TECHNISCHE UNIVERSITÄT DRESDEN

PROFESSUR FÜR MENSCH-COMPUTER-INTERAKTION

Loomo-Emotionen 2: Verbesserung der Emotionserkennung mittels Machine-Learning

Martin Eisoldt, Thomas Hauptvogel

Dresden, 8. Mai 2019

Inhaltsverzeichnis

1 Aufgabenstellung **3**
1.1 Projektziel . . . 3
1.2 Schwerpunkte . . . 3

2 Ausgangssituation **3**

3 Related Work **6**
3.1 Emotionserkennung durch Bilder . . . 6
3.2 Emotionserkennung durch Sprache . . . 6

4 Ansätze zur Verbesserung der Emotionserkennung durch die Kamera **8**
4.1 Machine Learning Framework . . . 8
4.1.1 TensorFlow Lite . . . 8
4.1.2 Caffe2 . . . 8
4.2 Gesichtserkennung . . . 8
4.2.1 OpenCV . . . 9
4.2.2 Capture Result . . . 9
4.3 Datenbasis . . . 10
4.3.1 FER2013 . . . 10
4.3.2 (Extended) Cohn-Kanade . . . 10
4.3.3 Weitere Datensätze . . . 10
4.4 Bildvorverarbeitung . . . 11
4.5 Kamerabilder . . . 12
4.5.1 Lichtverhältnisse . . . 12
4.5.2 Perspektive . . . 13
4.6 Modellarchitektur . . . 13
4.7 Training . . . 14
4.8 Ergebnisse . . . 17
4.8.1 Gewähltes Vorgehen . . . 17
4.8.2 Erzielte Verbesserungen . . . 17
4.9 Fehlerursachen . . . 18

5 Emotionserkennung durch Sprache **19**
5.1 Grundlagen . . . 19
5.1.1 Akustische Größen und Einheiten . . . 19
5.1.2 Erfassung von Audiodaten . . . 19
5.2 Auswahl der Technologie . . . 20
5.2.1 Loomo SDK . . . 20
5.2.2 Native Libraries in Android . . . 20
5.2.3 TarsosDSP . . . 21
5.3 Extraktion der Audio Features . . . 22
5.4 Verarbeitung des Erregungslevels . . . 23
5.5 Weitere untersuchte Ansätze . . . 24
5.5.1 Gestik . . . 24
5.5.2 Audio Datenbanken . . . 25
5.6 Fehlerquellen . . . 25

6 Kombination von Bild und Sprache **26**

7 Datenschutz **28**

8 Benutzung **28**

9 Diskussion **29**

10 Ausblick **29**

Literatur **31**

1 Aufgabenstellung

1.1 Projektziel

Die Steuerung eines Roboters mittels Emotionen ist für die Entwicklung eines barrierefreien Roboters besonders erstrebenswert, da die Umsetzung einer Art "künstlichen Empathie" einen zusätzlichen Kommunikations-Layer ermöglicht, der insbesondere im Umgang mit Menschen mit Demenz eine wichtige Rolle spielt. Jedoch profitieren auch andere Nutzergruppen davon, wenn der Roboter angepasst auf die Stimmung des Nutzers mit dem Nutzer interagieren kann. Im Rahmen eines Komplexpraktikums wurde bereits ein Mechanismus implementiert, der das Machine-Learning-Framework Tensorflow von Google auf dem Loomo lauffähig macht, sodass eine Gesichtserkennung mit anschließender Emotionsabschätzung ausgeführt wird. Die Abschätzung der Emotionen funktioniert allerdings noch nicht zuverlässig. Diesem Problem soll sich dieses Forschungsprojekt widmen, sodass zum Ende des Projektes eine zumeist zuverlässige Erkennung sehr starker bzw. eindeutiger Basisemotionen möglich ist.

1.2 Schwerpunkte

Die Zielerreichung setzt eine umfassende Planung voraus. Das Szenario soll in einer einzelnen App bereitgestellt werden. Bestenfalls wird die bereits bestehende App zur Emotionserkennung weiterentwickelt.

- Arbeiten Sie sich in die bestehende Anwendung ein, sodass Sie ihre Funktionsweise verstehen
- Analysieren Sie mögliche Ursachen für die geringe Erkennungsgenauigkeit von Emotionen
- Erarbeiten Sie ein Konzept zur Verbesserung der Erkennungsgenauigkeit. Berücksichtigen Sie, dass es eine Vielzahl von unterschiedlichen Emotionen gibt jedoch die sechs Basisemotionen Angst/Furcht, Glück/Freude, Wut, Traurigkeit, Neugier/Überraschung und Ekel im Zentrum der Betrachtung stehen.
- Zur Verbesserung der Erkennungsgenauigkeit kann man sich einerseits der Verbesserung des bisherigen Verfahrens widmen und dabei Fehlerursachen wie Datenbasis, Perspektive, Lichtverhältnisse, etc. behandeln. Dies ist eine Entwicklungsweg, dem sich das Projekt widmen soll.
- Ein weiterer Weg zur Verbesserung der Erkennungsgenauigkeit ist das Berücksichtigen weiterer Informationen, wie bspw. akustischer Signale. Welche Informationen kann man hier berücksichtigen und ebenfalls in das bestehende Verfahren mit integrieren? Dies soll ebenfalls innerhalb des Forschungsprojekts behandelt werden.
- Implementierung der Anwendung: Erkennen von Emotionen und Ausgabe der erkannten Emotion auf dem Display (wie bisherige App)
- Dokumentieren Sie Ihre Arbeit präzise und auf wissenschaftliche Weise
- Beachten Sie bei der Bearbeitung, dass die Arbeitsanteile aller Forschungsprojekt-Teilnehmer gut voneinander abgegrenzt werden können, sodass eine individuelle Bewertung möglich ist, wie bei einem Forschungsprojekt gefordert.

2 Ausgangssituation

Die bisher bestehende App basiert auf Android 5 (API 21). Die Gesichtserkennung erfolgt dabei über die in Android enthaltene Funktion CaptureResult.STATISTICS_FACES [Anda]. Die dadurch erkannten Gesichter werden anschließend an TensorFlow Mobile übergeben. Auf Basis des auf dem Fer2013 [kag] trainierten Modells erfolgt dann eine Auswertung. Der zugrundeliegende Datensatz wurde dabei ohne weitere Vorverarbeitung genutzt. Das Ergebnis mit der höchsten Wahrscheinlichkeit wird dann an das

Frontend übergeben und dort in Form eines Smileys auf dem erkannten Gesicht dargestellt. Außerdem wird die Wahrscheinlichkeit der erkannten Emotion dargestellt. Insgesamt erzielt die App dabei Erkennungsraten von unter 30%.
Der Aufbau der App wird dabei durch das Klassendiagramm in Abbildung 1 dargestellt.

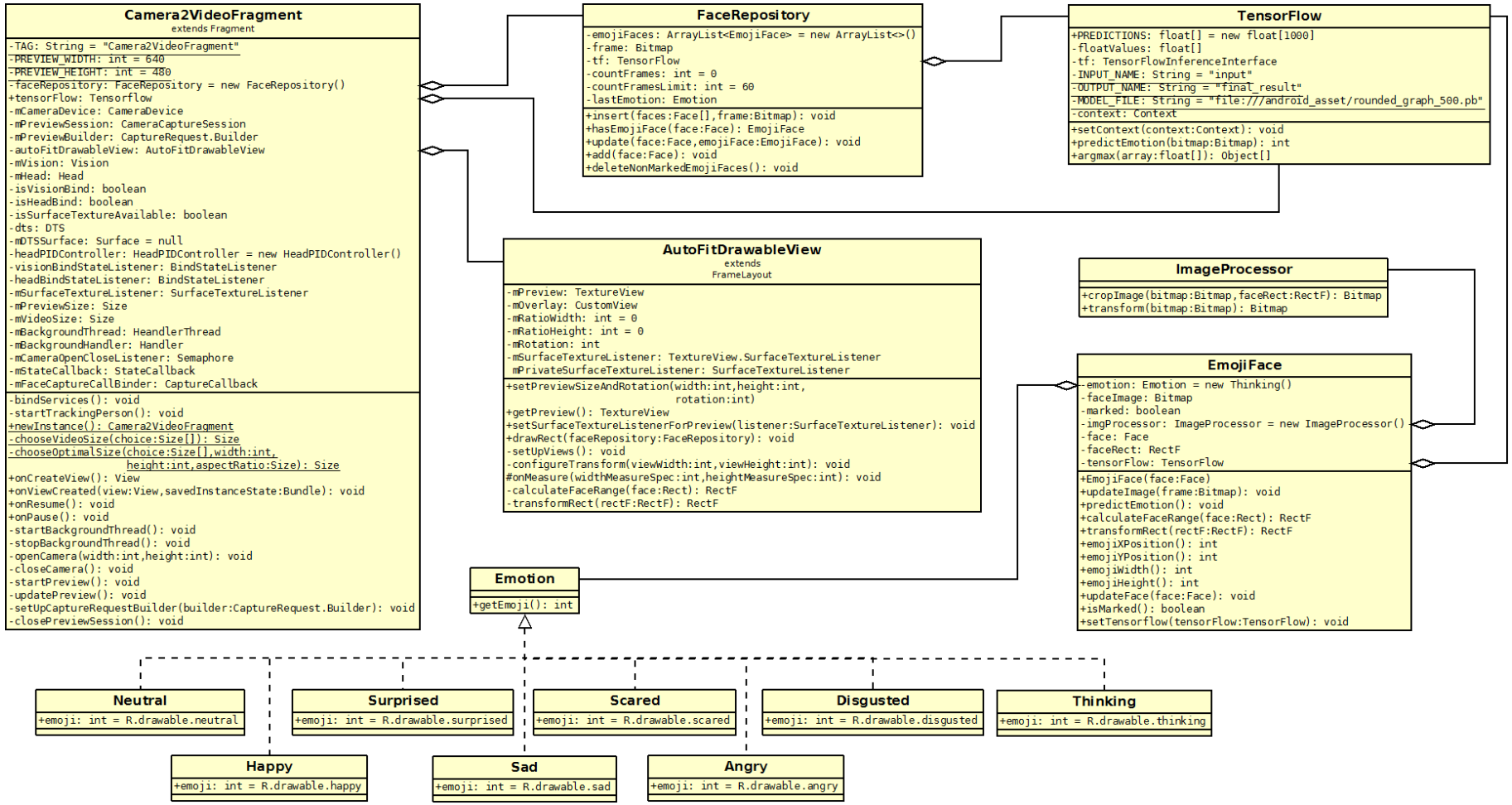

Abbildung 1: Klassendiagramm, entstanden im KP Loomo Emotion Sommersemester 2018

3 Related Work

Im folgenden Abschnitt werden verwandte Arbeiten erläutert, welche nützliche Ergebnisse für dieses Projekt lieferten oder ähnliche Ansätze verfolgten. Dabei werden diese Arbeiten in zwei Kategorien eingeteilt, je nachdem ob sie sich mit der Emotionsbestimmung über die Gesichtserkennung oder über die Sprache und Stimme beschäftigt haben.

3.1 Emotionserkennung durch Bilder

Martin Eisoldt

Orientierungspunkt für das Ausgangsprojekt dieser Arbeit war [CJOS16]. Darin wird mittels TensorFlow ein System darauf trainiert, die Emotionen von Menschen in Gesichtern zu erkennen. Dieses basiert dabei auf Python und läuft auf einem Desktoprechner. Dabei wird allerdings auch festgestellt, dass die Erkennungsgenauigkeit bei verschiedenen Emotionen stark variieren kann.
In [YTK+18] wird sich genauer mit den Möglichkeiten der Bildvorverarbeitung in OpenCV auseinandergesetzt. Im Mittelpunkt stehen die Optionen der Ausrichtung eines Gesichts anhand der Augen sowie der Zuschnitt auf das erkannte Gesicht. Allerdings findet dabei keine Untersuchung darauf statt, ob auf diese Art Ergebnisse des maschinellen Lernens verbessert werden.
Die Arbeit [Spi16] behandelt ebenfalls eine Emotionserkennung auf TensorFlow Basis. Dabei werden verschiedene Machine Learning Frameworks miteinander verglichen. Die Entscheidung für TensorFlow wurde jedoch nicht anhand von quantitativen Daten getroffen, sondern auf Basis der Menge an Nutzern und der hohen Wahrscheinlichkeit, dass dieses Framework lange weiterentwickelt wird.
Eine weitere Arbeit, die den Fokus auf Emotionserkennung mit neuronalen Netzen legt, ist [AVTP17]. Außerdem wird hierbei die Geschlechtserkennung betrachtet. Dieses Projekt wurde dabei auch mit einem Roboter, dem Care-O-bot 3, im Rahmen de RoboCup@Home eingesetzt. Ergebnis der Arbeit ist dabei, dass eine fast 100 %-ig genaue Bestimmung des Geschlechts anhand des Gesichts möglich ist. Jedoch ist bei der Emotionsbestimmung mit starken Ungenauigkeiten zu rechnen. In diesem Bereich konnte nur noch eine Erfolgsquote von 66 % gemessen werden.

