

```
1 import glob, os, shutil, sys
2
3 if not os.path.exists("source_emotion"):
4     print("source_emotion wurde nicht gefunden!")
5     sys.exit(0)
6 if not os.path.exists("source_images"):
7     print("source_images wurde nicht gefunden!")
8     sys.exit(0)
9 if os.path.exists("sorted_set"):
10    shutil.rmtree("sorted_set")
11
12 # setup destination directory
13 emotions = ["neutral", "anger", "contempt", "disgust", "fear", "happy", "sadness", "surprise"]
14 os.makedirs("sorted_set")
15 for emotion in emotions: # create folder for each emotion
16     os.makedirs("sorted_set/%s" % emotion)
17
18 participants = sorted(glob.glob("source_emotion/*")) # Returns a list of all folders with participant numbers
19 for participant in participants:
20     part = "%s" % participant[-4:] # store current
21         participant number
22     sessions = sorted(glob.glob("%s/*" % participant))
23     participant_neutral_handled = False
24     for session in sessions: # Store list of sessions for
25         current participant
26         files = sorted(glob.glob("%s/*" % session))
27         for file in files:
28             current_session = file[20:-30]
29             file = open(file, 'r')
30             # store emotion as int
31             emotion = int(float(file.readline()))
32             # get belonging emotion sequence
33             images = sorted(glob.glob("source_images/%s/%s/*"
34             % (part, current_session)))
35             # copy last image from sequence to sorted set
36             sourcefile_emotion = images[-1]
```

```
34     destination_emotional = "sorted_set/%s/%s" % (
35         emotions[emotion], sourcefile_emotion[25:])
36     shutil.copyfile(sourcefile_emotion,
37                     destination_emotional)
38     # handle the neutral image only if not done for
39     # this participant before
40     if not participant_neutral_handled:
41         sourcefile_neutral = images[0]
42         destination_neutral = "sorted_set/neutral/%s"
43         % sourcefile_neutral[25:]
44         shutil.copyfile(sourcefile_neutral,
45                         destination_neutral)
46         participant_neutral_handled = True
47
48 print("sorted_set with emotion images created!")
```

Listing 7.5: Trainingsdaten als Arrays extrahieren

```
def get_training_sets():
    training_data = []
    training_labels = []
    for emotion in emotions:
        emotion_number = emotions.index(emotion)
        training = glob.glob("dataset/%s/*" % emotion)
        for item in training:
            training_data.append(cv.imread(item))
            training_labels.append(emotion_number)
    return training_data, training_labels
```

Listing 7.6: Skript zum Trainieren und Testen eines FisherFaceRecognizer

```
1 import glob, random, time
2 import numpy as np
3 import cv2 as cv
4
5 emotions = ["neutral", "anger", "contempt", "disgust", "fear", "
6 happy", "sadness", "surprise"]
7 face_cascade = cv.CascadeClassifier('haarcascades/
8     haarcascade_frontalface_default.xml')
9
10 def get_training_sets():
11     training_data = []
12     training_labels = []
13     for emotion in emotions:
14         emotion_number = emotions.index(emotion)
15         training = glob.glob("dataset/%s/*" % emotion)
16         for item in training:
17             training_data.append(cv.imread(item))
18             training_labels.append(emotion_number)
19     return training_data, training_labels
20
21
22 def create_fishface():
23     return cv.FisherFaceRecognizer.create()
24
25
26 def train_classifier(fishface, data, labels):
27     fishface.train(data, np.asarray(labels))
28     return fishface
29
30
31 def test_classifier(fishface):
32     cam = cv.VideoCapture(0)
33     cv.namedWindow("Classifier Test")
34     while True:
35         ret, img = cam.read()
36         gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
37         faces = face_cascade.detectMultiScale(gray, 1.3, 5)
38         for (x, y, w, h) in faces:
39             img = cv.rectangle(img, (x, y), (x + w, y + h),
40                               (255, 0, 0), 2)
41             gray = gray[y:y + h, x:x + w]
```

```
36     try:
37         gray = cv.resize(gray, (350, 350))
38         prediction, conf = fishface.predict(gray)
39         cv.putText(img, emotions[prediction], (x, h),
40                     cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 0, 0),
41                     lineType=cv.LINE_AA)
42     except:
43         pass
44     cv.imshow("Classifier Test", img)
45     if not ret:
46         break
47
48     k = cv.waitKey(1)
49     if k%256 == 27: # ESC pressed, closing the window
50         cam.release()
51         cv.destroyAllWindows()
52         break
53
54 def save_classifier():
55     fishface.write("second_fisher_face_classifier.xml")
56
57 training_data, training_labels = get_training_sets()
58 fishface = create_fishface()
59 train_classifier(fishface, training_data, training_labels)
60 save_classifier()
61 test_classifier(fishface)
```

