OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Emotion Recognition

P2 - Deep Learning Eye Catcher

Bericht zum Laborpraktikum
im Sommersemester 2019 von

Lukas Schulz
Matrikelnummer: 3042081

Noor Alrabea
Matrikelnummer: 3103436

Markus Hofer
Matrikelnummer: 3019148

Benjamin Bauer
Matrikelnummer: 3092255

Mona Ziegler
Matrikelnummer: 3091609

Anna Will
Matrikelnummer: 3086698

Julia Karnaukh
Matrikelnummer: 3099379

David Wolf
Matrikelnummer: 3100998

Fakultät Informatik und Mathematik
Ostbayerische Technische Hochschule Regensburg
(OTH Regensburg)

Prüfer: Prof. Dr. Christoph Palm
Abgabedatum: 25. Juni 2019

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation und Einführung

Künstliche Intelligenz ist bereits in vielen Situationen des Lebens alltäglich geworden. So steckt hinter der Gesichtserkennung im Smartphone oder Sprachassistenten wie Amazons "Alexa" Künstliche Intelligenz. Der große Vorteil und Nutzen an KI liegt vor allem in der Mustererkennung und Zuordnung zu vorgegebenen Kategorien. In der Medizin spielt das vor allem bei der Verarbeitung der Ergebnisse Bildgebender Diagnoseverfahren eine große Rolle. Eine KI kann Muster in beispielsweise Röntgenbildern erkennen, und so den Arzt bei der Diagnosefindung unterstützen. Daran wird auch im Labor des ReMIC (Regensburg Medical Image Computing) geforscht und verschiedene Anwendungen zur Bildanalyse entwickelt. Hierbei liegt der Fokus immer mehr auf Deep-Learning Ansätzen. Um nun mehr interessierte Studierende der OTH über diese Tätigkeit zu informieren, soll eine einfache Deep-Learning Anwendung zur Emotionserkennung entwickelt werden. Um so viele Studierende wie möglich dafür zu begeistern, wurde entschieden, ein Spiel zu entwickeln.

1.2 Das Programm

Neben der Tür des ReMIC gibt es ein "Schaufenster", hinter diesem ist ein Bildschirm aufgebaut, der im Moment genutzt wird um allgemeine Informationen über das Labor anzuzeigen. In Zukunft soll dort zwischen den Vorlesungszeiten das entwickelte Programm spielbar sein. Hierfür wird eine Kamera über den Bildschirm aufgebaut.
Das Spiel wird gestartet, wenn das Gesicht eines Spielers in einem ausgewiesenen Bereich erkannt wurde. Dem Spieler wird nun eine zufällig ausgewählte Emotion vorgegeben, die er nun nachmachen soll. Das Programm ist in der Lage sieben Emotionen zu erkennen: Neutral, Angewidert, Wütend, Glücklich, Überrascht, Traurig, Ängstlich. Die erreichte Punktzahl pro Runde ergibt sich daraus, wie gut die Emotion durch das Programm erkannt werden konnte. Die höchste pro Runde erreichbare Punktzahl ist 100, für eine zu 100% erkannte Emotion.
Falls kein oder mehrere Gesichter erkannt wurden, soll eine Fehlermeldung auf dem Bildschirm ausgegeben werden.
Das Spiel endet, wenn der Spieler den von der Kamera erfassten Bereich verlässt. Danach werden die Bilddaten des Spielers gelöscht.

1.3 Künstliche Intelligenz und Deep Learning

Traditionell liegen die Stärken einer Maschine bei rechenaufwändigen, für den Menschen kaum oder nur sehr langsam lösbare Aufgaben. Auch große Datenmengen stellen dabei für einen Computer keine große Herausforderung dar. Im Gegensatz dazu sind Probleme, die der Mensch intuitiv oder situationsabhängig lösen würde, durch konventionelle Programmierung kaum erfassbar. Der Programmierer müsste jede mögliche Situation abfangen, in vielen Fällen ist dies aufgrund des Umfangs nicht effizient umsetzbar. Bei der Emotionserkennung beispielsweise hat der Mensch keine Schwierigkeiten, in verschiedenen Gesichtern Emotionen zu erkennen und zu unterscheiden, obwohl sich die Ausprägung verschiedener Merkmale dabei in jedem Gesicht stark unterscheiden kann. Um eine solche Aufgabe maschinell lösen zu können, wird die Verwendung einer künstlichen Intelligenz notwendig.

Ein Vielversprechender Ansatz zur Entwicklung einer künstlichen Intelligenz ist das Deep Learning. Dabei werden neue, komplexe Probleme gelöst indem sie aus bekannten, einfachen Problemen zusammengesetzt werden. Hierfür werden neuronale Netze verwendet. Diese sind vom Aufbau und der Struktur der Verschaltung des menschlichen Gehirn nachempfunden. Es gibt Knotenpunkte (= Neuronen), die mit anderen Knotenpunkten verbunden sind, und so Informationen weitergeben. Beim Deep Learning sind diese Knoten in Schichten angeordnet. Die erste Schicht ist sichtbar, und umfasst die rohen Eingangsdaten. Diese werden abstrahiert, auf eine nächste versteckte Schicht abgebildet, wieder verarbeitet und an die nächste Schicht weitergegeben. Dies geschieht einige Male, bis die Daten an die letzte Ebene weitergegeben werden. Dort erhält man ein prozentuales Ergebnis, zu welcher Kategorie die Eingangsdaten zugeordnet werden können. Wie genau die versteckten Schichten arbeiten, an welchen Punkten sie sich orientieren, oder worauf die letztendliche Entscheidung basiert, ist von außen nicht ersichtlich. Die internen Parameter müssen nicht von einem Programmierer festgelegt werden, sondern werden vom neuronalen Netz selbst über ein Trainingsverfahren gelernt [1].

Zum Training wird in der Praxis meist Supervised Learning angewendet. Dabei werden neben den Daten, die in eine Kategorie eingeordnet werden sollen, auch der gewünschte Output übergeben. Nun sollen die internen Parameter so angepasst werden, dass sich der Fehler minimiert. Um dieses Minimum zu finden, wird häufig das Gradientenabstiegsverfahren verwendet. Hierfür berechnet der Lernalgorithmus für jeden Parameter einen Gradienten, der angibt, in wie weit sich der fehlerhafte Output verbessert oder verschlechtert, falls der Parameter leicht erhöht wird. Der Parametervektor wird dann in die entgegengesetzte Richtung zum Gradienten angepasst, um so einen minimalen Wert für den Fehler zu erhalten [2].

2 Automatische Emotionserkennung

Mimiken entstehen durch die Kontraktion verschiedener Gesichtsmuskeln, welche zu den Änderungen der jeweiligen Gesichtsmerkmale führen. So kommt es zum Beispiel zum Zusammenkneifen der Augenlider, Heben der Augenbrauen und Lippen oder Hochziehen der Nase. Durch das Auftreten von Grübchen und Fältchen kann sich auch die Textur der Haut verändern. Verschiedene Emotionen führen zu unterschiedlichen charakteristischen Änderungen im Gesicht, wobei manche Emotionen schwieriger zu erkennen sind als andere. Wir beschäftigen uns hier mit den sechs Basis Emotionen: Freude, Trauer, Angst, Ekel, Überraschung und Wut [3]. Wichtig um diese Emotionen zu erkennen sind die Lage, Intensität und Dynamik der entscheidenden Merkmale [3]. Um Emotionen automatisch erkennen zu können müssen drei grundlegende Phasen durchlaufen werden. Als erstes muss das Gesicht und dessen Komponenten erkannt werden, dann müssen die Merkmale extrahiert und letztendlich als ein Ausdruck klassifiziert werden [4].