3.2 Emotionserkennung durch Sprache

Thomas Hauptvogel

Da am Anfang der Arbeit noch kein Ansatz für die Emotionserkennung auf dem Loomo per Stimme existierte, wurden verschiedene Arbeiten zur Emotionserkennung mittels Tonaufnahmen durchsucht. Als erste Arbeit wurde ein Projekt von Zhang et al. untersucht, welches Emotionen über Pulsrate und emotionale Schlüsselworte erkennen wollte [ZML+13]. Die Emotionen wurden zuerst über den Puls klassifiziert und anschließend um zugehörige Wörter erweitert. Es wurde eine Genauigkeit von bis zu 84,7% erreicht, allerdings wurde der Versuch nur mit chinesischen Wörtern durchgeführt. Die Nutzung von emotional behafteten Wörtern ist ein vielversprechendes Verfahren, sodass dieser Ansatz auch auf dem Loomo untersucht wurde.
Ein Versuch bei dem probiert wurde Gesichtserkennung und Stimme zur Bestimmung von Emotionen zu kombinieren, ist von Busso et al. durchgeführt worden[BDY+04]. Dabei wurde festgestellt, dass die alleinige Erkennung über das Gesicht bessere Ergebnisse liefert, als wenn nur die Stimme genutzt wird. Die beste Robustheit und Leistung wurde allerdings mit einer Kombination aus Stimme und Bild erzielt. Ebenso wurde herausgefunden, dass einige Emotionen leicht verwechselt werden können, nämlich Trauer und Neutralität sowie Freude und Wut. Außerdem wurde in der Arbeit von Busso et al. auch die Tonhöhe und die Dauer der Aufnahmen analysiert, was auch für dieses Projekt gemacht wurde.
Eine Mischung aus Worterkennung und Stimmfeatures zur Bestimmung von Emotionen wurde von Lee und Narayanan untersucht[LN05]. Die Worterkennung wurde hier auch mit englischen Wörtern und

nicht nur mit chinesischen Wörtern wie bei Zhang untersucht[ZML+13], was für den europäischen Kulturraum besser geeignet ist. Als akustische Features wurden Frequenz, Energie, Dauer und Formanten[1] untersucht, welche bis auf die Formanten auch in unserem Projekt untersucht wurden. Die extrahierten Audiofeatures wurden in verschiedene Machine Learning Algorithmen eingespeist, sodass anschließend Emotionen bestimmt werden konnten.
In der Arbeit von Breazeal ging es nicht direkt darum menschliche Emotionen zu erkennen, sondern um den autonomen humanoiden Roboter Kismet, welcher mit Menschen in sozialen Austausch treten sollte [Bre05]. Da Kismet selber emotional betont antworten sollte, wurde auch untersucht, welche akustischen Eigenschaften die Stimme bei gewählten Emotionen, wie beispielsweise Furcht oder Freude hat. Die Analyse diente hier aber zur Ausgabe der Stimme und nicht der Analyse zur Erkennenung von Emotionen.

[1] Konzentration akustischer Energie in einem fixen Frequenzbereich

4 Ansätze zur Verbesserung der Emotionserkennung durch die Kamera

Martin Eisoldt

Um die vorhandene App so zu verbessern, dass die Gesichter mit einer besseren Genauigkeit erkannt werden, gibt es verschiedene Ansatzpunkte. Diese werden im Folgenden beleuchtet und auch die Ergebnisse der Umsetzung analysiert.

4.1 Machine Learning Framework

Die erste nötige Anpassung stellt das Machine Learning Framework da. Der wichtigste Grund hierbei ist, dass das verwendete Framework TensorFlow Mobile seit 2019 nicht mehr unterstützt wird [Gooa]. Die zugehörige Website ist seit März 2019 nicht mehr aufrufbar.
Durch die Verwendung eines anderen Machine Learning Frameworks kann auf verbesserte Genauigkeiten bei zeitgleich weniger Zeit- und Ressourcenbedarf gehofft werden.
Dabei sind TensorFlow Lite und Caffe2 in den Fokus der Aufmerksamkeit gerückt. Dies ist dem Umstand geschuldet, dass die Mehrzahl der Machine Learning Frameworks nicht auf mobile Endgeräte ausgelegt sind. Da diese Geräte nur über stark limitierte Ressourcen verfügen, kann dann ein optimaler Einsatz nicht mehr garantiert werden.

4.1.1 TensorFlow Lite

Erster Kandidat für die Umstellung war der von Google angebotene Nachfolger von TensorFlow Mobile, genannt TensorFlow Lite [Goob]. In der Anwendung ist der wesentliche Unterschied die Erstellung des Modells. Dabei wird weiterhin zuerst mit Python ein Modell aus den vorhandenen Daten erstellt. Dieser Schritt wird durch ein durch Google angebotenes Skript vereinfacht [Cod]. Nach der Erstellung des TensorFlow Modells wird dieses in ein .lite Modell umgewandelt. Diese Modellart zeichnet sich dadurch aus, dass weniger Speicherplatz benötigt wird und durch die geringere Größe auch eine höhere Geschwindigkeit erzielt wird.
In selber durchgeführten Tests hat sich gezeigt, dass die Leistungsfähigkeit größer ist als bisher. Mit TensorFlow Mobile wurde nur jedes 60. Kameraframe durch den Machine Learning Algorithmus ausgewertet. Mit TensorFlow Lite kann aber problemlos jedes 30. Frame ausgewertet werden. Über einen möglichen Genauigkeitsgewinn, der alleine auf das Machine Learning Framework zurückführbar ist, kann keine Aussage getroffen werden. Dies liegt daran, dass zeitgleich auch die Datensätze vor der Modellerstellung bearbeitet wurden (Vorgehen in Abschnitt 4.4 beschrieben).

4.1.2 Caffe2

Bei Caffe2 handelt es sich um ein Machine Learning Framework von Facebook. Dieses ist nach Unternehmensangaben 1.2 bis 5 mal schneller als Tensorflow [Bha]. Allerdings sind bereits bei der Einbindung des Frameworks in die App Probleme aufgetreten. Diese sind auf Kollisionen der Android Versionen zurückzuführen, welche wir nicht beheben konnten. Wir vermuten, dass dies dem geschuldet ist, dass auf dem Loomo nur Android 5.1 läuft, Caffe2 aber erst seit 2016 [YJ] existiert und somit nicht auf diese verhältnismäßig alte Android Version ausgelegt ist.

4.2 Gesichtserkennung

Die vermutlich am besten funktionierende Lösung für die Gesichtserkennung wäre sicher der FaceDetector aus dem Vision Package der Google PlayServices. Da aber auf dem Loomo keine PlayServices installiert sind, kann diese Erkennung nicht verwendet werden [Andb].
Somit ergeben sich noch zwei Möglichkeiten für die Gesichtserkennung. Dies sind sowohl die Umstellung auf OpenCV als auch die Weiterverwendung der existierenden Erkennung.

4.2.1 OpenCV

Die Gesichtserkennung mit OpenCV verspricht auf den ersten Blick die besten Ergebnisse. Aus diesem Grund wurde eine App getestet, die auf OpenCV basiert. Dabei handelt es sich um eine auf Android angepasste Version von OpenCV. Die Funktionalität ist somit gegenüber der originalen Python Variante teilweise limitiert, aber auf die geringeren Ressourcen der mobilen Endgeräte angepasst.
Für die Gesichtserkennung kann dabei auf verschiedene Modelle zurückgegriffen werden. Standardmäßig stehen Haarcascade und LBPcascade Dateien zur Verfügung. Dabei handelt es sich um Modelle, die bereits fertig durch andere Nutzer trainiert wurden und den Ansprüchen für OpenCV genügen. Dabei sind die Haarcascade Modelle genauer, benötigen aber auch eine längere Zeit für die Erkennung. Außerdem ist es auch möglich, eigene Modelle zu trainieren. Davon wurde aber aufgrund der zu hohen Komplexität (z.B. Sammeln ausreichender Bilddaten in angemessener Qualität) im Rahmen dieses Projekts abgesehen.
Die zugrunde liegende Technik, die unabhängig von den Modelldateien verwendet wird, ist hierbei die Viola-Jones-Methode [VJ01]. Dabei werden die Bilddateien in rechteckige Felder unterteilt, die auf Hell-Dunkel Unterschiede untersucht werden. Dafür werden Pixel in weiß markierten Bereichen addiert und in schwarzen subtrahiert. So ergibt sich für die gesuchten Objekte ein hoher Wert und für alle anderen ein niedriger. Wird ein Schwellwert überschritten, wurde das Objekt gefunden. Diese Rechtecke können an alle möglichen Positionen des Bildes verschoben werden. Eine beispielhafte Anwendung dieser Methode wird in Abbildung 2 dargestellt. Diese Methode zeichnet sich besonders durch Robustheit und Schnelligkeit aus. Allerdings müssen die Gesichter frontal zu sehen sein und auch gut beleuchtet sein [VJ01].
Die Hoffnungen an eine Verbesserung der Gesichtserkennung wurden auch dadurch erhöht, dass die Vor-

Abbildung 2: feature selection an Beispielbild [VJ01]

verarbeitung der Daten mit OpenCV geschieht (siehe Abschnitt 4.4). Damit werden die Bildausschnitte der Datenbasis und Livebilder auf die selbe Art erstellt.
Allerdings hat sich im direkten Vergleich gezeigt, dass die Ergebnisse auf diesem Weg nur minimal verbessert wurden. Problematisch war an dieser Stelle, dass Gesichter nicht konstant erkannt wurden, auch wenn sich weder das Gesicht noch der Roboter bewegt haben. Dies würde darin resultieren, dass bei jeder Neuerkennung eine erneute Berechnung der Emotion durchgeführt wird. Dadurch kommt es wiederum zu einer hohen Auslastung der Rechenkapazitäten. Somit stellt diese Umsetzung keine Alternative für die bisherige Umsetzung dar.

4.2.2 Capture Result

Die Dokumentation durch Google für diese Funktion ist sehr mangelhaft. Es wird nur erläutert, wie die Funktion anzuwenden ist, aber nicht wie sie funktioniert [Anda]. Somit können keine Aussagen zu den verwendeten Modellen für die Gesichtserkennung getroffen werden. Genauso wenig ist über eventuell automatisch stattfindende Bildvorverarbeitungsschritten verfügbar. Es kann lediglich angenommen werden, dass die Modelle ausreichend genau sind, da diese Funktion offiziell durch Android ausgeliefert wird und auch erst seit SDK 21 genutzt werden kann.

4.3 Datenbasis

Der Datenbasis kommt eine besondere Bedeutung zu. Mit der Qualität dieser steigt auch die Zuverlässigkeit der Emotionserkennung. Außerdem ist auch eine Vielzahl an Bildern von unterschiedlichen Personen nötig. Umso vielfältiger die Daten sind, desto besser ist auch die Auswertung der echten Bilder. Dabei bedeutet Vielfältigkeit sowohl möglichst viele verschiedene Personen als auch verschieden Bildhintergründe und Bleuchtungsstärken beziehungsweise -winkel.

(a) FER2013 [kag]

(b) Cohn-Kanade [LCK+10]

Abbildung 3: Beispielbilder aus den Datensätzen ohne Vorverarbeitung

4.3.1 FER2013

Der FER2013 Datensatz besteht aus 28.709 Bildern [kag]. Diese stellen die Emotionen Wut, Angst, Ekel, Freude, Traurigkeit, Überraschung sowie einen neutralen Gesichtsausdruck dar. Die Bilder wurden im Rahmen des „Challenges in Representation Learning: Facial Expression Recognition Challenge“ Wettbewerbs [kag] durch automatische Google Bildersuche mit Suchbegriffen, die in den Rahmen der Emotionen passen, erhoben. Die Bilder weisen dabei eine Auflösung von 48x48 Pixeln auf. Die genutzten Farben beschränken sich auf die Graustufen. Da es sich nicht um im Labor gefertigte Bilder handelt, kann von einer hohen Authentizität ausgegangen werden. Auch sind zum Teil verschiedene Beleuchtungshelligkeiten erfasst.

4.3.2 (Extended) Cohn-Kanade

Der Cohn-Kanade CK (beziehungsweise der extended Cohn-Kanade CK+) Datensatz besitzt 593 Bildsequenzen von 123 Personen. Bei einer Sequenz handelt es sich dabei um eine Reihe von Bildern, die die Entwicklung des Gesichts von einer neutralen Pose hin zu der Emotion zeigen. Die Anzahl der Bilder variiert dabei, so dass es am Ende zu 10.727 Bildern kommt. Diese wurden zum Großteil im Labor gefertigt und stellen die Emotionen Wut, Freude, Überraschung, Traurigkeit, Angst und Ekel sowie Neutralität dar. Für jede Person gibt es eine Bildreihe, die die verschiedenen Stufen zum Erreichen des Gesichtsausdrucks darstellt. Die Auflösung der Bilder ist hierbei 640x490 Pixel [LCK+10].

4.3.3 Weitere Datensätze

Datensätze wie zum Beispiel der Japanese Female Facial Expressions (JAFFE) Datensatz haben sich nicht als praktikabel erwiesen, da die Datengrundlage für dieses Projekt nicht passend sind. Ein Training mit Gesichtsausdrücken von Personen aus dem asiatischen Kulturraum lässt bei Anwendung in Europa häufiger fehlerhafte Bewertungen erwarten. Auch ist eine Beschränkung ausschließlich auf Frauen nicht zielführend [MLG97].

Anderer Datensätze wiederum konnten aus Copyright Gründen und damit einhergehender Zahlungsverpflichtung nicht verwendet werden. Dazu gehört unter anderem F-M FACS 3.0 (EDU, PRO& XYZ versions) [FM18].

