Kapitel 1

Material und Methoden

Die Entwicklung des Emotionserkennungsspiel ist in drei Bereichen geteilt: die graphische Benutzeroberfläche, die dahinter steckende Logik bzw. das Neuronale Netz, das für die Emotionserkennung zuständig ist und die Schnittstelle zwischen beiden. Im Folgenden werden die Methoden der einzelnen Bestandteile des Programms beschrieben.

1.1 Neuronale Netze

Auf Grund seiner höheren 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 das Netz l̈ehrenünd seine 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 [**?**], 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 einzelne Bilder zu bewerten reicht das neuronale Netz allein nicht aus. Man benötigt auch ein Framework, das Bildverarbeitungs-Algorithmen zur Verfügung stellt. Ein solches Framework ist OpenCV, da es sich gut mit Python kompilieren lässt. Zur Verfügung standen drei bereits trainierten neuronalen Netze bzw. Modelle:

- FER2013 [**?**]
- CK+
- FERPlus

Eines dieser Modelle bindet man in das Skript ein, um auf der Basis des ausgewählten Modells die Emotionserkennung zu ermöglichen. Bei dem im Emotionserkennungsspiel verwendeten Datensatz handelt es sich um den FERPlus-Datensatz. [**?**] Der Datensatz wurde von einer Forschungsgruppe bei Microsoft entwickelt. Er unterscheidet sich von seinem Vorgänger FER2013 im genaueren Crowdsourcing für den Tagging- Vorgang. [**?**] 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 nacheinander werde eingelesen. Zunächst wird überprüft, ob auf dem Bild ein Gesicht gefunden bzw. erkannt werden kann. Laut den Spielregeln darf es nur einen Spieler geben, daher liefert das Skript eine bestimmte Meldung zurück, sollte die Neuronale Netz mehr als ein Gesicht oder kein Gesicht erkennen. Falls aber tatsächlich ein Gesicht erkannt wird, geht das Spiel weiter. Das Gesicht auf jedem Bild wird nach Emotionen Untersucht. Das Ergebnis wird auf der Kommandozeile in Form eines Feldes ausgegeben, in dem auf der ersten Position das Index von der Emotion steht, die am wahrscheinlichsten erkannt wurde.

Kapitel 2

Klassenbeschreibung

2.1 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.

2.2 Beschreibung der einzelnen Klassen und deren Funktionalitäten

2.2.1 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.

2.2.2 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.

2.2.3 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

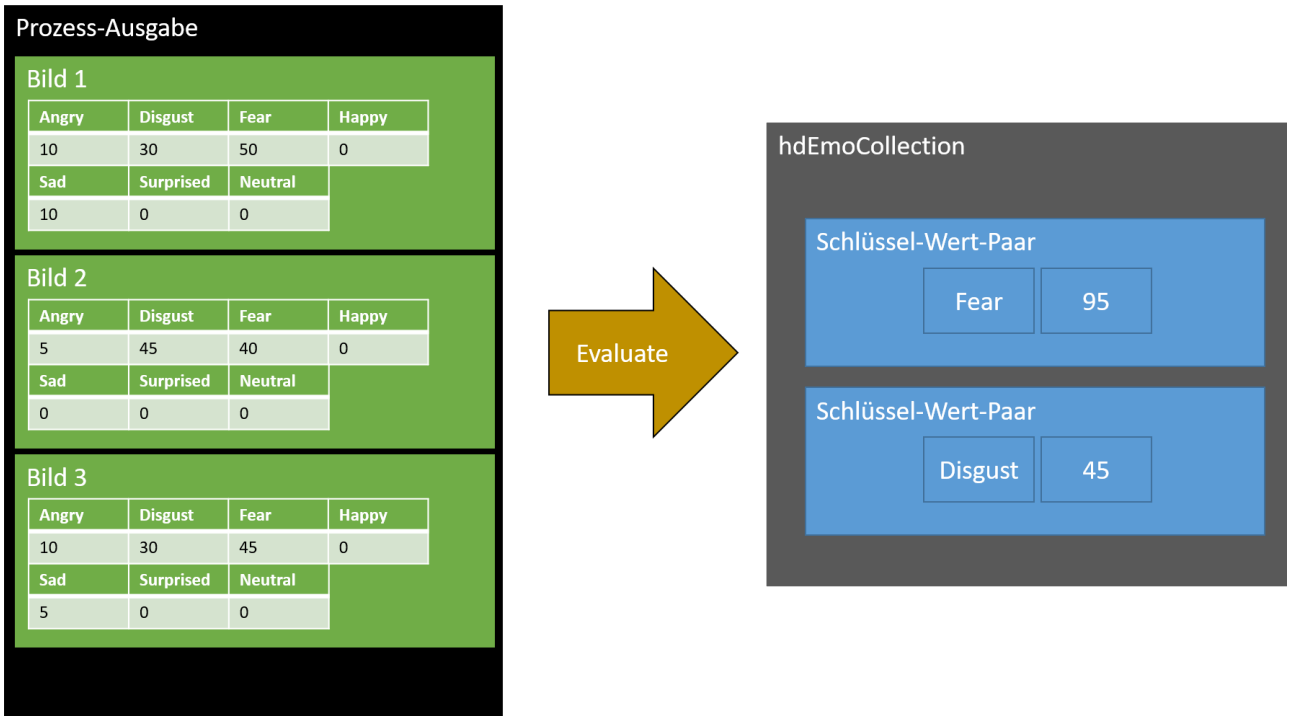

Abbildung 2.1: Visualisierung des Auswertungsvorgangs

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 obengenannten 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 2.1)

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.

2.3 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.

2.3.1 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.

2.3.2 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 P̈repareModel - Konstruktorf̈indet 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 Äblaufëingehender 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.

2.3.3 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-, Stabilitääts- 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 eine 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 *SinglePicOutput*-Subpatterns als *strings* an die *Evaluate*-Methode üübergeben. 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.

Kapitel 3

Fazit, Diskussion und Zusammenfassung

3.1 Ergebnisse

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 [**?**] 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.

Ergebnisse in der Verwendung aller Programmteile

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

- Es muss Python 3.6 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[**?**] 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.

3.2 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 werden. 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 besserer 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 Emoji vielleicht nicht erkennen können. Solche Nachteile müssen mit einer geschickten Platzierung der Kamera ausgeglichen werden.

3.3 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.