Listing 7.7: Klassifizierer mit der Input GUI testen

```
def test_classifier(fishface):  
    cam = cv.VideoCapture(0)  
    cv.namedWindow("Classifier Test")  
    while True:  
        ret, img = cam.read()  
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)  
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)  
        for (x, y, w, h) in faces:  
            img = cv.rectangle(img, (x, y), (x + w, y + h),  
                               (255, 0, 0), 2)  
            gray = gray[y:y + h, x:x + w]  
            try:  
                gray = cv.resize(gray, (350, 350))  
                prediction, conf = fishface.predict(gray)  
                cv.putText(img, emotions[prediction], (x, h),  
                           cv.FONT_HERSHEY_SIMPLEX, 1.0, (255, 0, 0),  
                           lineType=cv.LINE_AA)  
            except:  
                pass  
        cv.imshow("Classifier Test", img)  
        if not ret:  
            break  
  
        k = cv.waitKey(1)  
        if k%256 == 27: # ESC pressed, closing the window  
            cam.release()  
            cv.destroyAllWindows()  
            break
```

Verzeichnisstrukturen

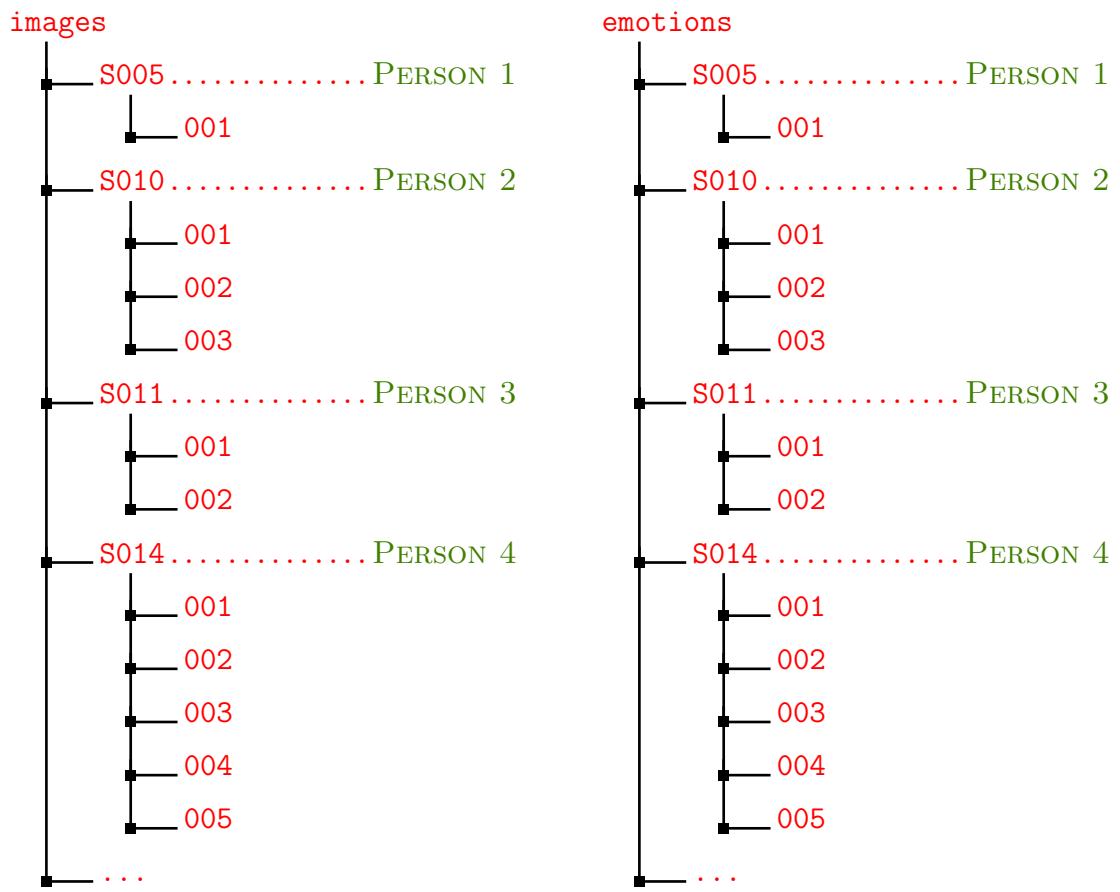


Abbildung G.B: Verzeichnisstrukturen des Cohn-Kanade Datasets

Sonstiges

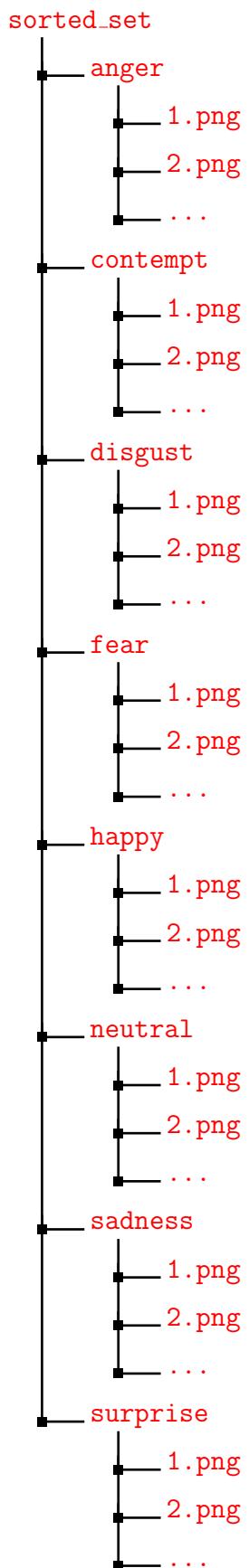


Abbildung G.C: Struktur überarbeitetes Dataset

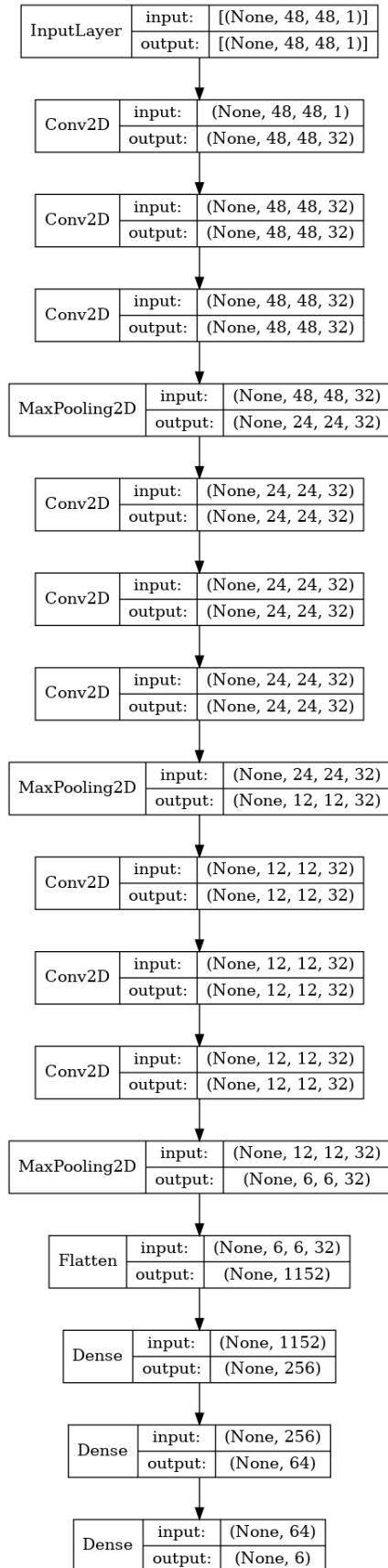


Abbildung G.D: Model Summary with several Layers and I/O Shapes