4.4 Bildvorverarbeitung

(a) FER2013 [kag]

(b) FER2013 [kag] false positives

(c) Cohn-Kanade [LCK$^+$10]

Abbildung 4: Beispielbilder aus den Datensätzen nach Vorverarbeitung

Um optimale Trainingsergebnisse erreichen zu können, war es nötig, die Bilder auf die Teile zu beschneiden, in denen Gesichter dargestellt sind. Bilder wie Abbildung 3b besitzen zu viele Informationen, die nicht für die Emotionserkennung nötig sind. So würden auch der Hintergrund und die Kleidung der Personen mit in das Training einfließen. Aus diesem Grund ist es nötig, solche nicht benötigten Informationen zu entfernen.
Sowohl im FER2013 als auch im CK+ Datensatz liegen die Bilder als CSV Dateien vor. Diese müssen zuerst in JPG Dateien umgewandelt werden, da nur dieses Dateiformat als Input für TensorFlow genutzt werden kann. Zeitgleich werden die Bilder entsprechend der Emotionen in Ordner abgelegt. Ein Auszug aus dem Skript für die Gesichtserkennung ist in Abbildung 5 dargestellt. Dabei werden die Daten zuerst ausgelesen und danach auf eine Größe von 350x350 Pixel skaliert. Im Anschluss daran findet eine Umwandlung in ein Bild in Graustufen statt. Dieses wird danach noch normalisiert, so dass die Helligkeitswerte alle Bilder vergleichbar sind und nicht fehlerhafter Weise als Information über die einzelnen Emotionen betrachtet werden. Auf dieses normalisierte Bild wird nun die Gesichtserkennung angewendet. Dies geschieht mit dem durch OpenCV bereitgestellten Haarcascade Klassifizierer. Für die weitere Verwendung werden die Bildausschnitte auf 224x224 Pixel vergrößert, um sie dann bestmöglich mit TensorFlow verarbeiten zu können. Dabei wurden im FER2013 Datensatz noch 10.454 Bilder abgespeichert. Die Verteilung auf die verschiedenen Emotionen ist jedoch nicht gleichmäßig. „Freude “ umfasst 2.925 Bilder, „Wut“ 1.523 und „Ekel“ nur 155.
Der CK+ Datensatz umfasst nach den selben Vorverarbeitungsschritten noch 630 Bilder. Dabei weist „neutral“ 324 Bilder auf, „Fröhlich“ 69 und „Trauer“ 28. Hierbei ist auch noch zu beachten, dass dieser Datensatz für jede Testperson die Entwicklung des Gesichtsausdrucks von „neutral “ zur entsprechenden Emotion darstellt. Von Interesse für dieses Projekt ist dabei nur der finale Gesichtsausdruck sowie der neutrale zu Beginn der Bilderreihe. Eine entsprechende separate Verarbeitung mittels eines Skripts musste vorgenommen werden. Außerdem gibt es mehrere Bilder je Person für einen neutralen Gesichtsausdruck. Diese müssen von Hand entsprechend sortiert werden, so dass es für jede Person nur noch ein Bild mit neutralem Gesicht gibt. Dies erklärt die Abweichung der Anzahl der Bilder zwischen dem Ausgangsdatensatz und dem Ergebnisdatensatz.

Ergebnisse dieser Vorverarbeitung sind in Abbildung 4a beziehungsweise 4c zu sehen.
Teilweise ist es auch zu False Positives gekommen, die danach noch per Hand aussortiert werden mussten. Beispielhafte Bilder sind in Abbildung 4b.
Um die Vergleichbarkeit der Bilder weiter zu erhöhen, bietet es sich an, die Gesichter auszurichten. Dies wurde beispielsweise in [YTK+18] vorgeschlagen. Allerdings auf einer anderen Datenbasis. Ausrichtung nach den Gesichtern bedeutet dabei, dass die Bilder soweit gedreht werden, bis die Augen parallel zur Horizontalen sind. Jedoch haben sich dabei mit dem genutzten FER Datensatz die False Positives zu stark bemerkbar gemacht. So wurden häufig auch Nasen für Augen gehalten. Somit hat das Ausrichten keinen merkbaren Erfolg im Training erzielt.

```
def detect_faces(emotion):
    # get a list of all images with their emotions
    files = glob.glob("sorted_set/%s/*" %emotion)
    filenumber = 0
    for f in files:
        frame = cv2.imread(f) # Open image
        frame = cv2.resize(frame, (350, 350))
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        normalizedImage = np.zeros((350, 350)) # init the normalized face
        normalizedImage = cv2.normalize(src=gray, dst=normalizedImage,
                                        alpha=0, beta=255,
                                        norm_type=cv2.NORM_MINMAX,
                                        dtype=cv2.CV_8U)
        face = faceDet.detectMultiScale(normalizedImage, scaleFactor=1.1,
                                        minNeighbors=10, minSize=(5,5),
                                        flags= cv2.CASCADE_SCALE_IMAGE)
```

Abbildung 5: Ausschnitt aus dem Skript zur Extraktion der Gesichter aus dem Datenset

4.5 Kamerabilder

Die durch die Kamera aufgenommenen Bilder spielen eine wesentliche Rolle in der Erkennung der Emotionen. Ihre Qualität entscheidet maßgeblich darüber, wie gut die Auswertung ist. Aus diesem Grund wird im Folgenden untersucht, wie auf variierende Lichtverhältnisse reagiert werden kann und Verzerrung, die durch die Perspektive entsteht.

4.5.1 Lichtverhältnisse

Um die durch Loomo erhaltenen Bilder denen anzugleichen, auf denen trainiert wurde, werden die Histogramme ausgeglichen. Dies geschieht nur für die Bildbereiche, in denen auch ein Gesicht entdeckt wurde. Dieser Ausschnitt wird dann jeweils durch OpenCV zuerst in ein Bild in Graustufen umgewandelt und anschließend erfolgt der Histogrammausgleich. Somit weisen die Bilder trotz verschiedener Beleuchtungsintensitäten ähnliche Kontraste auf. Dadurch werden Emotionen nicht auf Basis der Helligkeit, sondern anhand der tatsächlich vorhandenen Bildinhalte gefällt.
Jedoch kann keine Veränderung der Lichtverhältnisse (z.B. durch den Histogrammausgleich mit OpenCV) vor der Gesichtserkennung durchgeführt werden. Dies wird durch die Android Klasse CaptureResult nicht unterstützt. Somit ist die einzige mögliche Variante, die Helligkeitswerte durch die Kameraeigenschaften zu beeinflussen. Dies ist jedoch nicht zielführend. Wenn die Beleuchtung aufgrund hoher

Lichtintensität im Hintergrund reduziert wird, dann hat dies zu Folge, dass das Gesicht entsprechend dunkler wird und somit nicht mehr sinnvoll weiterverwendet werden kann.

4.5.2 Perspektive

Eine weitere mögliche Quelle für schlechtere Ergebnisse ist die Perspektive. In den meisten Fällen schaut der Loomo von unten nach oben, um ein Gesicht zu erkennen. Dadurch entsteht eine gewisse Verzerrung der Gesichter.
An dieser Stelle bietet auch OpenCV wieder eine Lösungshilfe an. Allerdings werden für die Perspektivtransformationen feste Referenzpunkte benötigt. Diese können aber nicht geliefert werden, da sich die erfassten Szenen immer wieder verändern. Eine rein Winkel basierte Transformation wird durch keines der gängigen Tools unterstützt [Ope].
Es kann aber vermutet werden, dass durch die Machine Learning Algorithmen die Perspektive in gewissen Maß toleriert wird. So funktionieren gängige Algorithmen zur Gesichtserkennung auch mit leicht verschobenen Winkeln. Diese Vermutung konnte in eigenen Tests mit dem Loomo bestätigt werden. Beispielsweise konnte auch bei einem nur seitlich sichtbaren Gesicht eine Emotionserkennung erfolgreich durchgeführt werden. Die Trainingsdaten wiederum bestehen aber nur aus frontalen Gesichtsbildern.

4.6 Modellarchitektur

Die Modellarchitektur stellt einen wesentlichen Faktor in der Effektivität des Machine Learning Algorithmus dar. Dabei handelt es sich um die Architektur des neuronalen Netzes. Dabei sind in diesem Anwendungsfall die Genauigkeit (möglichst hoch) und die Latenz (möglichst gering) von Interesse.
Hierfür haben sich drei verschiedene Architekturen als betrachtenswert erwiesen. Dies sind Inception v3, 0.50 MobileNet-224 sowie 1.0 MobileNet-224. Diese drei Modelle haben sich aus verschiedenen Gründen in das Zentrum des Interesses geschoben. Inception v3 zeichnet sich durch eine hohe Genauigkeit aus. 0.50 MobileNet-224 wiederum hat eine sehr geringe Latenz mit guter Genauigkeit, während sich 1.0 MobileNet-224 als guter Kompromiss zwischen den beiden anderen Architekturen anbietet. Genauere Daten können der Tabelle 3 entnommen werden.
Die MobileNet Architektur gibt dabei zwei verschiedene Parameter, die zu unterschiedlichen Ergebnissen führen. Dies ist zum einen der Multiplikator für die Breite der Schichten. Dabei sind die Werte 1, 0.75, 0.5 und 0.25 möglich. Umso höher der Wert ist, desto mehr Parameter tauchen in dem Modell auf und somit steigt auch die Anzahl der Operationen. Dies hat bedeutet einen höheren Zeitaufwand, allerdings führt dieser auch zu höheren Genauigkeiten.
Der andere Parameter der Modell ist der Auflösungsmultiplikator. Dieser erlaubt die Kontrolle der Auflösung der Bilder, die in das Modell eingegeben werden. Dabei sind die Auflösungen 224, 192, 160 und 128 möglich (die Bilder müssen immer quadratisch sein). Dabei bleibt die Anzahl der Parameter immer gleich, jedoch steigt mit höherer Auflösung auch die Anzahl der benötigten Rechenoperationen. Eine höhere Auflösung führt hier zu verbesserter Genauigkeit, weshalb ausschließlich Modelle mit einer Eingabauflösung von 224x224 Pixel betrachtet wurden. Dies hatte sich bereits im vorangegangen Komplexpraktikum Loomo-Emotionen als bessere Variante erwiesen [HZC+17].
Gegenüber anderen Modellen zeichnet sich die MobileNet Architektur durch eine geringere Komplexität bei zeitgleich hoher Genauigkeit aus. Besonders ist außerdem, dass diese Architektur für die Anwendung auf mobilen Endgeräten entwickelt wurde. Aus diesem Grund wird diese Modellart durch Google auch die Verwendung mit TensorFlow Lite empfohlen.
Bei der Inception Architektur handelt es sich um GoogleNet. Dabei sind nicht mehr alle Eingangspunkte eine Verbindung zu allen Eingängen haben. Das liegt einfach an der Tatsache, dass viele der Aktivierungen entweder null sind oder aber redundant sind, da sie über andere Beziehungen erreichbar sind. Somit wird die Modellgröße reduziert. Außerdem werden die Werte nicht in einen vollständig verbundene Ausgangsschicht gegeben, sondern diese durch eine Schicht mit Durchschnittswerten dargestellt. Dieser Filter reduziert die Anzahl der Parameter um einen weiteren Teil, ohne die Genauigkeit bedeutend zu beeinflussen [kou].

Breitenmultiplikator	Genauigkeit	Rechenoperationen (in mio)	Paramter (in mio)
1.0 MobileNet-224	70.6 %	569	4.2
0.75 MobileNet-224	68.4 %	325	2.6
0.5 MobileNet-224	63.7 %	149	1.3
0.25 MobileNet-224	50.6 %	41	0.5

Tabelle 1: MobileNet Breitenmultiplikator nach [HZC+17]

Breitenmultiplikator	Genauigkeit	Rechenoperationen (in mio)	Paramter (in mio)
1.0 MobileNet-224	70.6 %	569	4.2
1.0 MobileNet-192	69.1 %	418	4.2
1.0 MobileNet-160	67.2 %	290	4.2
1.0 MobileNet-128	64.4 %	186	4.2

Tabelle 2: MobileNet Auflösung nach [HZC+17]

Auf Basis dieser Vorbetrachtungen wurden sowohl das Inception v3 Modell als auch das 1.0 MobileNet-224 getestet. Allerdings war die Verarbeitungszeit mit dem Inception Modell zu lang, als das es sich für einen weiteren Einsatz empfohlen hat. Somit sind alle zukünftigen Schritte auf dem 1.0 MobileNet-224 geschehen.