2.1 Gesichtsdetektion

Der erste Schritt für eine funktionierende Emotionserkennung ist die richtige Detektion des Gesichts. Im besten Fall enthält ein Emotionserkennungsprogramm einen Gesichtsdetektor, der es erlaubt ein Gesicht in komplexen Szenen mit einem schwierigen Hintergrund zu erkennen. Manche Methoden zur Emotionserkennung brauchen die exakte Position des Gesichts, bei anderen, wie beim Active Appearance Modell reicht die grobe Lokalisierung [3].
Bei der Detektion des Gesichts kann man zwischen Merkmal und Bild basierten Methoden unterscheiden. Ersteres eignet sich gut für Echtzeitsysteme bei denen Farbe und Bewegung beteiligt ist, letzteres ist eine stabile Technik für statische Grauwertbilder [5].
Die Entwicklung des merkmalsbasierten Ansatzes kann man weiter in drei Bereiche unterteilen: die Low-Level Analyse, Merkmalsanalyse und der Verwendung aktiver Formmodelle. Die Low-Level Analyse befasst sich zunächst mit der Segmentierung visueller Merkmale unter Verwendung von Pixeleigenschaften wie Graustufen und Farben. Allerdings sind aufgrund der niedrigen Ebene die aus dieser Analyse generierten Merkmale nicht eindeutig. Deshalb macht man als nächstes eine Merkmalsanalyse, bei der visuelle Merkmale, unter Verwendung von Informationen zur Gesichtsgeometrie, zu einem globaleren Konzept des Gesichts und der Gesichtsmerkmale organisiert. Dadurch wird die Mehrdeutigkeit von Merkmalen reduziert und die Position von Gesicht beziehungsweise Gesichtsmerkmalen bestimmt. Der letzte Schritt umfasst die Verwendung aktiver Formmodelle von Schlangen bis hin zu neueren Punktverteilten Modellen (PDM) [5].

Beim bildbasierten Verfahren wird eine Fensterabtasttechnik zum Erfassen von Gesichtern angewendet [5]. Der Fensterabtastalgorithmus ist im Wesentlichen nur eine Suche auf dem Eingabebild nach in allen Maßstäben möglichen Gesichtsorten, wobei es Unterschiede bei der Implementierung dieses Algorithmus für fast alle bildbasierten Systeme gibt. Typischerweise variieren die Größe des Abtastfensters, die Unterabtastrate, die Schrittgröße und die Anzahl der Iterationen in Abhängigkeit von dem vorgeschlagenen Verfahren und der Notwendigkeit eines rechnerisch effizienten Systems [5].

2.2 Merkmal Extrahierung

Hat man das Gesicht einmal detektiert, geht es daran die Merkmale zu extrahieren. Man kann sich hierbei auf das gesamte Gesicht oder nur bestimmte Teilbereiche beziehen [3]. Um Gesichtsausdrücke aus einem frontal aufgenommenen Bild zu erkennen, muss eine Reihe von Schlüsselparametern, die die Gesichtsausdrücke am besten beschreiben, aus dem Bild erarbeitet werden. Diese Parameter, auch Merkmalsvektor genannt, werden genutzt, um später zwischen den verschiedenen Emotionen unterscheiden zu können. Der wichtigste Aspekt einer erfolgreichen Merkmalerkennung ist die Menge an Informationen, die aus einem Bild in den Merkmalsvektor extrahiert werden kann [6]. Wenn der Merkmalsvektor des einen Gesichts mit dem eines anderen Gesichts übereinstimmt, können die beiden Gesichter mittels Merkmalsklassifikation nicht mehr unterschieden werden. Dieser Zustand heißt Merkmal Überlappung und sollte bei einer idealen Merkmal Extrahierung niemals auftreten [6].
Für automatische Emotionserkennungssysteme wurden verschiedene Arten herkömmlicher Ansätze untersucht. Die Gemeinsamkeit dieser Ansätze besteht darin, den Gesichtsbereich zu erfassen und geometrische Merkmale, Erscheinungsmerkmale oder eine Mischung aus beidem aus dem Zielgesicht zu extrahieren [4]. Für die geometrischen Merkmale wird die Beziehung zwischen Gesichtskomponenten verwendet, um einen Merkmalsvektor für das Training zu konstruieren. Die Erscheinungsmerkmale werden normalerweise aus dem globalen Gesichtsbereich extrahiert oder aus Regionen mit unterschiedlichen Arten von Informationen [4].

2.2.1 Gabor Wavelets

Mehrere Untersuchungen in der Bildverarbeitung haben ergeben, dass die Feature Extraktion für Gesichtserkennung- und Tracking mit Gabor Filtern gute Ergebnisse liefert, weshalb sie auch eine vielversprechende Methode für die Emotionserkennung darstellt [6].

Der allgemeine Rahmen zur Musterdarstellung in Bildern basiert hierbei auf topographisch geordneten, räumlich lokalisierten Filtern, die aus einer mehrfach auflösenden und orientierten Sammlung von Gabor Wavelet Funktionen bestehen [7]. Eine 2D-Gabor-Funktion ist eine ebene Welle mit dem Wellenfaktor k, die durch eine Gauß'sche Hüllkurvenfunktion mit der relativen Breite σ begrenzt wird [6]. Um Informationen über Gesichtsausdrücke zu extrahieren, wird jedes Bild mit einer mehrfach

auflösenden und orientierten Sammlung von Gabor Wavelets Gk_+ und Gk_- gefaltet [7]. Das Vorzeichen stellt gerade (+) oder ungerade (-) Phasen dar, während k, der Filterwellenvektor, die räumliche Frequenz und Ausrichtung des Filters bestimmt [7]. Die Antworten der Filter auf das Bild werden zu einem Vektor zusammengefügt, der folgende Komponenten besitzt:

$$R_{\vec{k},\pm}(\vec{r}_0) = \int G_{\vec{k},\pm}(\vec{r}_0,\vec{r})I(\vec{r})d\vec{r}, \text{ mit}$$

$$G_{\vec{k},+}(\vec{r}) = \frac{k^2}{\sigma^2}e^{-k^2\|\vec{r}-\vec{r}_0\|2/2\sigma^2}cos(\vec{k}\cdot(\vec{r}-\vec{r}_0)) - e^{-\sigma^2/2}, \text{ und}$$

$$G_{\vec{k},-}(\vec{r}) = \frac{k^2}{\sigma^2}e^{-k^2\|\vec{r}-\vec{r}_0\|2/2\sigma^2}sin(\vec{k}\cdot(\vec{r}-\vec{r}_0))$$

[7]

Um das System gegenüber der absoluten Beleuchtungsstärke unempfindlich zu machen, wird das Integral des Cosinus Gaborfilters vom Filter subtrahiert. Beim Sinusfilter ist dies nicht notwendig, da dieser nicht von der absoluten Beleuchtungsstärke abhängt [7].
Es werden drei räumliche Frequenzen mit folgenden Wellennummern benötigt: k = $\pi/2$, $\pi/4$, $\pi/8$ [7]. In allen Kalkulationen wird die Bandbreite auf $\sigma = \pi$ gesetzt [7]. Es werden sechs Wellenvektor Orientierungen genutzt, mit Winkeln von 0 bis π, die im gleichmäßigen Abstand von $\pi/6$ angeordnet sind [7].
Die Komponenten des Gabor Vektors R_k werden definiert als die Amplitude der kombinierten geraden und ungeraden Filterantworten $R_{\vec{k}} = \sqrt{R^2_{\vec{k},+} + R^2_{\vec{k},-}}$ [7]. Im Gegensatz zu linearen Filterantworten reagiert hier die Antwortamplitude weniger empfindlich auf Positionsänderungen [7].
Um den Ähnlichkeitsraum von Gabor kodierten Gesichtsbildern zu untersuchen, können Filterantworten mit der gleichen räumlichen Frequenz und Orientierung in übereinstimmenden Punkten zweier Gesichtsbilder verglichen werden [7]. Das normalisierte Skalarprodukt wird verwendet, um die Ähnlichkeit zweier Gabor Antwortvektoren zu quantifizieren [7]. Die Ähnlichkeit wird dabei als Durchschnitt der Gaborvektorähnlichkeit über alle korrespondierenden Gesichtspunkte berechnet. Da Gaborvektoren bei Nachbar Pixel stark korreliert und redundant sind, reicht es aus, den Durchschnitt auf einem relativ spärlichen Gitter, welches das Gesicht bedeckt, zu berechnen [7].
Das hinzufügen mehrerer Gitterpunkte könnte die Leistung des geometrischen Maßes erhöhen, was allerdings mit einem stark erhöhten Rechenaufwand verbunden wäre. Die Lokalisierung der Gitterpunkte ist der teuerste Teil eines voll automatischen Systems [7]. Eine Verbesserung, allerdings nur eine geringfügige, könnte die Kombination von Gabor und Geometrie Systemen bieten[7].