4.7 Training

Das Training der verschiedenen Modelle findet auf einem stationären Computer statt. Dies liegt in erster Linie daran, dass durch Tensorflow kein Training auf mobilen Endgeräten unterstützt wird. Somit kann aber auch die höhere Rechenleistung genutzt werden und mit zusätzlichen Konfigurationen auch auf Grafikkarten trainiert werden. Es ist auch nicht lohnenswert, dass Modell bei jedem Aufruf der Anwendung live zu erstellen, da das Training mehrere Stunden umfassen kann. Außerdem ändert sich der Ausgangsdatensatz nicht, womit eine einmalige Erstellung der Modelle ausreichend ist.
Um das Training zu verbessern, können verschiedene Parameter angepasst werden. Allerdings ist hier auch immer die Gefahr des Overfitting zu berücksichtigen. Dies bedeutet, dass das Modell zu sehr auf die Trainingsdaten zugeschnitten ist. Somit kann stellt das Modell die Trainingsdaten sehr gut dar, erzielt aber sehr schlechte Ergebnisse auf den reellen Daten. Für das Training werden die Daten in drei verschiedene Mengen unterteilt. Dies ist die größte Datenmenge mit den Daten, auf denen das Training ausgeführt wird. Diese Menge ist die Größte und aus ihr werden nach jedem Schritt zufällig Daten ausgewählt, mit denen das Training validiert wird. Außerdem gibt es die Menge an Validierungsdaten, die vor dem Training aus der Gesamtmenge der Daten entnommen wird. Diese wird nur zum validieren der Trainingsdaten eingesetzt, aber nie für das Training. Somit kann auf diese Art beurteilt werden, wie gut das Modell auf neue Daten passt. Standardmäßig werden 10 % der Bilder für die Validierung verwendet Zu diesen Parametern gehört die Anzahl der Traininigsschritte. Dies bedeutet, wie oft die Daten durchlaufen werden um neue Informationen zu erhalten. Diese Einstellung hat einen wesentlichen Einfluss auf den Erfolg des Trainings. Gerade mit den ersten 5.000 bis 10.000 Schritten werden große Erkenntnisgewinne erzielt. Dies ist beispielsweise in der Abbildung 6 zu sehen. Zu jedem Zeitpunkt ist allerdings

	Inception v3	0.50 MobileNet-224	1.0 MobileNet-224
Latenz (mit TF Lite)	1433 ms	47.6 ms	160.1 ms
Top-1 Genauigkeit	77,9 %	63.2 %	71,0 %
Top-5 Genauigkeit	93,8 %	84.9 %	89,9 %
Modellgröße	95,3 Mb	5.3 Mb	16.9 Mb

Tabelle 3: Vergleich der verschiedenen Modelle auf Basis von [Ten]

ID	Modellparameter		Ergebnisse		
	Lernrate	Anzahl Trainingsschritte	Testgenauigkeit	Validierungsgenauigkeit	Overfitting?
1	0.01	50.000	51,4 %	45,0 %	nein
2	0.01	100.000	50,6 %	47,0 %	nein
3	0.01	200.000	50,3 %	49,0 %	nein
4	0.001	50.000	54,9 %	51,7 %	ja
5	0.0001	50.000	53,7 %	52,0 %	nein
6	0.0001	100.000	53,7 %	52,5 %	Tendenz vorhanden

Tabelle 4: Vergleich der verschiedenen Trainingsparameter

zu beachten, dass ein Anstieg der Genauigkeit für die Trainingsdaten (rot) bei zeitgleicher Stagnation oder Abnahme der Genauigkeit der Validierungsdaten (grün) gleichbedeutend mit Overfitting ist. Zu den Abbildungen ist anzumerken, dass die Graphen einen Mittelwert darstellen, die erzielten Werte sind im Hintergrund ausgegraut zu sehen.
Ein weiterer Parameter der betrachtet wurde, ist die Lernrate. Diese bedeutet, wie stark die Gewichtungen bei jedem Durchlauf angepasst werden. Eine große Lernrate bedeutet eine starke Anpassung der Gewichtungen, aber zeitgleich kann es dadurch passieren, dass das Optimum nicht erreicht wird. Eine kleine Rate wiederum führt dazu, dass sehr viele Durchläufe benötigt werden, um die optimalen Modellparameter zu erreichen.
Außerdem erlaubt Tensorflow das Einfügen von Verzerrungsfaktoren in die Bilder. Dabei werden die Bilder rotiert, nur Bildausschnitte verwendet und die Helligkeit verändert. Dies soll dazu führen, das mögliche Ungenauigkeiten wie sie in der Realität vorliegen können, auch ins Training einfließen. Diese Verzerrungsoperationen haben sich als sehr rechenintesiv herausgestellt und konnten mit der vorhandenen Infrastruktur nicht umgesetzt werden.
Aufgrund mangelnder Informationen über optimale Trainingsparamter im Zusammenhang mit dem verwendeten Datensatz (FER 2013, siehe Kapitel 4.3.1), mussten diese durch Tests ermittelt werden. Ziel ist dabei, zu maximieren, während die Entropie minimiert werden soll. Das bedeutet, die Ergebnisse sollen genau werden und die Unsicherheit minimal sein.
In Abbildung 6 ist das Training mit den Parametern der Einstellung 4 zu sehen. Dabei sind zwei Unterdiagramme dargestellt. Das obere steht für die Genauigkeit, das untere für die Entropie. Der orange Graph repräsentiert die Trainingsergebnisse, der blaue die Validierungsergebnisse. Es ist ersichtlich, dass die Validierungsgenauigkeit bei ca. 12.000 Schritten ihr Maximum erreicht und dann immer wieder schwankt, währen die Trainingsgenauigkeit weiterhin zunimmt. Dies ist ein erster Indikator für Overfitting. Da ab circa 28.000 Trainingsschritten die Entropie der Validierungsmenge zunimmt, kann hier auf jeden Fall von Overfitting ausgegangen werden. Deswegen können die Parameter dieser Einstellung nicht für das Training des Modells verwendet werden, das am Ende auf dem Loomo zum Einsatz kommt. Die Graphen wurden TensorBoard entnommen, einem Analysetool, das Bestandteil von Tensorflow ist.

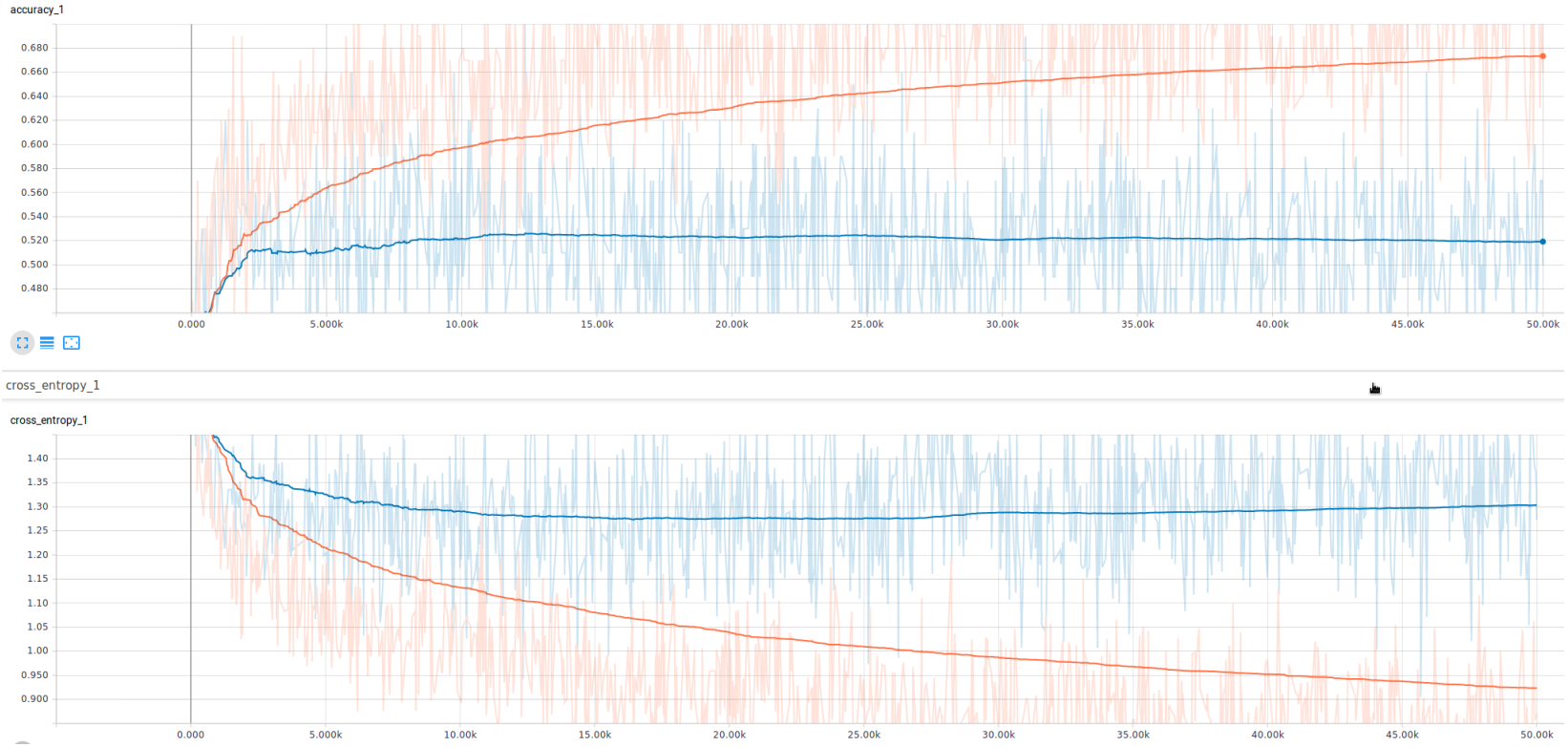

Abbildung 6: Trainingsgraph zu Einstellung 4

Abbildung 7: Zuordnungen zur Emotion *Wut*

In Tabelle 4 werden die verschiedenen Einstellungen der Trainingsparameter sowie deren Auswirkung auf die Genauigkeit untersucht. Dabei hat sich die Standardeinstellung der Lernrate mit 0.01 nicht als die die beste Variante herausgestellt. Dies drückt sich in der abnehmenden Testegenauigkeit bei zunehmender Anzahl an Trainingsschritten aus. Die Entscheidung ist auf die Einstellung 5 gefallen. Diese weißt zwar eine minimal geringere Validierungsgenauigkeit als Variante 6 auf, dafür gibt es aber keine Tendenz zum Overfitting und auch die Trainingszeit ist geringer.
Es ist aber zu beachten, dass dies nur für den FER-2013 Datensatz die besten Parameter sind. Mit einem anderen Datensatz oder anderem Architekturmodell können diese anders sein.

4.8 Ergebnisse

4.8.1 Gewähltes Vorgehen

Aus den zuvor vorgestellten Verbesserungsansätzen wurden am Ende folgende gewählt:

- Machine Learning Framework: TensorFlow Lite
- Gesichtserkennung: Android CaptureResult.STATISTICS_FACES
- Datenbasis: FER2013 inklusive Datenvorverarbeitung (Gesichtserkennung, zurechtschneiden)
- Kamerabilder: Histogrammausgleich bei erkannten Gesichtern durch OpenCV
- Modellarchitektur: 1.0 MobileNet-224
- Training mit folgenden Parametern: Lernrate 0,0001; 100.000 Traininigsschritte

4.8.2 Erzielte Verbesserungen

In anderen Arbeiten wurde bereits mit diesen beiden Datensätzen gearbeitet. So hat Quinn in [QSR17] festgestellt, dass die menschliche Erkennungsrate der korrekten Emotion für den fer2013 Datensatz bei 65% +/- 5% liegt. In ihrer Arbeit haben die Forscher eine Erkennungsrate von 71 % Prozent erreichen können. Aufgrund der Tatsache, dass es sich bei Tensorflow Lite um ein auf mobile Endgeräte spezialisiertes Framework handelt, konnten diese Werte als das maximal Erreichbare betrachtet werden. Somit sind die erzielten Genauigkeiten von 52 % ein gutes Ergebnis. Da die Erkennungsrate über 50 % liegt, werden die Emotionen besser zugeordnet, als wenn geraten würde.
Zu beachten ist an dieser Stelle auch, dass sich die Genauigkeitswerte nur auf das trainierte Modell beziehen. Die Nutzererfahrung auf dem Loomo kann davon durchaus abweichen. In eigenen Tests konnten wir aber bestätigen, dass Modelle, die nach dem Training besser bewertet wurden, auch in der Realität zu besseren Ergebnissen geführt haben.
Außerdem kann jedes 30. Frame ausgewertet werden, statt nur jedem 60. in der Ausgangsversion der App. Somit kann schneller auf Veränderungen reagiert werden. Außerdem erlaubt die Verwendung von *TensorFlow Lite* einen einfachen Austausch des Modellformats. Somit können bei Weiterentwicklungen der Modellarchitekturen diese einfach eingesetzt werden.

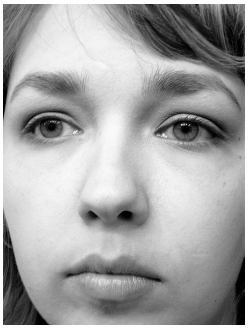

Abbildung 8: neutrales Gesicht von [Spo]

4.9 Fehlerursachen

Mögliche Fehlerursachen ergeben sich aus der Festlegung der Emotionskategorien. Diese stellen die Basisemotionen dar, welche aber nicht die menschlichen Emotionen in ihrer gesamten Komplexität widerspiegeln. Zum Beispiel wird in [DTM14] davon gesprochen, dass es Emotionen wie „wütende Überraschung “ und „wütenden Ekel “ gibt. Diese in das vorliegende Schema einzuordnen, fällt allerdings schwer. So kann es zu falschen Zuordnungen kommen, da das System entweder mehr Wut oder mehr Überraschung erkennt.
Teilweise ist auch die Klassifizierung der Emotionen im Trainingsdatensatz diskussionswürdig. Beispielbilder dafür sind in Abbildung 7 zu sehen. Allerdings stellt die manuelle Zurodnung aufgrund der Größe des Datensatzes keine Alternative dar.
Auch haben in Tests mit dem Loomo die Lichtverhältnisse trotz des Histogrammausgleichs einen Einfluss auf das Ergebnis gehabt. So wurde bei entsprechendem Schattenwurf häufiger „Wut “ erkannt, als wirklich dargestellt wurde. In der Gegenprobe mit einem perfekt ausgeleuchteten Bild der Googlebildersuche (siehe Abbildung 8 wurde hingegen immer die korrekte Emotion erkannt und auch mit der immer gleichen Wahrscheinlichkeit. Dies zeigt, dass die App korrekt arbeitet.

5 Emotionserkennung durch Sprache

Thomas Hauptvogel

Dieser Teil soll sich den Ansätzen zur Erkennung der Emotionen durch Spracheingaben und dazugehörigen Grundlagen und Problemen widmen. Bisher wurde auf dem Loomo nur mit vordefinierten Sprachbefehlen gearbeitet, sodass eine geeignete Aufnahmemöglichkeit für rohe Stimmdaten erst einmal gefunden werden musste. Dafür wurden verschiedene technische Möglichkeiten getestet. Es mussten passende Features der menschlichen Stimme gefunden werden, welche mit Hilfe des Loomos extrahiert werden konnten. Am Ende des Kapitels wird noch erläutert wie Bild und Ton miteinander kombiniert wurden um die Emotionen der aufgenommenen Personen zu bestimmen.

5.1 Grundlagen

Um zu verstehen, wie man Audiodaten erfassen und verarbeiten kann, ist ein wenig Hintergrundwissen über Akustik nötig, welches in diesem Abschnitt erläutert werden soll. Außerdem soll gezeigt werden, welche Möglichkeiten es zur Erfassung von Audiodaten gibt.

5.1.1 Akustische Größen und Einheiten

Zur Messung der Stärke eines Schallereignisses, z.B. der Spracheingabe in ein Mikrofon, wird der **Schalldruckpegel** verwendet. Der Schalldruckpegel, kurz auch SPL, ist ein relativer Wert, welcher das logarithmierte Verhältnis des Schalldrucks zum Quadrat eines Referenzwertes beschreibt. Der Referenzwert für Luftschall bezieht sich dabei auf die menschliche Hörschwelle und liegt bei circa $20\mu Pa$.
Der **Schalldruck** bezeichnet in diesem Fall die Druckschwankungen innerhalb der Luft, bei der Ausbreitung des Schalls. Schalldruck kann mit Mikrofonen gemessen werden, wobei der messbare Bereich des SPL ungefähr zwischen 0 und 160 dB liegt. Der Schalldruck und dessen Pegel sind technische Größen, d.h. die von Menschen wahrgenommene Lautstärke nicht proportional zu den Werten des SPL ist. Eine Erhöhung von 30 dB auf 40 dB könnte man als Mensch schon als Verdopplung der Lautstärke wahrnehmen, sodass man bei der Emotionsbestimmung auch schon geringe Wertänderungen beachten sollte.
Die Maßeinheit **Dezibel**(dB) wird hier als unterstützende Einheit benutzt um einfacher vorstellbare Werte zu erhalten. Je nach Entfernung beträgt der durchschnittliche Wert für Zimmerlautstärke zwischen 40 dB und 50 dB[HUG]. Die **Tonhöhe** ist die Frequenz hörbarer Töne. Der Mensch kann Frequenzen zwischen 20 und 20.000 Hertz hören, wobei in dieser Arbeit aber wichtiger ist, welche Frequenzen mit der menschlichen Stimme erzeugt werden können. Die Grundtonhöhe eines Mannes liegt bei circa 125 Hz, wohingegen der Wert bei einer weiblichen Stimme um die 250 Hz liegt.

5.1.2 Erfassung von Audiodaten

Es gibt zwei grundlegende Möglichkeiten um Informationen aus Audiodaten in Echtzeit zu erhalten und zwar die Aufnahme von rohen Audiodaten, also unkomprimierten Daten, oder über das Erkennen von Wörtern.
Für die Aufnahme von unkomprimierten Audiodaten wird etwas mehr Speicherplatz benötigt, aber es geht keine Rechenleistung für die Komprimierung verloren und es entsteht auch kein Datenverlust. Über einen Listener können aufgenommene Mikrofondaten einfach als Raw Data abgespeichert und direkt weiter verarbeitet werden.
Bei der Erkennung von Wörtern kann man als Beispiel die Alexa Voice Services von Amazon nennen. Der Service wird dabei durch ein sogenanntes *Wake-Word* aufgeweckt bzw. aktiviert. Danach gibt man üblicherweise einen Sprachbefehl an das System weiter, um beispielsweise das Wetter zu erfahren und falls bestimmte Wörter in dem Satz erkannt werden, so werden dazugehörige Aktionen ausgeführt. Wie in Abschnitt 3.2 beschrieben, wurden in anderen Arbeiten bestimmte Wörter zur Bestimmung von Emotionen genutzt, was nach dem gleichen Prinzip funktioniert.

5.2 Auswahl der Technologie

Im Laufe der Arbeit wurden der im Loomo SDK integrierte RawDataListener, ein mit Hilfe von nativen Android Bibliotheken selbst implementierter Listener und das Audio Framework *TarsosDSP* getestet. Dabei wurde beachtet, in welcher Qualität die Daten aufgenommen werden konnten, ob alles in Echtzeit lief und ob die Lösung robust und einigermaßen performant lief. Letztendlich wurde aufgrund der Performanz und Möglichkeiten der Aufnahme und Verarbeitung von Audio Daten das Framework *TarsosDSP* genommen.

5.2.1 Loomo SDK

In einigen Arbeiten wurden bestimmte emotional behaftete Schlüsselwörter verwendet um Emotionen zu bestimmen. Loomo hat in seinem SDK bereits Funktionen implementiert um Wörter zu erkennen. Leider muss Loomo durch ein Wake-Word geweckt werden, in diesem Fall beispielsweise *"Hey Loomo"*, um die Erkennung von Wörtern zu starten. Da die Emotionen ununterbrochen bestimmt werden sollen und Loomo nicht ständig erneut dafür aktiviert werden soll, wurde der Ansatz der Emotionserkennung über bestimmte Wörter hier erstmal verworfen.
Eine Möglichkeit um rohe Audiodaten über die Mikrofone des Loomo zu erhalten, ist der im Loomo SDK integrierte *RawDataListener*. Dieser nimmt die ganze Zeit neue Daten im *Byte* Format auf und anschließend können diese weiterverarbeitet werden. In Abb.9 ist ersichtlich, dass der vom Loomo SDK bereitgestellte Listener keine weitere Funktionalität bietet.
Im Loomo SDK gibt es die Funktion *Recognizer.startBeamFormingListen(RawDataListener mRawDatalistener)*, welche laut Dokumentation das Beamforming[2] für die Audioaufnahme aktiviert. Da die im Test aufgenommenen Daten des integrierten RawDataListener im Allgemeinen eher von schlechter Qualität waren, bemerkte man bei aktivierten Beamforming auch keine signifikanten Unterschiede in den Werten. Da keine weitere Funktionalität vom Loomo für die Aufnahme und Weiterverarbeitung von rohen Audiodaten gegeben waren und diese auch nur im Byte-Format zur Verfügung standen, wurde dieser Ansatz nicht weiter verfolgt.

```
mRawDataListener = new RawDataListener() {
    @Override
    public void onRawData(byte[] bytes, int i) {
        \\ Do something with the audio data ...
    }
};
\\ Starts Beamforming with the raw data
recognizer.startBeamFormingListen(mRawDataListener);
```

Abbildung 9: Code Beispiel RawDataListener

5.2.2 Native Libraries in Android

Eine weitere Möglichkeit zur Aufnahme von Audiodaten ist die Nutzung des Android NDK[3]. Damit können C/C++ Code und auch Plattformbibliotheken genutzt werden, um auf physische Gerätekomponenten zuzugreifen. Dafür muss das Android NDK installiert werden und die genutzten C/C++ Komponenten müssen beispielsweise mit *cross-platform make(CMake)* für Android Studio nutzbar gemacht werden. Dabei wird mit Hilfe eines Skriptes ein Makefile erstellt, welches von der Entwicklungsumgebung und vom Compiler genutzt werden kann. CMake ist zwar das Standardtool für Android Studio, aber es muss

[2] In diesem Kontext ist damit das Ausblenden von Hintergrundgeräuschen gemeint.
[3] Native Development Kit

auch erst installiert werden und zwar zusammen mit dem LLDB Debugger, um nativen Code debuggen zu können.
Durch diese Komponenten lässt sich ein eigener Listener für Audiodaten implementieren, welcher die Mikrofondaten in einem geeigneteren Datentyp, wie z.B. dem Double Typ, abspeichern kann. Android bietet schon eine *AudioRecord* Klasse und eine *MediaRecorder* Klasse an, welche den Zugriff auf das Mikrofon erlauben und eine Verarbeitung der aufgenommenen Daten ermöglichen. Für die Versuche mit dem Loomo wurden AudioRecord-Objekte mit einer Sample Rate von 44100 Hz und 16 Bit Sampletiefe erstellt. Es sind zwar mehrere Mikrofone vorhanden, aber die Aufnahmen wurden alle nur mit einem Channel (Mono) aufgezeichnet, da mehrere Kanäle für die Auswertung der Stimmdaten nicht notwendig sind. Die Aufnahme läuft permanent im Hintergrund, sodass Werte wie maximale Amplitude oder die durchschnittliche Lautstärke mit Hilfe von Observables beobachtet wurden. Ein Problem war allerdings, dass keine Möglichkeit zur Erfassung der Frequenz der aufgenommenen Stimmen vorhanden war, welche für die Erfassung des Arousal allerdings zwingend notwendig ist. Hier hätte man noch die ganzen Daten immer wieder filtern und anschließend eine schnelle Fourier Transformation durchführen müssen. So wäre eine Berechnung diskreter Frequenzwerte für die aufgenommenen Daten möglich gewesen, aber aufgrund des Implementierungs- und Rechenaufwandes wurde noch nach weiteren Möglichkeiten gesucht.
Google hat auch eine *SpeechRecognizer* Klasse, mit der Wörter erkannt werden können. Allerdings ist mit dieser Klasse für das Android-System auf dem Loomo kein Offline-Support gegeben. Loomo kann sich zwar mit dem Internet verbinden, aber ein stabiles und schnelles Netz ist gerade in Deutschland nicht überall erreichbar und die Verarbeitungszeit von erkannten Audiodaten würde somit noch weiter steigen.

5.2.3 TarsosDSP

TarsosDSP ist ein von Joren Six entwickeltes Java Framework, welches Audio Analyse und Verarbeitung in Echtzeit ermöglicht[SCL14]. Die Möglichkeit Features zur Laufzeit zu extrahieren und verarbeiten lässt dieses Framework im Java Umfeld positiv heraus stechen. Vor allem da es einfach als Android Library einzubinden ist.
Weiterverarbeitung von Audiodaten, um beispielsweise den Pitch zu verändern oder einen Delay Effekt einzufügen, sind zwar Möglichkeiten, welche das TarsosDSP Framework zur Verfügung stellt, aber im Kontext der Emotionserkennung nicht benötigt werden. Man kann auch Spektogramme erstellen, welche zur Analyse der Frequenzen über einen festgelegten Zeitraum genutzt werden können, aber da diese Funktionalität nicht in Echtzeit läuft ist sie nicht für diesen Anwendungsfall geeignet. Letztendlich wurden ein Silence und ein Pitch Detector genutzt, um die benötigten Features extrahieren zu können. Der Silence Detector ist ein Audio Processor mit einer definierten Schwelle für Stille. Alle Töne unterhalb der Schwelle werden nicht verarbeitet, sodass nicht jedes Hintergrundgeräusch vom Loomo verarbeitet wird. Zur Erkennung der Tonhöhe wird ein Pitchprocessor genutzt, welcher zur Erkennung der Tonhöhe eine schnellere Version des YIN-Algorithmus nutzt[CK02]. Der YIN-Algorithmus gehört zu den *Pitch Detection Algorithms(PDA)*, von denen bisher keine ideale Lösung existiert. Dieser hier basiert auf Autokorrelation und ist besser für die Bestimmung der Tonhöhe bei der menschlichen Stimme geeignet. Es werden noch mehr Algorithmen zur Nutzung bereitgestellt, welche zawr relativ ähnliche Ergebnisse liefern, aber eher für Lieder geeignet oder langsamer sind. Die über ein Mikrofon aufgenommenen Audiodaten werden direkt an einen sogenannten *AudioDispatcher* gesendet, in kleinere Blöcke zerlegt und dann zur Verarbeitung durch die Audioprozessoren geschickt. Die Audioprozessoren enthalten den kompletten Code zur Verarbeitung um beispielsweise die Tonhöhe zu berechnen. Für beliebige Audio Events, wie zum Beispiel das Erkennen einer Tonhöhe, kann über einen Handler dann eine Weiterverarbeitung der ausgegebenen Daten erfolgen (Abb. 10).
Aufgrund der bereitgestellten Möglichkeiten, der unkomplizierten Einbindung und vor allem der Echtzeitverarbeitung wurde dieses Framework als beste Option für dieses Projekt gewählt. Zwar bietet TarsosDSP keine Möglichkeit um gesprochene Sätze bzw. Wörter zu erkennen, aber diese Möglichkeit hat

sich auch bei den anderen Ansätzen nicht ergeben, sodass die Erkennung von emotionalen Schlüsselwörtern hier für dieses Projekt komplett verworfen wurde.