2.2.2 Active Appearance Modell

Active Appearance Modelle (AAM) haben sich als eine gute Methode bewährt, um vordefinierte lineare Formmodelle zu einem Quellbild, welches das zu untersuchende Objekt enthält, auszurichten [8]. Das Prinzip der AAMs besteht darin, sich ihren Form-

und Erscheinungsbildkomponenten durch eine Gradientenabstammungssuche anzupassen [8].
Die Form s eines AAMs wird durch ein 2D trianguliertes Netz beschrieben, wobei insbesondere die Koordinaten der n Gitter Eckpunkte die Form $s = [x_1, y_1, x_2, y_2, \ldots, x_n, y_n]$ definieren [8]. Diese Eckpunkte korrespondieren mit einem Quellbild, von dem aus die Form ausgerichtet wurde [8]. Da AAMs lineare Form Variation erlaubt, kann die Form s als Basisform s_0 ausgedrückt werden, plus einer linearen Kombination von m Formvektoren s_i:
$s = s_0 + \sum_{i=1}^{m} p_i s_i$ (1) [8]
wobei der Koeffizient $p = (p_1, , p_m)^T$ die Formparameter bildet. Diese Formparameter können typischerweise in rigide ähnliche Parameter p_s und nicht rigide Objektdeformationsparameter p_0 unterteilt werden, sodass sich $p_T = [p_s^T, p_0^T]$ ergibt [8]. Ähnlichkeitsparameter werden assoziiert mit einer geometrischen Ähnlichkeitstransformation (zum Beispiel Translation, Rotation, Skalierung) [8]. Die objektspezifischen Parameter sind die Restparameter, die nicht starre geometrische Variationen darstellen, die der bestimmenden Objektform zugeordnet sind (zum Beispiel öffnen des Mundes, Augen schließen, etc.) [8]. Prokrustes Anpassung wird verwendet, um die Grundform s_0 abzuschätzen [8].
Wenn das Gesicht des Probanden durch Abschätzen der Form und Erscheinung der AAM Parameter einmal erkannt wurde, kann man die Information nutzen um folgende Eigenschaften abzuleiten:

SPTS Die normalisierte Form s_n bezieht sich auf die n Eckpunkte für die x- und y-Koordinaten, was in einem rohen 2n-dimensionalen Merkmalsvektor resultiert [8]. Diese Punkte sind die Scheitelpunkte, nachdem alle starren geometrischen Abweichungen (Translation, Rotation und Skalierung) in Bezug auf die Grundform entfernt wurden [8]. Die normalisierte Form s_n kann erhalten werden durch synthetisieren einer Form Instanz von s, durch das Nutzen der Gleichung (1), welche die Ähnlichkeitsparameter p ignoriert [8].

CAPP Das kanonisch normalisierte Erscheinungsbild a_0 bezieht sich auf den Bereich, in dem alle nicht rigiden Formvariationen normalisiert wurden, wobei die Basisform s_0 berücksichtigt wird [8]. Dies wird erreicht durch Anwenden einer stückweise affinen Neigung auf jede dreieckige Form im Quellbild, sodass es mit der Basisgesichtsform einhergeht [8]. Wenn man die starren Formvariationen weglassen würde, würde dies zu einer schlechteren Leistung führen [8].

2.3 Klassifizierung

2.3.1 Neuronale Netze

Ein guter Klassifikator, der für viele Mustererkennungsanwendungen verwendet wird, ist das künstliche neuronale Netzwerk (ANN). Es ist effektiv bei der Modellierung nichtlinearer Abbildungen und hat eine gute Klassifizierungsleistung bei einer relativ

geringen Anzahl von Trainingsbeispielen. Fast alle ANNs können in drei Grundtypen eingeteilt werden: Multilayer Perceptron Klassifikator (MLP), wiederkehrende neuronale Netze (RNN) und Netze für radiale Basisfunktionen (RBF) [9].

Sobald die Struktur von ANN vollständig spezifiziert ist, sind MLP relativ einfach zu Implementieren und besitzen einen gut definierten Trainingsalgorithmus, weshalb sie auch bei der Emotionserkennung eingesetzt werden [7]. ANN-Klassifikatoren weisen jedoch im Allgemeinen viele Entwurfsparameter auf, die normalerweise ad hoc festgelegt werden, wie zum Beispiel die Form der Neuronenaktivierungsfunktion, die Anzahl der verborgenen Schichten und die Anzahl der Neuronen in jeder Schicht [9]. Die Leistung von ANN hängt stark von diesen Parametern ab.

2.3.2 Vector Machines

Ein weiterer Klassifikator ist die Support Vector Machine (SVM), welche ein wichtiges Beispiel für die Diskriminanzklassifikatoren darstellt [9]. In einer Reihe von Mustererkennungsaufgaben, die das Gesicht und die Gesichtsaktionserkennung betreffen, haben sich SVMs als eine nützliche Methode zur Klassifizierung erwiesen [8] und übertreffen nachweislich andere bekannte Klassifizierungsmethoden [9], da die globale Optimalität des Trainingsalgorithmus und die Existenz hervorragender datenabhängiger Verallgemeinerungsgrenzen große Vorteile bieten. SVM-Klassifikatoren basieren hauptsächlich auf der Verwendung von Kernelfunktionen, um die ursprünglichen Features nichtlinear auf einen hochdimensionalen Raum abzubilden, in dem Daten mithilfe eines linearen Klassifikators gut klassifiziert werden können [9]. Da es keine systematische Möglichkeit gibt, die Kernelfunktion auszuwählen, ist die Trennbarkeit der transformierten Merkmale nicht garantiert. Dies stellt bei Mustererkennungsanwendungen allerdings kein Problem dar, weil dort bei vielen keine perfekte Trennung der Trennungsdaten angestrebt wird, um eine Überanpassung zu vermeiden [9].

2.4 Facial Action Coding System

Nachdem die verschiedenen Merkmale des Gesichts extrahiert und klassifiziert wurden, können sie mit Hilfe des Facial Action Coding Systems (FACS) nach Ekman und Friesen in visuelle Klassen kodiert werden.

FACS ist ein umfassendes anatomisch basiertes System zur Erfassung aller visuell erkennbaren Gesichtsbewegungen, welches auf 44 sogenannten Action Units (AUs), sowie verschiedenen Kategorien von Augen und Kopf Positionen beziehungsweise Bewegungen basiert [10]. Kombiniert man diese AUs miteinander entsteht eine große Variation verschiedener Gesichtsausdrücke. Die Verbindung der AUs 12 und 13, welche das Hochziehen der Mundwinkel und Füllen der Backen beschreiben, mit den AUs 25 und 27, die das leichte öffnen des Mundes darstellen, und dem AU 10, dem anheben der oberen Lippe, ist nur eine Kombination von vielen, um ein Lachen zu beschreiben. Nach diesem Prinzip lassen sich alle Emotionen aus den einzelnen AUs zusammensetzen.

Trotz seiner Einschränkungen, das Fehlen einer zeitlichen und detaillierten räumlichen Information, ist diese Methode die weitverbreitetste für das messen menschlicher Gesichtsbewegungen [11].

3 Einsatzgebiete

Automatisierte Emotionserkennung kann in vielen Bereichen zum Einsatz kommen. Besonders in der Marktforschung und im Marketing kann sie von großem Nutzen sein. Viele Firmen haben bereits erkannt, dass es eine gute Werbestrategie ist, die Emotionen des Kunden anzusprechen. Lässt sich der Kunde emotional auf eine Marke ein, steigt nicht nur die Wahrscheinlichkeit, dass er das Produkt kauft, sondern auch dass er es weiterempfiehlt und einem Konkurrenzprodukt vorzieht [12]. Um nun diesen Effekt zu überprüfen, wird bereits heute automatisierte Emotionserkennung angewendet. Auch in der Software Entwicklung kann eine solche Technik während Usability-Tests zum Einsatz kommen um festzustellen, ob der Kunde mit der Nutzung zufrieden ist [13]. In der Medizin könnte sie Rückschlüsse auf das Befinden eines Patienten geben und, eingebaut in ein Smart Home System, im Ernstfall einen Notdienst kontaktieren. Im Bereich der Mensch-Maschinen-Interaktion können mittels Emotionserkennung große Fortschritte erzielt werden. Die menschliche Interaktion ist von Feinheiten geprägt. In der Regel passt man sein eigenes Verhalten der Stimmung seines Gesprächspartners an. So geht man mit einem aggressiven oder verängstigten Menschen anders um als mit jemandem der gut gelaunt ist. Eine Maschine muss diese Emotionen erst erkennen, um dann ihr Verhalten anpassen zu können. Hierfür kann neben dem Gesichtsausdruck auch die Stimme eine große Rolle spielen [14].

Konkretes Anwendungsbeispiel Die durch das Fraunhofer Institut für Integrierte Schaltungen entwickelte Bildanalyse Software "SHORE" (Sophisticated Highspeed Object Recognition Engine) ist in der Lage, neben einer Bestimmung des Geschlechts und Abschätzungen des Alters, Emotionen zu erkennen und zu unterscheiden. Theoretisch könnten 100 Emotionen erfasst werden, bei SHORE wurde sich auf die vier am einfachsten zu unterscheidenden Emotionen beschränkt: glücklich, traurig, überrascht und verärgert. Die Software wurde plattformunabhängig entwickelt, sodass sie in vielen Bereichen Anwendung finden kann. Beworben wird die Software für drei Anwendungsbereiche: Werbung und Marktforschung, Fahrassistenz-Systeme und Medizintechnik. Während bei der Marktforschung hier vor allem die Zielgruppenanalyse als Vorteil genannt wird, soll bei Fahrassistenz-Systemen auch die Stimmungslage des Fahrers analysiert werden, um so Unfallrisiken zu reduzieren. In der Medizintechnik soll die Software zum einen zur Schmerzerkennung von Patienten, die sich nicht mehr verständigen können, angewendet werden können. Zum anderen kann sie, eingebettet in ein Ambient-Assistent-Living-System, verwendet werden um älteren und erkrankten Menschen so lang wie möglich die Möglichkeit geben selbstständig aber dennoch sicher zu leben. Eine weitere Möglichkeit der Verwendung von SHORE besteht darin, in einer Datenbrille eingebunden beispielsweise autistischen Menschen das Leben zu erleichtern. Erkrankte haben dabei in den meisten Fällen große Probleme, die Gefühlslage ihres Ge-

sprächspartners und den Subtext so mancher Aussagen zu erkennen. Durch die über die Brille erhaltenen Informationen kann so der Alltag stark erleichtert werden [15].

4 Material und Methoden

Die Entwicklung von dem Emotionserkennungsspiel ist in drei Bereiche geteilt: die graphische Benutzeroberfläche, die dahinter steckende Logik bzw. das Neuronale Netz, die für die Emotionerkennung zuständig ist und die Schnittstelle von den beiden. Im Folgenden werden die Methoden von den einzelnen Bestandteilen des Programmes beschrieben.

4.1 Neuronale Netze

Auf Grund seiner hohen Geschwindigkeit und großen Anzahl von verschiedenen bereitgestellten Bibliotheken gilt Python als eine der besten Programmiersprachen für die Entwicklung einer Künstlichen Intelligenz-basierten Software.
Python wird in Kombination mit dem Deep Learning Framework TensorFlow angewendet, um die Netz "lehren" und ihre Ergebnisse interpretieren zu können. Bei diesem Projekt wurden die Python Version 3.6 und die TensorFlow Version 1.7 verwendet.
Als Vorlage diente ein Beispielskript [16], das ein Videosignal über die Webkamera empfängt, das Video dann in einzelne Bilder (Frames) zerteilt und jedes Bild nach Emotionen untersucht. Das Skript wurde während des Laborpraktikums umgebaut und an die Benutzeroberfläche und die Schnittstellen angepasst.
Um die einzelnen Bilder zu bewerten reicht das neuronale Netz allein nicht aus. Man braucht auch ein Framework, das Bildverarbeitungsalgorithmen zur Verfügung stellt. So ein Framework ist OpenCV, das sich gut mit Python kompilieren lässt. Die ausgewählte OpenCV Version ist 3.4.
Zur Verfügung standen drei bereits trainierten neuronalen Netze bzw. Modelle:

- FER2013 [17]
- CK+
- FERPlus

Eins von diesen Modellen bindet man ins Skript ein, um auf der Basis von dem ausgewählten Modell die Emotionerkennung zu ermöglichen. Bei dem Emotionserkennungsspiel handelt es sich um den FERPlus-Datensatz [18]. Der Datensatz wurde von einer Forschungsgruppe bei Microsoft entwickelt. Der unterscheidet sich von seinem Vorgänge FER2013 im genauren Crowdsourcing für den Tagging-Vorgang [19]. Dieser Datensatz verfügt über sieben Emotionsklassen: Wut, Ekel, Angst, Freude, Trauer,

Überraschung und neutrale Emotion.
Im Gegensatz zum Beispielsskript, empfängt das für das Spiel entwickelte Skript keinen Stream von der Webkamera, sondern bekommt von der Benutzeroberfläche ein Ordnerverzeichnis übergeben. In dem Ordner befinden sich Bilder, die bereits aus dem Videostream ausgeschnitten worden sind. In einem Ordner sollen Bilder mit der gleichen Emotion abgespeichert werden. Die Bilder werden nacheinander eingelesen.
Zunächst wird es überprüft, ob auf dem Bild ein Gesicht gefunden bzw. erkannt werden kann. Laut den Spielregeln darf es nur einen Spieler geben, deshalb liefert das Skript eine Fehlermeldung zurück, wenn das neuronale Netz mehr als ein oder kein Gesicht erkannt hat. Falls es aber tatsächlich bloß ein Gesicht erkennt, geht das Spiel weiter. Auf jedem Bild wird das Gesicht auf Emotionen untersucht. Das Ergebnis wird auf der Kommandozeile in Form eines Feldes ausgegeben, in dem auf der ersten Position der Index der Emotion steht, die am wahrscheinlichsten erkannt wurde.

4.1.1 Emotionserkennung – Zusammenspiel der verschiedenen Klassen

Die Erkennung einer Emotion anhand der zur Verfügung gestellten Bilder erfolgt durch ein Zusammenspiel der Klassen PrepareModel und EmotionTableInterpreter. Das Starten des Python-Prozesses zur Ermittlung der erkennbaren Emotionen liegt im Aufgabenbereich der PrepareModel-Klasse. Des Weiteren startet PrepareModel die Auswertung der vom Python-Prozess zurückgegeben Werte mittels der EmotionTableInterpreter-Klasse. Ziel ist es, die am deutlichsten erkannte Emotion heraus zu filtern. Der Interpreter befüllt danach das Attribut ReturnObject der PrepareModel-Klasse mit Informationen bezüglich der Auswertung. Dieses Objekt dient zur übermittlung und zur Bewertung der enthaltenen Informationen.

4.1.2 Beschreibung der einzelnen Klassen und deren Funktionalitäten

PrepareModel

PrepareModel – Konstruktor

Im Konstruktor der Klasse PrepareModel wird eine Variable der Klasse Process definiert und mit den entsprechenden Start-Informationen befüllt. Darunter befindet sich beispielsweise der Pfad zu dem auszuführenden Python-Script oder auch der Pfad zu den auszuwertenden Bildern. Final wird die Funktion StartEmoRecTableInterpreter ausgeführt.

StartEmoRecTableInterpreter

In dieser Funktion wird eine neue Instanz der Klasse EmotionTableInterpreter erstellt und dabei die im Konstruktor definierte Prozess-Variable und das Attribut ReturnObject übergeben.

GetReturnObject

GetReturnObject gibt das Attribut ReturnObject der Klasse PrepareModel zurück, welches während der Auswertung befüllt wurde.

ReturnObject

Hierbei handelt es sich um eine Klasse, die zur Rückgabe der durch die Auswertung erlangten Informationen dient. Das Objekt umfasst beschreibende und prozentuale Informationen zur erkannten Emotion. Des Weiteren verfügt es über eine Enumeration, welche den Typ der Rückgabe definiert. Dieser Typ gibt Auskunft über die Auswertbarkeit der in dem Objekt enthaltenen Informationen.