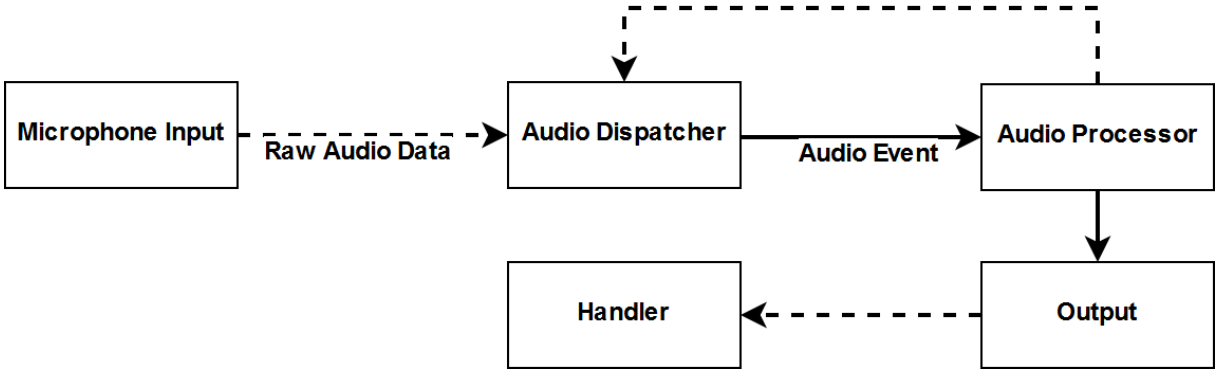

Abbildung 10: Vereinfachte Darstellung des TarsosDSP Verarbeitungsflow

5.3 Extraktion der Audio Features

Bei der Wahl der zu extrahierenden Features spielten sowohl die technischen Möglichkeiten, als auch die Bedingung der Echtzeitauswertung eine Rolle. Dabei wurde sich an den Erkenntnissen von Breazeal orientiert, welche in Abbildung 11 zu sehen sind.
Die Qualität der Stimme und die Art der Artikulation lassen sich leider in mit dem gewählten Framework in Echtzeit nicht erfassen. Dafür bräuchte es eine genaue Analyse im Nachhinein und eine technische Möglichkeit zur Worterkennung. Bei Breazeal wurden die Audiodaten auch nicht von Algorithmen sondern von Menschen ausgewertet, sodass eine Auswertung dieser beiden Features technisch eventuell gar nicht möglich ist. Somit haben sich für diese Untersuchung folgende Punkte ergeben:

- Sprechgeschwindigkeit
- Durchschnittliche Tonhöhe
- Streuung der Tonhöhe
- Intensität
- Wechsel der Tonhöhe

Da nicht direkt analysiert werden konnte, wie schnell eine Person einzelne Worte ausspricht, wurde für die Sprechgeschwindigkeit beobachtet, wie schnell die benötigte Anzahl an Audio Samples erreicht wird. Wenn Loomo in diesem Projekt Wörter erkennen könnte, dann wäre es auch möglich herauszufinden wie schnell hintereinander Wörter ausgesprochen werden. Im Endeffekt ist es auch nicht gewährleistet, dass die Erkennung von Wörtern bessere Ergebnisse liefern würde als das Erreichen einer gewissen Anzahl an Audio Samples. Für die Intensität beziehungsweise das Power Level wurde die Lautstärke beobachtet, wobei diese nur als Schalldruck in Pascall gegeben war und erstmal in Dezibel umgerechnet wurde.

$$Ratio_{db} = 20 \times \log_{10}(\frac{P_1}{P_0}) \tag{1}$$

Mit der Formel 1 kann der aktuelle Schalldruckpegel in die Einheit Dezibel umgerechnet werden. *TarsosDSP* gibt in der bereits implementierten Funktion für den Schalldruckpegel leider einen für uns unpassenden Wert zurück, da der Referenzwert für menschlich wahrnehmbaren Schalldruck dort nicht beachtet wird. Dieser beträgt $20\mu Pa$ und steht für Schwelle der menschlichen Geräuschwahrnehmung, also für uns noch wahrnehmbare absolute Stille. Der Wert ist wichtig, da es in diesem Projekt um die menschliche

The effect of emotions on the human voice

	fear	anger	sorrow	joy	disgust	surprise
speech rate	much faster	slightly faster	slightly slower	faster or slower	very much slower	much faster
pitch average	very much higher	very much higher	slightly lower	much higher	very much lower	much higher
pitch range	much wider	much wider	slightly narrower	much wider	slightly wider	
intensity	normal	higher	lower	higher	lower	higher
voice quality	irregular voicing	breathy chest tone	resonant	breathy blaring	grumbled chest tone	
pitch changes	normal	abrupt on stressed syllable	downward inflections	smooth upward inflections	wide downward terminal inflections	rising contour
articulation	precise	tense	slurring	normal	normal	

Abbildung 11: Auswirkungen von Emotionen auf ausgewählte Eigenschaften der menschlichen Stimme nach Breazeal[Bre05]

Stimme geht und nicht um Töne, welche von Menschen gar nicht erzeugt oder gehört werden können. Die Tonhöhe wird ununterbrochen erfasst und sobald die benötigten Samples aufgezeichnet wurden, werden daraus die Streuung der Werte und die durchschnittliche Tonhöhe berrechnet. Es werden dabei einfach alle aufgenommenen Tonhöhen aus den Samples miteinander verrechnet, da bei der Aufnahme in Echtzeit nicht zwischen verschiedenen Geschlechtern unterschieden werden kann.
Sobald die benötigten Samples erreicht und die aktuellen Werte zur Berechnung der Features genutzt wurden, wird daraus im nächsten Schritt das aktuelle Erregungslevel berechnet.

5.4 Verarbeitung des Erregungslevels

Wie man in Abbildung 11 erkennen kann, sind sich viele Emotionen in den hier gewählten Eigenschaften, also Tonhöhe, Sprechrate und Intensität ziemlich ähnlich. Da eine genaue Analyse der Aufnahmen zu viel Aufwand für die Ressourcen wäre und dies auch nicht zur Laufzeit passieren kann, orientierten wir uns hier an der Einteilung nach Russel [Rus80]. Dadurch erfolgte nur noch die Einteilung in Valenz und Erregung. Die Valenz, also positive oder negative Konnotation, lässt sich ohne weitere Datenanalyse über die Stimme schwer abgrenzen, sodass die Berechnung des Erregungslevels hier im Vordergrund stand. Je nachdem welche Eigenschaften die letzten Audiosamples aufzeigen wird das Erregungslevel, wie in Tabelle 5 dargestellt, ständig angepasst.

Neben der Erregung wurde noch der zugehörige Zeitstempel abgespeichert und anschließend wurden beide Werte in der Klasse *Speech* in einer HashMap hinterlegt. In Abbildung 12 sieht man, dass Tonhöhe und Lautstärke für zwanzig Werte vorhanden sein müssen, damit die Erregung berechnet und in *Speech* gesetzt wird. Dieser Vorgang wird permanent durchgeführt, auch wenn kein Gesicht erkannt wird.

	niedrige Sprechrate	hohe Sprechrate	hohe Pitch-Streuung	niedrige Pitch-Streuung
Erregung	sinkend	steigend	steigend	sinkend

	hoher ØPitch	niedriger ØPitch	hohe ØIntensität	niedrige ØIntensität
Erregung	steigend	sinkend	steigend	sinkend

Tabelle 5: Änderung des Erregunslevel je nach Eigenschaften der Aufnahme

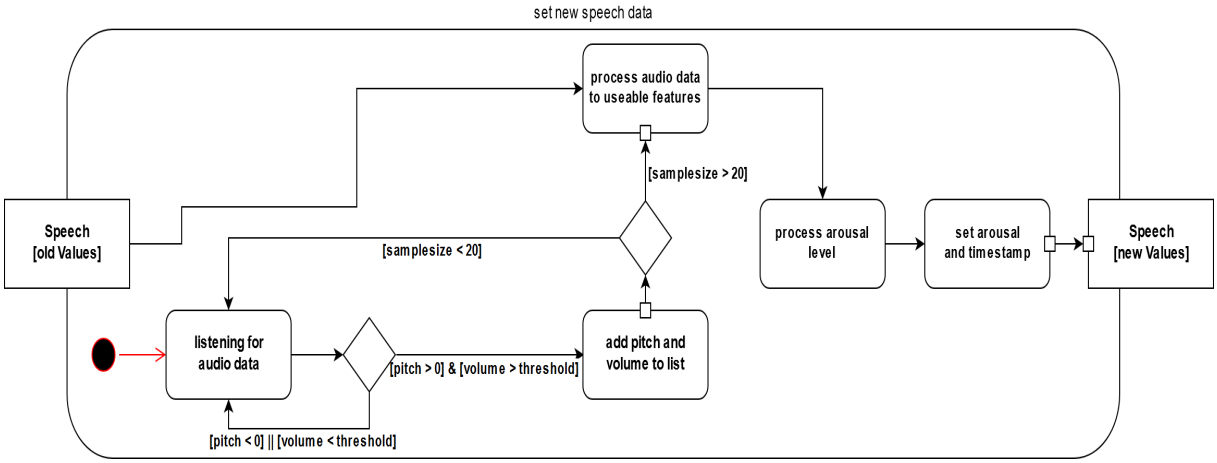

Abbildung 12: Setzen eines neuen Erregungslevel in der Speech-Klasse

5.5 Weitere untersuchte Ansätze

Ziel dieser Arbeit war es nicht nur Emotionen über Audioaufnahmen zu bestimmen, sondern auch zu untersuchen welche Möglichkeiten es sonst noch gibt um den Gefühlszustand eines Nutzers mit Hilfe des Loomo herauszufinden. Außerdem wurde noch geschaut, ob es schon vorhandene klassifizierte Audiodaten gibt, mit denen ein Machine Learning Algorithmus gefüttert werden könnte.

5.5.1 Gestik

Eine Möglichkeit zur Emotionsbestimmung die noch mit in Betracht gezogen wurde, waren körperliche Gesten. Gesten bieten ein ebenso großes Potenzial wie das Gesicht oder die Stimme, da Gesten interkulturell ein Symbol zur Expression von Emotionen sind. Aber auch bei den Gesten gibt es das Problem, dass sich mehr wissenschaftliche Arbeiten auf das Gesicht beschränken und die körperlichen Merkmale und Posen eher vernachlässigt werden. Noroozi et al. beschreiben in einem Survey, wie Emotionen über Gesten technisch erkannt werden können und welche kulturellen und biologischen Unterschiede es gibt[NCK+18]. Nicht nur der Gesichtsausdruck, sondern auch wie man seinen Kopf dreht und neigt sagt einiges über den Gemütszustand aus, beispielsweise kann man aus Furcht das Gesicht verdecken oder aus Ekel den Kopf wegdrehen. Allerdings könnte dann nicht mehr das Gesicht erkannt werden und man müsste zur Emotionsbestimmung trotzdem schauen, ob eine Person im Blickwinkel des Roboters vorhanden ist oder nicht.
Kulturelle Unterschiede gibt es natürlich auch, da ein nach oben gerichteter Daumen in einigen Kulturräumen Zustimmung bedeutet und wiederum woanders für eine Beleidigung steht. Diese unterschiedlichen Auffassungen der gleichen Geste haben im Laufe der Zeit durch die Globalisierung etwas abgenommen. Ebenso gibt es zwischen den Geschlechtern Unterschiede, da Männer mit ihrer Gestik eher Dominanz und Macht zeigen wollen, wohingegen Frauen bereitwilliger Emotionen zeigen, da sie diese mit Freundlichkeit und Zuneigung verbinden.
Mit Hilfe von Tiefenkameras und Modellen des menschlichen Körpers können Gesten und Posen erfasst werden und darüber könnte man die Emotion bestimmen. Der Loomo besitzt eine *Intel RealSense*

ZR300 3D Kamera, welche aber leider am nicht neigbaren Teil des Loomo befestigt ist, sodass Gesten nur aus bestimmter Entfernung richtig erkannt werden könnten. Außerdem besteht bei mehreren Personen das Problem, dass sich die erkannten Körper überlagern und erkannte Gesten eventuell falsch gedeutet werden. Selbst wenn Menschen Gesten und Emotionen einander zuordnen sollten, dann war die durchschnittliche richtige Erkennungsrate bei sechs Emotionen ungefähr 20 Prozent schlechter als wenn nur vier Emotionen zur Auswahl standen [PSO+14].
Aufgrund der mangelhaften Dokumentation des Loomo, den kulturellen und geschlechtsspezifischen Unterschieden und der hier im Projekt führenden Gesichtserkennung wurde der unterstützende Ansatz über Gesten hier erst einmal nicht mehr weiter verfolgt. Denn aus der Entfernung und Perspektive in der Loomo die Gesichter der Nutzer gut erkennen kann, werden viele Gesten höchstwahrscheinlich nicht richtig erkannt werden.

5.5.2 Audio Datenbanken

Eine weitere Idee war es, einfach nach Datensätzen zu suchen, welche bereits emotional behaftete Audio Dateien enthalten, um die Emotionen dann wie bei der Gesichtserkennung per Machine Learning zu bestimmen. Leider gibt es kaum frei verfügbare und verwertbare Datenbanken für diesen Zweck. Einzig und allein die ”Berlin Database of Emotional Speech“ bot sich dafür an, aber auch diese Datenbank hat Probleme, denn sie allein ist zu klein um ein repräsentatives Trainingsdatenset zu bilden. Die Datenbank enthält Aufnahmen von jeweils fünf Männern und Frauen, welche geskriptete Texte in sieben verschiedenen Emotionen(Ärger, Angst, Freude, Trauer, Ekel, Langeweile, neutral) vorgetragen haben[BPR+05]. Den Akteuren wurden kurze Anweisungen gegeben um die Aufnahme simpel zu halten, aber dadurch sind die Emotionen in der Stimme nicht wirklich natürlich. Bei der Auswertung mussten von 800 Beispielen um die 500 Stück rausgeworfen werden, da diese als zu unnatürlich eingestuft oder zu schlecht erkannt wurden. In Verbindung mit anderen Datensätzen ist diese Datenbank allerdings trotzdem sehr nützlich und wenigstens frei verfügbar. Zusätzlich hätte man aus allen Daten Spektrogramme bilden müssen, um die Audio Features auszuwerten, was in Echtzeit auf dem Loomo viel Rechenzeit gekostet hätte und damit wäre die Sprache als unterstützende Funktion eher der Gesichtserkennung zur Last gefallen.

5.6 Fehlerquellen

Obwohl die Erkennung der Stimme die meiste Zeit stabil und fehlerfrei läuft, kann es teilweise zu falschen Einschätzungen kommen. Wenn mehrere Personen in Hörweite des Loomo reden, so können Lautstärke und Frequenzen der verschiedenen Stimmen stark variieren und damit beispielsweise wütende Stimmen erkannt werden, aber traurige Personen werden in einer Gruppe dann nicht korrekt identifiziert. Wenn Personen viel zu leise reden, dann hat der Roboter auch manchmal Probleme die Stimme richtig zu erkennen. Allerdings hängt die Schwelle zur Erkennung hier einerseits vom Code und andererseits von den eingebauten Mikrofonen ab. Ein letztes Problem hängt noch mit TarsosDSP zusammen, da die Dispatcher manchmal das Mikrofon nicht korrekt freigeben und dieses dann von anderen Anwendungen nicht korrekt genutzt werden kann, wobei das Beenden der *Emotion App* das Mikrofon auch wieder komplett freigeben kann, falls dieser Fall eintritt.