EmotionTableInterpreter

EmotionTableInterpreter – Konstruktor

Dem Konstruktor der EmotionTableInterpreter-Klasse werden sowohl die Prozess-Variable, sowie das Rückgabe-Objekt der PrepareModel-Klasse übergeben. Der Prozess wird innerhalb des Konstruktors gestartet und dessen Ausgabe mit Hilfe von regulären Ausdrücken verarbeitet. Bevor die Auswertung der Prozessausgabe durch die Evaluate-Funktion erfolgt, wird zuerst die Auswertbarkeit der Daten überprüft. Enthält eines der übergebenen Bilder mehr als ein erkanntes Gesicht wird die Auswertung nicht gestartet und entsprechende Informationen mit Hilfe des ReturnObjects zurückgegeben. Enthalten die Bilder kein einziges Gesicht erfolgt ebenfalls keine Auswertung und das Rückgabe-Objekt ist vom Typ „NoFaceDetected“. Die Auswertung durch Evaluate startet, wenn in einer Bilderserie mindestens ein Bild genau ein Gesicht enthält und alle anderen Bilder kein Gesicht oder maximal ein Gesicht aufweisen.

Evaluate

Die Ausgabe des Python-Prozesses liefert Informationen über die im Bild erkannten Emotionen. Erkennbar sind Wut, Ekel, Angst, Fröhlichkeit, Traurigkeit, ein überraschter und ein neutraler Gesichtsausdruck. Die Prozess-Ausgabe pro Bild liefert für jede der oben genannten Emotionen eine Gewichtung. Durch die Funktion Evaluate wird von jedem bewerteten Bild die höchst gewichtete Emotion dem Attribut hdEmoCollection der Klasse EmotionTableInterpreter hinzugefügt. Bei hdEmoCollection handelt es sich um ein Feld des Typs Dictornary, welches sich als Sammlung mehrerer Schlüssel-Wert-Paare definiert. Bei der Zuordnung der sogenannten „highest detected emotion“ durch die Evaluate-Funktion wird wie folgt vorgegangen. Die Beschreibung der Emotion (bspw. „Fear“) dient als Schlüssel eines hdEmoCollection-Schlüssel-Wert-Paares und die Gewichtung der signifikantesten Emotion als Wert. Wird die Emotion „Fear“ mehrmals pro Auswertung erkannt erhöht dies den Wert des hdEmoCollection-Elements um die jeweilige Gewichtung. So liefert die Funktion Evaluate eine Zusammenfassung der am besten erkannten Emotionen mit den aufsummierten Gewichtungen. (Siehe Abbildung 1)

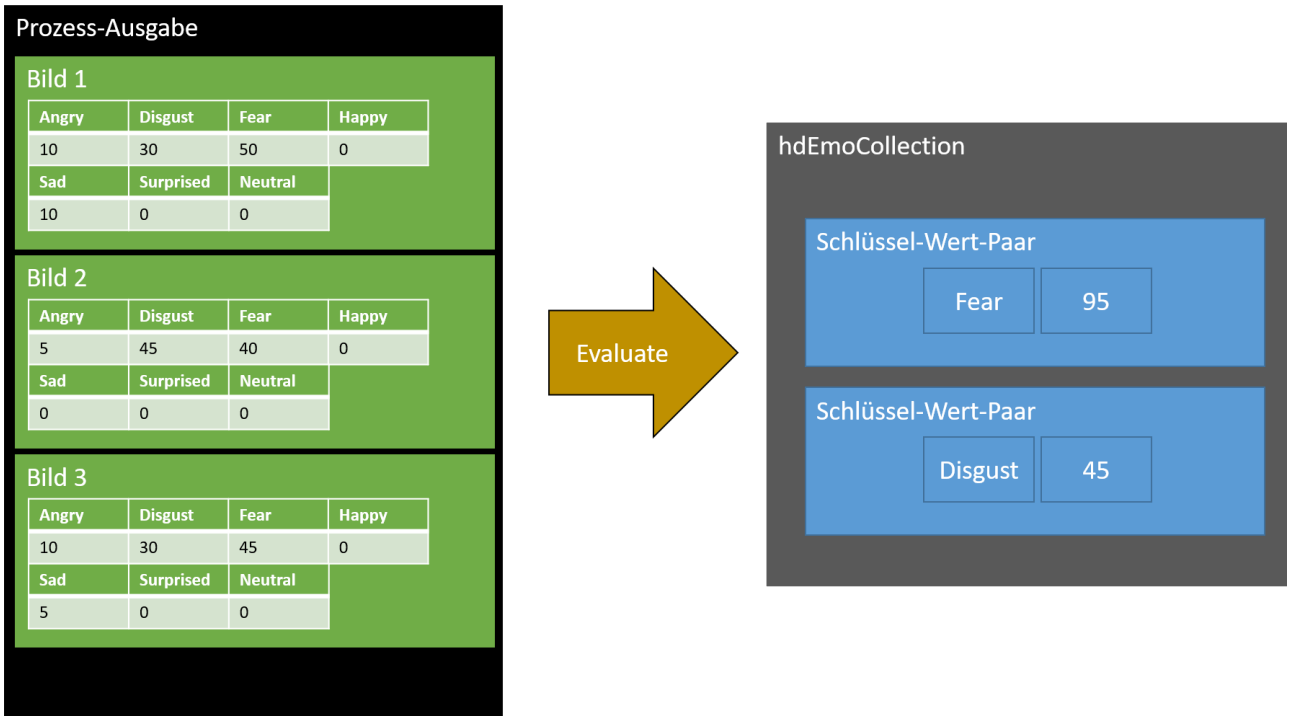

Abbildung 1: Visualisierung des Auswertungsvorgangs

GetEmotion

Diese Funktion liefert das Schlüssel-Wert-Paar der hdEmoCollection, welches den höchsten Wert und somit auch die höchste Gewichtung enthält. Tritt der Fall ein, dass die Emotion „Neutral“ und eine beliebig andere Emotion, beispielsweise „Happy“, den gleichen und höchsten Wert aufweisen, liefert die GetEmotion-Funktion „Happy“. Bei „Neutral“ handelt es sich um keine vom Benutzer zu fordernde Emotion und somit rechtfertigt sich diese Sonderregelung.

4.1.3 Reguläre Ausdrücke in der Ergebniserkennung

In der Klasse EmotionTableInterpreter werden drei Reguläre Ausdrücke benutzt, um die Ergebnisse des Python-Skripts, die als Konsolenausgabe vorliegen, in das Programm zu importieren. Zwei dieser Ausdrücke werden zur Erkennung verwendet, der dritte löscht Steuerzeichen, die im Eingabestring möglicherweise vorhanden sind. Die System.Text.RegularExpressions-Bibliothek bindet die Funktionalität zur Verwendung der Regulären Ausdrücke ein.

Begriffsdefinitionen

- Eingabestring: String, dessen Inhalt auf Matches durchsucht wird.
- Match (plural: Matches): Teilstring(s) des Eingabestrings, der mit dem Pattern übereinstimmt.

- Pattern: Regulärer Ausdruck
- Subpattern: Teil eines Regulären Ausdrucks, der benannt wird, um leichter auf ihn zugreifen zu können. Einzelne Subpattern können auch gematcht werden.

Reguläre Ausdrücke für die Prozessausgabe

Die einzelnen Pattern zur Auswertung der Ausgabe werden im Folgenden erklärt.

patternFaceFoundCount

Das Pattern (Faces found: (?<FacesFoundCount>[0-9]*)){1} hat den Zweck, die Anzahl der erkannten Gesichter in einem Bild auszulesen. Das Subpattern ((?<FacesFoundCount>[0-9]*)){1} mit dem Namen FacesFoundCount hat als Match eine Zahl aus dem Intervall $[0;9] \in \mathbb{N}$ und gibt diese bei einem Aufruf zurück. Dieser Wert wird verwendet, um zu entscheiden, ob die vom Modell erzeugten Daten im EmotionTableInterpreter ausgewertet werden sollen, oder nicht. Im Abschnitt PrepareModel - Konstruktorfindet sich eine genauere Erklärung.

patternArray

Der erste Teil des Patterns heißt SinglePicOutput. Dieses Subpattern umfasst das gesamte Pattern und soll den Zugriff auf den Gesamtoutput eines einzelnen Bildes vereinfachen. Direkt nach der Benennung des Subpatterns beginnt das erste Array, dessen Zahl auf dem Index 0 die erkannte Emotion anzeigt. Auf diese Zahl kann später mit dem Subpattern (ar+ay([(?<HighestEmotionIndex>[0-6]) zugegriffen werden. Ebenso liefert es so den Index im zweiten Array, welches die Gewichtung der vom Netz erkannten Emotion enthält. Die restlichen sechs Elemente des Arrays werden nicht in einem Subpattern erfasst. Das Pattern muss sie trotzdem erkennen, da sonst die Erkennung des zweiten Arrays schwieriger ist. Dies erfolgt mit dem folgenden Abschnitt: ([,][ ]*[0-6])*][, dtype=int[32|64]*]*)[,][]*. Der Abschnitt erkennt eine unbekannte Anzahl an Zahlen aus dem Intervall $[0;6] \in \mathbb{N}$, denen jeweils ein Komma folgt. Zwischen dem Komma der letzten Zahl und der nächsten Zahl kann sich wiederum eine unbekannte Anzahl an Leerzeichen befinden. Die Gewichtungen im zweiten Array, genannt (?<EmotionWeightArray> werden als Fließkommazahlen dargestellt. Niedrige Wahrscheinlichkeiten werden mit Hilfe der Exponentialschreibweise ausgegeben. Dies kann zu Zahlen wie 2.8267652e-05 ($=> 2.8267652 \cdot 10^{-5}$) führen. Da diese Zahlen potentiell alle das Gewicht für die höchste erkannte Emotion darstellen können, werden sie alle getrennt mit Hilfe von eigenen Subpattern erkannt, damit einfach auf sie zugegriffen werden kann. Dies wird im Abschnitt "Ablauf" eingehender besprochen. Die Subpattern für die Fließkommazahlen sind gleich aufgebaut. (?<EmotionWeightArray2>[0-9]*[.]*[0-9]*[e]*[-|+]*[0-9]*) erkennt die Gewichtung an der dritten Stelle im Array. Die Zahl ist in sechs Bereiche unterteilt, die jeweils den Anteil vor und nach dem Komma und dem e der Exponentialschreibweise, sowie das Komma und das e selbst darstellen. Jeder Teil des Subpatterns ist so gestaltet, dass die einzelnen Teile nicht im Eingabestring enthalten sein müssen.

Ablauf

In diesem Teil soll erklärt werden, wie die Regulären Ausdrücke zur Datenextraktion genutzt werden. Zuerst werden aus dem Eingabestring alle Steuerzeichen entfernt. Da die Erkennung dieser aufgrund von Performance-, Stabilitäts- und übersichtlichkeitsgründen nicht im Pattern patternArray enthalten sind. Daraufhin wird auf der Grundlage von patternFaceFoundCount der Eingabestring durchsucht und entsprechende Daten als Variable des Typs MatchCollection gespeichert. Die Daten, die gegen patternFaceFoundCount gematcht wurden, werden mit dem Namen des Subpatterns wie folgt aufgerufen: MatchCollection[Index].Groups[NameDesSubpattern].Value. Diese Daten sind vom Typ String und können in einen anderen Typ konvertiert werden. Hier ist dies notwendig, um auf die Anzahl der erkannten Gesichter zuzugreifen und zu entscheiden, wie weiter vorgegangen werden soll. Danach wird der Eingabestring mit Hilfe des Patterns patternArray noch einmal gematcht und die einzelnen SinglePic-Output-Subpatterns als strings an die Evaluate-Methode übergeben. Dort wird zuerst das Subpattern HighestEmotionIndex ausgelesen und auf die Subpatterns des EmotionWeightArray angewendet, um den durch das Modell zugewiesenen Wert der Gewichtung der höchsten Emotion des Bildes auszulesen. Jetzt wird auf den Namen der als wahrscheinlichsten erkannten Emotion aus dem Member string [] emoDefinition mit dem erkannten Wert des Subpatterns HighestEmotionIndex zugegriffen. Die danach stattfindenden Berechnungen werden in den Abschnitten Evaluate und GetEmotion dargestellt.

4.2 Benutzeroberfläche

4.2.1 Werkzeuge Oberflächenlogik

C#

Was ist C#?
C# ist eine typsichere, objekt-orientierte Programmiersprache. Sie wurde 2001 von Anders Hejlsberg im Auftrag von Microsoft entwickelt. Es ist eine plattformunabhängige Sprache, wurde aber für die .NET-Umgebung entworfen. C# basiert auf den Programmiersprachen C und C++ und ist eng verwandt mit Java[20].
Warum haben wir C# gewählt?
Da C# grundlegend mit denen von uns bereits im Zuge des Studiums gelernten Programmiersprachen C sowie C++ verwandt ist, aber zusätzlich eine zeit- und aufwandsarme Umsetzung des MVVM Schemas bietet, haben wir uns für diese Sprache entschieden.

.NET Framework

Das .NET Framework wurde 2002 von Microsoft entwickelt. Ziele des Frameworks wie auch der Plattform .NET sind unter anderem Kompatibilität zwischen verschiedenen Plattformen, Zusammenführung bisher inkompatibler Microsoft Technologien und Verbesserung veralteter Standards. [21]

Einer der zwei Hauptbestandteile des .NET Framework ist die .NET Klassenbibliothek. Damit ist es möglich sowohl schnell einfache Anwendungen als auch Unternehmensanwendungen im großen Stil zu entwickeln. Auch Umsetzungen der „Gang of Four"-Pattern, sowie andere Strukturen, wie das von uns benutzte Model-View-Viewmodel-Strukturmuster sind im .NET Framework enthalten.

Die Objektorientierung von C# ist auch in der .NET Klassenbibliothek fortgeführt, sodass jedes Element als Objekt, von der Instanzen bestehen können, behandelt wird. Diese Objekte können von wiederum anderen Objekten erben, d.h. diese um Parameter und Funktionen erweitern. Die Wurzel der .NET-Klassenhierarchie ist die Klasse Objekt von der alle anderen Klassen ableiten. Auf diese Weise entsteht eine mehr oder weniger ausgeprägte Baumstruktur[22].

Die zweite Komponente ist die Common Language Runtime (CLR). Die CLR ist die Umgebung, in der die .NET-Anwendungen ausgeführt werden. Diese Umgebung beinhaltet unter anderem Dienste wie:

- Class Loader, um Klassen in die Laufzeitumgebung zu laden
- Type Checker, der unzulässige Typkonvertierungen unterbindet
- Garbage Collector, der eine automatische Speicherbereinigung anstößt, wenn Objekte nicht mehr benötigt werden
- Exception Manager, der die Ausnahmebehandlung unterstützt
- Debug Machine zum Debuggen der Anwendung

, und viele mehr [22]

WPF

Was ist WPF?
Mit dem .NET Framework 3.0 wurde 2006 eine neue Programmierschnittstelle für Windows-Anwendungen eingeführt, die sich Windows Presentation Foundation (WPF) nannte.
Mit WPF lassen sich grafische Benutzeroberflächen entwickeln. Im Gegensatz zum Vorgänger, der WinFormAPI, trennt WPF die Präsentations- und Geschäftslogik. Dadurch wird eine Umsetzung des Model-View-Viewmodel-Strukturmusters möglich.
Die Benutzeroberfläche (View) wird mit einer an XML angelehnten Sprache beschrieben. Diese Sprache heißt eXtensible Application Markup Language (XAML). Dadurch ist es möglich, die Beschreibung der Benutzeroberfläche strikt vom logischen Code zu trennen[22].
Warum haben wir WPF gewählt?
Durch die Trennung von Präsentations- und Geschäftslogik ist eine übersichtliche Strukturierung des Projekts möglich. Da unser Team auf verschiedene Kernthemen unterteilt ist, die im Großen und Ganzen der Aufteilung des WPF entspricht, lag die Wahl auf WPF sehr nahe.

MVVM

Was ist MVVM?
Das Model-View-Viewmodel-Strukturmuster (MVVM) ist eine Variante des Model-View-Controller-Musters (MVC). Hier entspricht das Viewmodel dem Controller und übernimmt dessen Funktion.