6 Kombination von Bild und Sprache

Um die Ergebnisse aus der Bild und Sprachanalyse zu kombinieren, haben wir uns für eine Einordnung der Emotionen nach [Rus80] entschieden. Dieser ordnet diese nach Valenz(positiv/negativ) und Erregung(hoch/niedrig) ein (siehe Abbildung 15. Diese Einteilung ist dahingehend vorteilhaft, dass sie vier Dimensionen hat, welche verschiedene Emotionen enthalten. Die Eigenschaften von Ton und Bild überschneiden sich für einige Emotionen, sodass es schwierig ist diese korrekt zu unterscheiden. Vor allem die Tonaufnahmen bieten eher gute Möglichkeiten um das Erregungslevel herauszufinden und die Gesichtserkennung kann genutzt werden um zusätzlich noch die Valenz herauszufinden.
Die Bestimmung der Emotion orientiert sich hauptsächlich an der Gesichtserkennung, sodass auch Emotionen ausgegeben werden, wenn kein Ton erkannt wird. Allerdings trifft das nicht auf die umgekehrte Situation zu. Wenn Stimmen erkannt werden, dann wird weiterhin das Erregungslevel berechnet und sobald ein Gesicht erkannt wird, kann der aktuelle Erregungswert zur Bestimmung der Emotion genutzt werden. Dafür wurde die bestehende Lösung um die *Speech, SpeechEmotion* und *PictureEmotion* erweitert, wie in Abbildung 14 zu sehen ist. Die Klasse SpeechEmotion enthält ein Erregungslevel und einen Zeitstempel, wohingegen PictureEmotion eine Liste mit den Confidence-Werten der einzelnen Emotionen und auch einen zugehörigen Zeitstempel beinhaltet. Die beiden aktuellsten SpeechEmotion Objekte werden in der Klasse Speech in einer HashMap gespeichert. Da während der Berechnung der Emotion auch weiterhin das Erregungslevel aktualisiert wird, werden die Zeitstempel der Werte in Speech mit dem Zeitstempel der PictureEmotion verglichen. Das näherliegende Erregungslevel fließt dann in die Bestimmung der Emotion mit ein oder die Tonaufnahmen spielen gar keine Rolle mehr, falls die Zeitdifferenz mehr als drei Sekunden beträgt. Dieser ganze Prozess zur Bestimmung der aktuellen Emotion ist in Abbildung 13 modellhaft abgebildet.
Um die Ergebnisse noch weiter zu verbessern und ein „Springen “ der angezeigten Emotionen zwischen verschiedenen Emojis zu verhindern, werden zunächst die letzten 4 erkannten Emotionen zwischengespeichert. Aus diesen werden dann für jede Emotion jeweils die Durchschnittswerte gebildet und auf diese Werte die Gewichtungsfunktion angewendet. Wenn die Erregung null ist, findet keine Gewichtung statt und alle Ergebnisse werden so ausgegeben, wie sie anhand der Gesichter ermittelt wurden.
Sollte aber eine Erregung erkannt werden, so werden die Emotionen *Angst, Wut, Ekel, Überraschung und Freude* höher gewertet. Im Gegensatz dazu werden *neutral und Trauer* geringer bewertet. Ist die Erregung dagegen auf -1, erfolgt eine höhere Wertung für *Trauer*, allen anderen Werten wird dagegen ein geringeres Gewicht zugeordnet. Die Gewichtungsfunktion multipliziert dabei anhand der Sprach erkannte Emotionen mit 2 und halbiert nicht erkannte Emotionen.
Zusätzlich wird die aktuell erkannte Atmosphere auf dem Display ausgegeben. Diese bezieht sich auf die Emotion, die in den letzten Berechnungen am häufigsten erkannt wurde. Somit ist auch für den Nutzer nachvollziehbar, warum eine bestimmte Emotion angezeigt wird, auch wenn sich die Umstände möglicherweise bereits geändert haben.

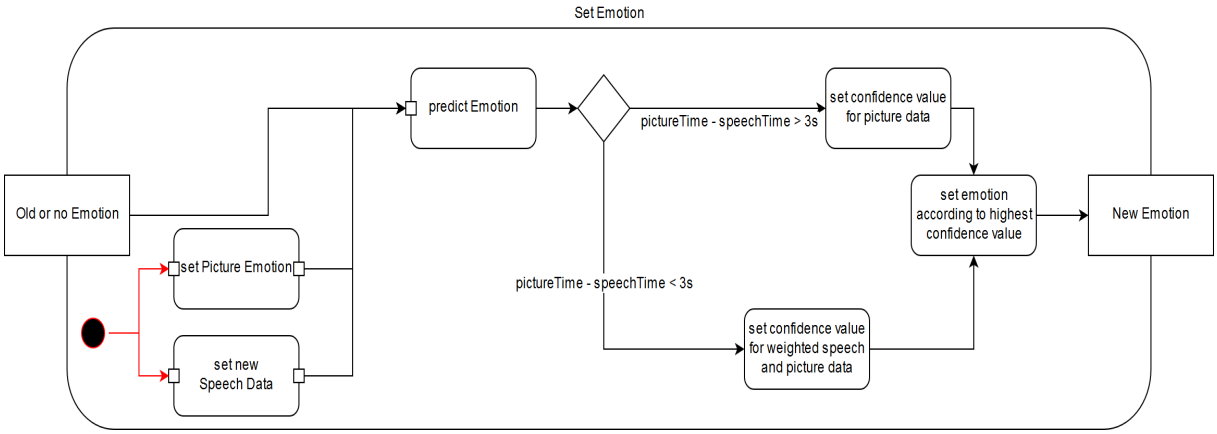

Abbildung 13: Prozess der Emotionsbestimmung

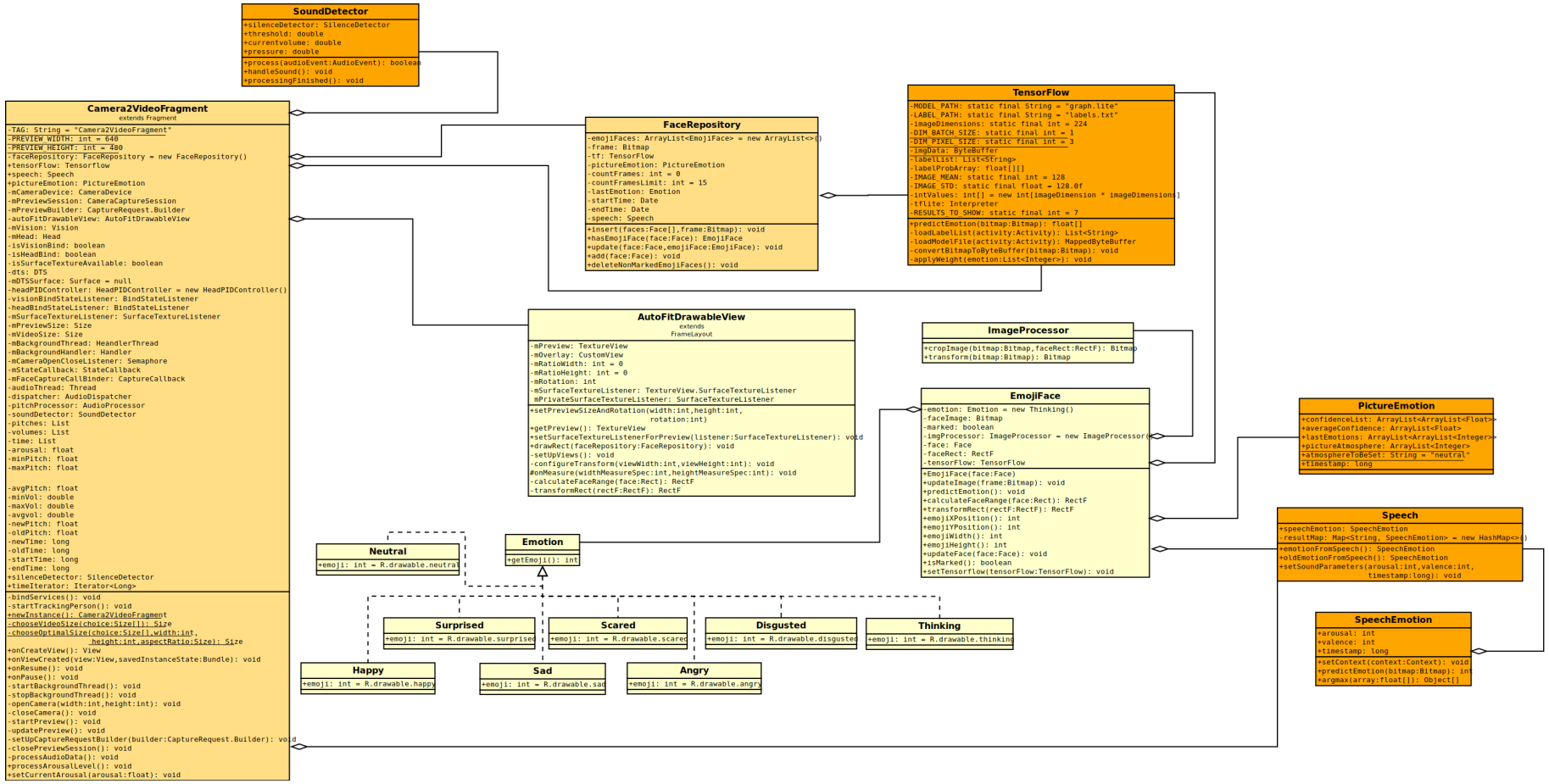

Abbildung 14: Klassendiagramm, erweitert und aktualisiert (dunkles orange -> neue Klasse, helles orange -> aktualisierte Klasse)

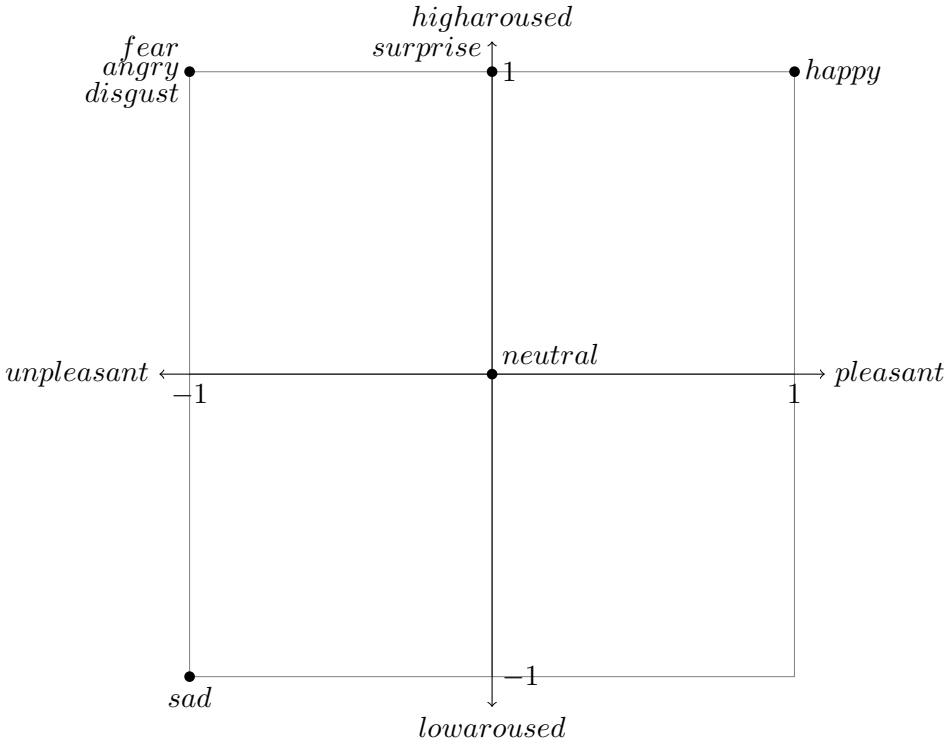

Abbildung 15: Einordnung der Emotionen nach Russel [Rus80]

7 Datenschutz

Da während der gesamten Ausführungszeit sowohl die Kamerabilder wie auch die Stimmen aus der Umgebung aufgenommen werden, stellen sich auch datenschutzrechtliche Bedenken. Diese können aber dadurch entkräftet werden, dass eine Speicherung von Bild und Ton nur lokal geschieht und zu keiner Zeit eine Zwischenspeicherung auf externen Datenträgern geschieht. Auch erfolgt die gesamte Datenverarbeitung nur lokal auf dem Loomo. Außerdem werden alle Daten lediglich im Arbeitsspeicher gehalten, somit sind sie spätestens mit Herunterfahren des Loomo nicht mehr verfügbar. Gesichts- und Tonaufnahmen werden auch nicht zur Identifizierung der aufgenommenen Personen genutzt, sondern nur zur Bestimmung der Emotion. Sollte ein Nutzer kurz den Aufnahmebereich verlassen, so weiß der Loomo nicht mehr, dass er diese Person schon mal aufgenommen hat.
Wenn die App gestoppt beziehungsweise pausiert wird, wird auch die Beobachtung der Umgebung mittels Kamera und Mikrofon gestoppt, da die Threads sonst im Hintergrund unbemerkt weiter aufzeichnen würden.