- Das Model (Datenmodell) stellt alle für die Verarbeitung einer Anforderung notwendigen Daten und Informationen bereit.
- Der Controller (Viewmodel) steuert die Verarbeitung, er nimmt die Anforderungen entgegen, befüllt das Model und gibt danach die Kontrolle an die View weiter.
- Die View (Ansicht) ist für die Darstellung der Informationen, die sie in Form des Models vom Controller erhält, zuständig[22].

4.2.2 Aufbau und Ablauf der Oberflächenlogik

Der Einstiegspunkt der Oberfläche und sogleich das einzige Fenster des Programms ist das sogenannte "MainWindow". In WPF kann man Fenster in unterschiedliche Dateien aufteilen. Im Falle des "MainWindows" wurde hierbei eine Unterteilung in eine "XAML" und eine "C Sharp" Datei durchgeführt. Die "XAML" Datei legt das Layout des Fensters, sowie alle Bezeichner der Komponenten fest, während die "C Sharp" Datei für die Interaktivität der Komponenten des Fensters sorgt.

Beim Starten der GUI müssen zuerst alle Komponenten des "MainWindows" initialisiert werden. Vorher kann kein Code mit diesen Komponenten arbeiten, da diese ansonsten noch nicht existieren. Anschließend wird in einem neuen Thread die Logik des Programms gestartet.

In WPF läuft die GUI innerhalb des Hauptthreads. Dieser behandelt Interaktionen mit dem User, wie beispielsweise Eingaben oder das Schließen des Fensters. Falls man nun Code innerhalb des Hauptthreads ausführen würde, könnte das System auf keinerlei Eingaben des Users mehr reagieren, da der Thread, der dafür zuständig sein sollte, durch den programmierten Code blockiert wird. Aus diesem Grund muss man für die spezielle Programmlogik einen seperaten Thread erstellen. Dabei ist zu beachten, dass dieser Thread als Background-Thread konfiguriert wird. Das bedeutet, dass beim Schließen des Programms und damit auch Terminieren des Hauptthreads, der seperate Thread ebenfalls terminiert wird. Würde man diese Konfiguration unterlassen, bleibt das Programm nach Schließen der Oberfläche weiterhin aktiv.

Innerhalb des neuen Threads wird eine Instanz von "MainWindowViewModel" erstellt. Das "MainWindowViewModel" stellt die Schnittstelle zwischen der tatsächlichen GUI (View) und der dahinter liegenden Logik und Daten (Model). Das "ViewModel"

ruft die benötigten Abhängigkeiten der Oberfläche auf (beispielsweise das neuronale Netz), verarbeitet deren Antwort und aktualisiert die Oberfläche um den User über den derzeitigen Zustand des Programms zu informieren.

Der Einstieg des "ViewModels" ist die Erkennung des Users. Hierbei kann es zu unterschiedlichen Fällen kommen. Werden mehrere User erkannt oder wird kein User erkannt, bricht das Programm die Analyse ab und probiert es zu einem späteren Zeitpunkt erneut. Falls jedoch ein einziges Gesicht erkannt wurde, startet das Programm eine Spielrunde.

Bei der Änderung der GUI durch das "ViewModel" muss ebenfalls ein wichtiger Punkt beachtet werden. Falls man eine Änderung auf der GUI durchführen möchte, beispielsweise die Änderung des Textes eines Labels, kann dies nicht im Background-Thread durchgeführt werden. Dies ist nur innerhalb des Hauptthreads erlaubt. WPF bietet Möglichkeiten um Aktionen für den Hauptthread einzustellen, die anschließend dort ausgeführt werden. Dies wird vielfach innerhalb das "MainWindowViewModels" aufgerufen.

Eine Spielrunde startet durch das randomisierte Generieren eines Emoticons. Hierbei wird zwischen "Wut", "Ekel", "Angst", "Freude", "Neutral", "Trauer" und "Überraschung" ausgewählt. Diese Emoticons sind als externe Bilder gespeichert und werden bei Bedarf geladen und in der GUI angezeigt. Anschließend wird der User gebeten die Emotion des Emoticons nachzumachen. Nach einer gewissen Zeit, wird ein Bild des Users aufgenommen und an einem vorgegebenen Ort als JPEG Datei gespeichert. Das neuronale Netz übernimmt nachfolgend den Programmablauf und analysiert das gespeicherte Bild.

Diese Analyse kann zu drei unterschiedlichen Fällen führen. Die erste Möglichkeit ist, dass kein Gesicht erkennbar war, da der User beispielsweise das Spiel verlassen hat. Hierbei wird eine Information ausgegeben und das System versucht erneut in regelmäßigen Abständen ein Gesicht für eine neue Spielrunde zu erkennen. Die zweite Möglichkeit ist, dass das neuronale Netz mehrere Gesichter erkannt hat. In diesem Fall wird eine Fehlermeldung ausgegeben und eine neue Spielrunde gestartet. Die letzte Möglichkeit ist, dass nur ein User erkannt wird. Das neuronale Netz gibt im Erfolgsfall eine Prozentzahl an, die spezifiziert wie gut eine bestimmte Emotion erkannt wurde. Wichtig ist, dass nur die Emotion mit der größten Prozentzahl verarbeitet wird. Falls diese mit der randomisierten Emotion übereinstimmt, bekommt der User die Prozentzahl auf sein Punktekonto gutgeschrieben. Andernfalls bekommt der User keine zusätzlichen Punkte und die nächste Runde beginnt.

4.2.3 WebCam

Die Webcam wird von der MainWindow-Klasse gesteuert. Sie ist über die Using-Direktive WebEye.Controls.Wpf eingebunden. Sobald das Fenster nach dem Start der

Applikation geladen hat, wird die Webcam über die Funktion OnWindowLoaded initialisiert. In der Funktion wird dabei zuerst eine Liste aller Webcams, die dem System verfügbar sind, erstellt. Die Webcam an der erster Stelle dieser Liste wird anschließend mit einer Funktion der Klasse webCameraControl gestartet. Dieser Kamerastream wird in der MainWindowViewModel-Klasse dazu benutzt Bilder für das neuronale Netz zu machen. Dies geschieht im fünf Sekunden Takt, wenn kein Gesicht erkannt wurde und im elf Sekunden Takt, wenn das Spiel gestartet wurde. Mithilfe des BitmapConverter wird das aktuelle Webcam Bild als Objekt des Typs BitmapSource gespeichert, das anschließend beim Verwenden des neuronalen Netzes genutzt wird.

5 Fazit, Diskussion und Zusammenfassung

5.1 Ergebnisse

5.1.1 Ergebnisse in der Programmierung

Die Klassen PrepareModel, EmotionTableInterpreter und ReturnObject wurden erstellt und so miteinander verbunden, dass sie einen externen Prozess starten können und die Ausgabe dieses Prozesses in den eigenen Prozess einlesen, verarbeiten und weitergeben. Die Klasse PrepareModel stellt je ein Objekt der Klassen System.Diagnostics.Process, ReturnObject und übergibt sie einem ebenfalls bereitgestellten Objekt der Klasse EmotionTableInterpreter. In der Klasse EmotionTableInterpreter wird das System.Diagnostics.Process-Objekt gestartet. Diese Aktion startet eine Instanz des externen Programms Python.exe. In diesem Programm werden vom aufrufenden Programm zur Verfügung gestellte Bilder mit Hilfe eines von Lewe Ohlsen [18] erstellten und trainierten neuronalen Netzes auf der Grundlage des FER+-Trainingsdatensatzes aus. Diese Daten werden als Objekt des Typs String wieder in die Klasse EmotionTableInterpreter eingelesen und dort von selbst entwickelten regulären Ausdrücken mit Hilfe der Systembibliothek System.Text.RegularExpressions auf die relevanten Daten untersucht und an das ReturnObject-Objekt übergeben. Dieses kann dann weiter verarbeitet werden. Teile des in Python geschriebenen, zum neuronalen Netz gehörigen Skripts wurden auf die Bedürfnisse des Programms angepasst. So ruft nun das Python-Skript abgespeicherte Bilder auf, anstatt dauernd den Input einer Webcam zu verarbeiten. Zusätzlich wurde der Aufruf der GUI des Skripts gelöscht.

5.1.2 Ergebnisse in der Verwendung aller Programmteile

Vor der ersten Ausführung des Programms müssen mehrere Punkte beachtet werden.

- Es muss Python 3.6.3 oder höher auf dem Zielgerät installiert sein.
- Python muss in die PATH-Variable des Zielgerätes eingefügt worden sein.
- Das Netz muss wie in der Anleitung[16] von Lewe Ohlsen mit dem Kommandozeilenbefehl python pip -install -r ”requirements.txt” installiert werden.

Im Betrieb des Programms fallen verschiedene Besonderheiten mit Bezug zum neuronalen Netz auf, die vor einer Installation bedacht werden sollten.

- Die ”Positiven” und die ”neutrale” Emotion werden am Einfachsten erkannt. ”Negative” Emotionen werden seltener erkannt.

- Manchmal werden Gesichter in Strukturen erkannt, die kein Gesicht sind. Verschiedene Experimente konnten keine Struktur hinter diesem Verhalten aufzeigen.

- Personen, die Brillen tragen müssen diese vor dem Spielstart abnehmen, da die Brillen das Ergebnis der Auswertung verändern und meistens die "Emotion" "Überrascht" erkannt wird.

5.2 Diskussion der Ergebnisse

Im programmierbaren Teil finden sich keine Überraschungen. Alle Teile der Sprache C# haben wie erwartet miteinander funktioniert und die gewünschten Ergebnisse geliefert. Der einfache Aufruf von Python-Skripten aus dem Programm heraus war eine erfreuliche Überraschung, die sehr viel Entwicklungsarbeit ersparte. Die regulären Ausdrücke kommen im Allgemeinen nicht sonderlich gut mit im Eingabestring vorhandenen Steuerzeichen zurecht. Dieser Fehler wurde erst in der Testphase erkannt und behoben. Der aktuell gewählte Weg ist zwar nicht optimal, entfernt aber Steuerzeichen zuverlässig. Leider war die Ausgabe des mit CK+ trainierten Models nicht so einfach auswertbar wie die Ausgabe des mit FER+ trainierten Models. Da CK+ manchmal als bessererer Trainingsdatensatz zur Emotionserkennung eingestuft wird, beeinflußt dieses Ereignis die Daten der Auswertung in ihrer Qualität negativ. Vor allem die Erkennung der "negativen" Emotionen ist im CK+-Datensatz erheblich verbessert. Nicht überraschend waren die Effekte, die erzeugt werden können, wenn ein Bild eines Brillenträgers ausgewertet wird. Aufgrund der Arbeitsweise des neuronalen Netzes werden Brillen als "aufgerissene Augen" erkannt. Das kann ein Nachteil für Brillenträger sein, die den angezeigten Emoticon vielleicht nicht erkennen können. Solche Nachteile müssen mit einer geschickten Platzierung der Kamera ausgeglichen werden.

5.3 Zusammenfassung

Unser gemeinsames Ziel als NN-Gruppe war es, dem neuronalen Netz von Lewe Ohlsen ein Gerüst zu geben, mit dem es ohne Probleme von einem C#-Programm aufgerufen und sein Output eingelesen werden kann.Wir verwendeten die Mölichkeiten, die uns von der Sprachen C# und Python zur Verfügung gestellt wurden. Besonders hervorzuheben sind System.Text.RegularExpressions und System.Diagnostics, welchen den Start, das Auslesen und das Auswerten des Ergebnisses eines von diesem Programm gestarteten Prozesses ermölichen. OpenCV und Tensorflow werden genutzt, um die auszuwertenden Bilder vorzubereiten und zu verarbeiten.Es wurden drei Klassen geschrieben, die einen bestimmten externen Prozess starten und auswerten können. Das Programm des Prozesses ist in einer anderen Programmiersprache geschrieben und die Ergebnisse müssen von den Klassen als String eingelesen und mit den geeigneten Mitteln ausgewertet werden. Die Ergebnisse des Python-Prozesses hängen in besonderem Maße vom gewählten Model ab. Je nach dem mit welchem Datensatz das Model trainiert wurde, werden Gesichtsausdrücke unterschiedlich bewertet und ausgegeben. Die ausgewerteten Daten werden mit regulären Ausdrücken verglichen und von geeigneten

Funktionen auf die Häufigkeit der aufgetretenen Emotion untersucht. Diese Information wird danach in einem Objekt an die aufrufenden Klassen zurückgegeben.
Aufgrund der Daten und der Umsetzung lässt sich schlussfolgern, dass das Programm erfolgreich Gesichtsausdrücken "Emotionen" erfolgreich zuordnen kann und die Ergebnisse dieser Zuordnung an aufrufende Klassen und Programme ausgeben kann.

Literatur

[1] I. Goodfellow, Y. Bengio und A. Courville. Deep Learning. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Y. LeCun, Y. Bengio und G. Hinton. Deep learning. Nature 521, 436 EP -, 2015.

[3] B. Fasel und J. Lüttin. Automatic Facial Expression Analysis: A survey. Pattern Recognition 36, 259–275, 2002.

[4] B. C. Ko. A Brief Review of Facial Emotion Recognition Based on Visual Information. Sensors 18(2), 401, 2018.

[5] E. Hjelmås und B. K. Low. Face Detection: A Survey. Computer Vision and Image Understanding 83(3), 236–274, 2001.

[6] S. Bashyal und G. K. Venayagamoorthy. Recognition of facial expressions using Gabor wavelets and learning vector quantization. Engineering Applications of Artificial Intelligence 21(7), 1056–1064, 2008.

[7] M. Lyons, S. Akamatsu, M. Kamachi und J. Gyoba. Coding facial expressions with Gabor wavelets. In: Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition, 200–205, Apr. 1998.

[8] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar und I. Matthews. 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, 94–101, Juni 2010.

[9] M. E. Ayadi, M. S. Kamel und F. Karray. Survey on speech emotion recognition: Features, classification schemes, and databases. Pattern Recognition 44(3), 572–587, 2011.

[10] P. Ekman und E. L. Rosenberg. What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS). Oxford University Press Inc., 1997.

[11] I. A. Essa und A. P. Pentland. Coding, analysis, interpretation, and recognition of facial expressions. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7), 757–763, Juli 1997.

[12] D. Consoli. A new concept of marketing: the emotional marketing. BRAND. Broad Research in Accounting, Negotiation, and Distribution 1(1), 52–59, 2010.

[13] A. Kołakowska, A. Landowska, M. Szwoch, W. Szwoch und M. R. Wróbel. "Emotion Recognition and Its Applications". In: Human-Computer Systems Interaction: Backgrounds and Applications 3. Hrsg. von Z. S. Hippe, J. L. Kulikowski, T. Mroczek und J. Wtorek. Springer International Publishing, S. 51–62. URL: https://doi.org/10.1007/978-3-319-08491-6_5.

[14] M. Brand, F. Klompmaker, P. Schleining und F. Weiß. Automatische Emotionserkennung – Technologien, Deutung und Anwendungen. Informatik-Spektrum 35(6), 424–432, Dez. 2012.

[15] F.-I. für Integrierte Schaltungen. Mit der Bildanalysesoftware SHORE® zur erfolgreichen Erkennung und Auswertung von Gesichtern. URL: https://www.iis.fraunhofer.de/de/ff/sse/ils/tech/shore-facedetection.html (besucht am 21. 06. 2019).

[16] L. Ohlsen. facial-expression-recognition. URL: https://github.com/leweohlsen/facial-expression-recognition (besucht am 23. 06. 2019).

[17] I. Goodfellow. Challenges in representation learning: A report on three machine learning contests, 2013.

[18] L. Ohlsen. Facial Expression Recognition with Convolutional Neural Networks, 2018.

[19] R. K. Pandey, S. Karmakar, A. G. Ramakrishnan und N. Saha. Improving Facial Emotion Recognition Systems Using Gradient and Laplacian Images, 2019.

[20] C# 2.0 : Complete Reference.

[21] I. Landwerth. Introducing .NET Standard. URL: https://devblogs.microsoft.com/dotnet/introducing-net-standard/ (besucht am 20. 06. 2019).

[22] A. Kühnel. Visual C# 2010 : Das umfassende Handbuch. Galileo Computing, 2010.