8 Benutzung

Für die Verwendung der App zur Weiterentwicklung wird folgendes benötigt:

- Android Entwicklungsumgebung (z.B. Android Studio)
- Android SDK 21 (Lollipop 5.1)
- OpenCV4Android 3.4.3
- TarsosDSP-Android
- tensorflow-lite

Für die Erstellung von neuen TensorFlow Lite Modellen wird die Verwendung von Python empfohlen. Dies bietet sich an, da Python seine Stärken gerade in dem Bereich des Machine Learning hat und es eine große Zahl an Hilfestellungen gibt. Dafür muss auch eine extra Installation von TensorFlow und OpenCV geschehen, damit diese auch in einer Desktopumgebung nutzbar sind.
Detailliertere Beschreibungen für die Benutzung sind unter anderem in der Dokumentation zum Komplexpraktikum Loomo-Emotionen zu finden.

9 Diskussion

Im Laufe der Arbeit wurden mehrere Datensätze zur Gesichtserkennung ausprobiert und auch die Machine Learning Frameworks *TensorFlow Lite* und *Caffe2* wurden miteinander verglichen. Für Tonaufnahmen wurden die Möglichkeiten des Loomo SDK und von nativen Android Bibliotheken ausgetestet, wobei am Ende dann das Audio-Framework *TarsosDSP* als bestmögliche Lösung gewählt wurde. Es gibt momentan noch Probleme wenn jemand zu leise spricht, da eine gewisse Schwelle an Lautstärke erreicht werden muss, ansonsten wird es nur als Stille interpretiert da keine zu leisen Audiodaten aufgenommen werden. Ebenso ist es schwer zu erkennen wie der emotionale Status der Stimme ist, wenn zu viele Leute in der Gegenwart des Loomo reden, da die im messbaren Bereich vorhandenen Stimmen alle aufgenommen und ausgewertet werden. Verschiedene Emotionen in Gruppen zu erkennen, war aber auch nicht das Ziel der Arbeit und da zusammengehörige Gruppen in den meisten Situationen ähnliche Emotionen zu empfinden, kann diese Eigenschaft der Aufnahme auch vorteilhaft sein. Die Valenz konnte leider über die Stimme nicht in Echtzeit bestimmt werden, da hier aufgrund der Performanz und Echtzeitauswertung nur die Erregung effizient bestimmt werden konnte. Da die Erkennung der Valenz über das Gesicht genauer ist und die Stimmaufnahmen hier nur unterstützend zur Emotionsbestimmung beitragen, führt dieser Abstrich allerdings nicht zu merkbar schlechteren Ergebnissen.
Schwierigkeiten bereitet auch noch die korrekte Gewichtung aller Emotionen gemäß der aus der Sprache erhaltenen Informationen. Dies liegt daran, dass keine Unterscheidung zwischen angenehmer und unangenehmer Atmosphäre stattfindet. Wenn diese Differenzierung noch zusätzlich mit in die Berechnung einfließt, können sicherlich noch bessere Ergebnisse erzielt werden.
Aktuell wird die Atmosphäre an der Emotion festgemacht, die durch die als erstes erkannte Person am häufigsten ausgedrückt wurde. Diese Person wird auf dem Display mit der niedrigsten ID dargestellt. Sicherlich kann dies zu Fehlern führen, man nehme nur folgende Situation als Beispiel: In einem Raum befinden sich 3 Personen. Davon machen 2 Witze auf Kosten der dritten Person. Somit sind Person A und B fröhlich, während Person C eher traurig ist. Wurde nun Person C als erstes erkannt, wird von einer traurigen Atmosphäre ausgegangen. Dabei ist aber die Mehrzahl der Personen fröhlich und, aus einer theoretischen Betrachtung, ist somit auch die Stimmung im Raum fröhlich.
Richtig perfektionieren lässt sich die Emotionserkennung auf dem Loomo wahrscheinlich nicht, da eine ausführliche Auswertung der Stimmen in Echtzeit, sowie eine genaue Gesichtserkennung für viele Personen aufgrund des Betriebssystems und der Hardware schwer umzusetzen sind. Allerdings muss der Loomo das auch nicht, da die Bestimmung der Atmosphäre in den meisten Fällen ausreicht um eventuell zukünftig auf die Situation reagieren zu können. Darüber hinaus können auch Menschen nicht immer eindeutig die Emotionen anderer Personen eindeutig bestimmen.

10 Ausblick

Vorstellbare Erweiterungen der App sind beispielsweise die Auslagerung der Emotionserkennung in die Cloud. So würde der Loomo nur noch Bild und Sprache aufnehmen, die Auswertungsschritte würden aber ausgelagert. Somit kann bei einer entsprechend latenzarmen Internetverbindung eine schnellere und häufigere Analyse durchgeführt werden. Auch können dann leistungsfähigere Algorithmen benutzt werden, die erhöhte Genauigkeiten versprechen. Allerdings muss bei der Auslagerung in die Cloud viel Wert auf Datenschutz und -sicherheit gelegt werden, da die Zuordnung von Bild und Ton jeder aufgenommenen Person ansonsten in falsche Hände geraten könnten.

Auch ist es vorstellbar, dass für die Sprachanalyse nicht nur die Geräuschpegel verwendet werden, sondern auch eine Untersuchung auf Schlagwörter durchgeführt wird. Diese Schlagwörter müssten so ausgewählt werden, dass sie bestimmte Emotionen implizieren (da sie vor allem im Zusammenhang mit bestimmten Emotionen genutzt werden). Natürlich variieren diese Wörter auch in verschiedenen Sprachen, sodass man dann auch untersuchen müsste ob es Überschneidungen in verschiedenen Sprachen gibt oder man müsste sich auf wenige Sprachen beschränken.
Eine weitere mögliche Verbesserung stellt die Analyse auf Körpersprache dar. Diese erlaubt auch unterstützende Rückschlüsse auf die Emotionen einer Person, allerdings ist an dieser Stelle zu beachten, dass die Körpersprache schon in Mitteleuropa stark variiert und somit eine genaue Festlegung auf einen Kulturkreis erfolgen müsste.
Um eine Datenbasis für die Bilderkennung zu erschaffen, die genau auf die Eigenschaften von Loomo (Blickwinkel) eingeht, könnte die App auch um einen Lernmodus erweitert werden. In diesem hätte der Nutzer die Möglichkeit, ein Feedback zu der erkannten Emotion zu geben. Dieser Input könnte dann dafür genutzt werden, ein neues Modell zu erstellen, mit dem dann die Erkennungsrate verbessert wird.
Letztendlich könnte auch noch probiert werden die ganze Funktionalität für Gruppen zu erweitern bzw. zu verbessern. Momentan läuft es zwar für mehrere Personen auch relativ gut, aber die Performanz könnte in diesem Bereich noch verbessert werden.

Literatur

[Anda] ANDROIDDEVELOPERS. *Android CaptureResult.STATISTICS_FACES*. https://developer.android.com/reference/android/hardware/camera2/CaptureResult#STATISTICS_FACES

[Andb] ANDROIDDEVELOPERS. *com.google.android.gms.vision.face*. https://developers.google.com/android/reference/com/google/android/gms/vision/face/package-summary

[AVTP17] ARRIAGA, Octavio ; VALDENEGRO-TORO, Matias ; PLÖGER, Paul: Real-time convolutional neural networks for emotion and gender classification. In: *arXiv preprint arXiv:1710.07557* (2017)

[BDY+04] BUSSO, Carlos ; DENG, Zhigang ; YILDIRIM, Serdar ; BULUT, Murtaza ; MIN LEE, Chul ; KAZEMZADEH, Abe ; LEE, Sungbok ; NEUMANN, Ulrich ; NARAYANAN, Shrikanth: Analysis of emotion recognition using facial expressions, speech and multimodal information, 2004, S. 205–211

[Bha] BHATIA, Richa. *TensorFlow Vs Caffe: Which Machine Learning Framework Should You Opt For?* https://www.analyticsindiamag.com/tensorflow-vs-caffe-which-machine-learning-framework-should-you-opt-for/

[BPR+05] BURKHARDT, Felix ; PAESCHKE, Astrid ; ROLFES, Melissa A. ; SENDLMEIER, Walter F. ; WEISS, Benjamin: A database of German emotional speech. In: *INTERSPEECH*, 2005

[Bre05] BREAZEAL, Cynthia L.: Sociable machines : expressive social exchange between humans and robots. (2005), 08

[CJOS16] CORREA, E ; JONKER, A ; OZO, M ; STOLK, R: Emotion Recognition using Deep Convolutional Neural Networks. In: *Tech. Report IN4015* (2016)

[CK02] DE CHEVEIGNÉ, Alain ; KAWAHARA, Hideki: YIN, A fundamental frequency estimator for speech and music. In: *The Journal of the Acoustical Society of America* 111 (2002), 05, S. 1917–30

[Cod] CODELABS, Google. *TensorFlow for Poets 2*. https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/index.html#3, Aufruf: 15.1.2019

[DTM14] DU, Shichuan ; TAO, Yong ; MARTINEZ, Aleix M.: Compound facial expressions of emotion. In: *Proceedings of the National Academy of Sciences* 111 (2014), Nr. 15, S. E1454–E1462

[FM18] FREITAS-MAGALHÃES, A: *Facial Action Coding System 3.0-Manual de Codificação Científica da Face Humana-FM BabyFACS*. Leya, 2018

[Gooa] GOOGLE. *TensorFlow Mobile*. https://www.tensorflow.org/lite/tfmobile/, Aufruf:15.01.2019

[Goob] GOOGLE. *TensorFlow Mobile*. https://www.tensorflow.org/lite/

[HUG] HUG TECHNIK UND SICHERHEIT GMBH. *Tabelle Schallpegel Einheiten*. https://www.hug-technik.com/inhalt/ta/schallpegel_laermpegel.html, Aufruf: 15.01.2019

[HZC+17] HOWARD, Andrew G. ; ZHU, Menglong ; CHEN, Bo ; KALENICHENKO, Dmitry ; WANG, Weijun ; WEYAND, Tobias ; ANDREETTO, Marco ; ADAM, Hartwig: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: *arXiv preprint arXiv:1704.04861* (2017)

[kag] KAGGLE. *Challenges in Representation Learning: Facial Expression Recognition Challenge*. `https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data`

[kou] KOUSTUBH. *ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks*. `https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/`

[LCK+10] LUCEY, Patrick ; COHN, Jeffrey F. ; KANADE, Takeo ; SARAGIH, Jason ; AMBADAR, Zara ; MATTHEWS, Iain: The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops* IEEE, 2010, S. 94–101

[LN05] LEE, Chul M. ; NARAYANAN, Shrikanth S.: Toward detecting emotions in spoken dialogs. In: *IEEE Transactions on Speech and Audio Processing* 13 (2005), March, Nr. 2, S. 293–303. – ISSN 1063–6676

[MLG97] MICHAEL, J ; LYONS, Miyuki K. ; GYOBA, Jiro: Japanese female facial expressions (JAFFE). In: *Database of digital images* (1997)

[NCK+18] NOROOZI, Fatemeh ; CORNEANU, Ciprian ; KAMIŃSKA, Dorota ; SAPIŃSKI, Tomasz ; ESCALERA, Sergio ; ANBARJAFARI, Gholamreza: Survey on Emotional Body Gesture Recognition. In: *IEEE Transactions on Affective Computing* (2018), 01

[Ope] OPENCV. *Geometric Transformations of Images*. `https://docs.opencv.org/3.4/da/d6e/tutorial_py_geometric_transformations.html`, Aufruf: 25.03.2019

[PSO+14] PIANA, Stefano ; STAGLIANÒ, Alessandra ; ODONE, Francesca ; VERRI, Alessandro ; CAMURRI, Antonio: Real-time Automatic Emotion Recognition from Body Gestures. (2014), 02

[QSR17] QUINN, Minh-An ; SIVESIND, Grant ; REIS, Guilherme: Real-Time Emotion Recognition From Facial Expressions. In: *Standford University* (2017)

[Rus80] RUSSELL, James A.: A circumplex model of affect. In: *Journal of personality and social psychology* 39 (1980), Nr. 6, S. 1161

[SBGWG91] SCHERER, Klaus ; BANSE, Rainer ; G. WALLBOTT, Harald ; GOLDBECK, Thomas: Vocal cues in emotion encoding and decoding. In: *Motivation and Emotion* 15 (1991), 01, S. 123–148

[Sch95] SCHERER, Klaus: Expression of emotion in voice and music. In: *Journal of voice : official journal of the Voice Foundation* 9 (1995), 10, S. 235–48

[SCL14] SIX, Joren ; CORNELIS, Olmo ; LEMAN, Marc: TarsosDSP, a Real-Time Audio Processing Framework in Java. In: *Proceedings of the 53rd AES Conference (AES 53rd)*, 2014

[Spi16] SPIERS, Daniel L. *Facial emotion detection using deep learning*. 2016

[Spo] SPOOSPA. *A woman with a neutral expression*. https://en.wikipedia.org/wiki/File:Face_of_SpooSpa.jpg

[Ten] TENSORFLOW. *Vergleich von Tensorflow Modellen*. https://www.tensorflow.org/lite/guide/hosted_models

[VJ01] VIOLA, Paul ; JONES, Michael: Robust Real-time Object Detection. In: *International Journal of Computer Vision*, 2001

[YJ] YANGQING JIA, Peter V. *Delivering real-time AI in the palm of your hand*. https://code.fb.com/android/delivering-real-time-ai-in-the-palm-of-your-hand/

[YTK+18] YIP, Benjamin ; TOWNER, Rachel ; KLING, Troy ; CHEN, Cuixian ; WANG, Yishi: Image Pre-processing Using OpenCV Library on MORPH-II Face Database. In: *arXiv preprint arXiv:1811.06934* (2018)

[ZML+13] ZHANG, Weishan ; MENG, Xin ; LU, Qinghua ; RAO, Yuan ; ZHOU, Jiehan: A Hybrid Emotion Recognition on Android Smart Phones, 2013, S. 1313–1318