Emotion Recognition and LED-Matrix Feedback

Frank Matthiesen (ehemals. Kindler)

A-Kennung: acj102

HAW Hamburg

frank.matthiesen@haw-hamburg.de

Abstract

Die vorliegende Seminararbeit beschäftigt sich mit dem alljährlichen Problem der Niedergeschlagenheit in der Herbst und Winterzeit. Nach der Theorie Facial Feedback Hypothese und den belegenden Studien von Strack und anderen (Strack, Martin, & Stepper, 1988) wird ein Gerät entwickelt dass den Nutzer auf seine Emotion hinweist. Zugrunde liegend ist ein *Embedded System* (Raspberry Pi 4) welches über eine Kamera zur Erkennung von Gesichtern und eine LED-Matrix für ein visuelles Feedback ausgestattet ist. Auf dem System läuft eine Software die basierend auf einem *Convolutional Neural Network* Gesichter verschiedene Emotionen einstuft und darauf basierend verschiedene Animationen auf der LED-Matrix darstellt. Das Neuronale besitzt hierfür 7 verschiedene Klassen und basiert auf dem FER-2013 Datensatz (*FER2013 Dataset*, 2018). Die Emotionserkennung erreicht hierbei eine Erkennungsrate von 65%

1 Einleitung

1.1 Motivation

Diese Arbeit entsteht am Ende des Jahres 2020. Die Tage werden kürzer, es wird kalt, verregnet und aufgrund der Pandemie Schutzbestimmungen ist der Alltag trist und abwechslungsarm. Da es laut Studien möglich ist die eigene Grundstimmung durch gezielte Muskelkontraktion zu verbessern (Strack et al., 1988), liegt es Nahe in eben dieser Zeit öfter daran erinnert zu werden. Da man Zuhause aber nicht immer eine Person hat die daran erinnert, bietet sich ein Computersystem zur Unterstützung an.

1.2 Aufgabe

Diese Arbeit beschäftigt sich damit die Emotionen von Personen zu erkennen. Dazu soll ein Computersystem, welches mit einer Kamera ausgestattet ist, zunächst die Gesichter von Personen erkennen. Hierfür wird die Objekterkennung von *OpenCV* eingesetzt (*OpenCV Cascade Classifier*, n.d.).

Wird ein Gesicht erkannt soll ein *Convolutional Neural Network* analysieren welche Emotion eben dieses ausdrückt. Die Entwicklung dafür basiert auf den Lehrveranstaltungen von Prof. Dr. Stephan Pareigis.

Für das anlernen des neuronalen Netzes wird der *FER-2013* (*FER2013 Dataset*, 2018) Datensatz verwendet. Dieser besteht aus $\sim$ 28.000 Bildern für das Trainieren des Netzes und $\sim$ 3.500 für die Validierung. Der Datensatz basiert auf *Google* Suchen für jede Emotion und seine Synonyme. Jedes Bild ist gelabeled als eine von 7 Emotionen.

Abschließend soll noch ein visuelles Feedback ausgegeben werden. Dazu wird an das Computer System eine LED Matrix angeschlossen und eine Animation bei allen anderen Emotionen als *Happy* wiedergegeben

werden. Für die Darstellung wird die Luma Bibliothek (*Luma Library*, 2017) genutzt um Wörter, Formen und Punkte auf der Matrix einfach zeichnen zu können.

2 Technisches Setup

2.1 Embedded System

Das hier verwendete Embedded System ist ein Raspberry Pi der vierten Generation, siehe Abbildung 1. Das Broadcom BCM2711 SoC bietet 4 64-Bit Arm-Cortex-A72 Rechenkerne die mit jeweils 1.5 Ghz laufen. Damit bietet er eine Rechenleistung die ausreichend ist für die Erkennung der Gesichter und Ihrer Emotionen. Für die Kamera ist ein serielles Interface vorhanden. Für die LED-Matrix wird *SPI* über die GPIOs genutzt. Der Raspberry Pi verfügt über ein WLAN Modul. Dadurch lässt sich das Projekt an verschiedenen Orten aufbauen und via HTML Interface konfigurieren und warten.

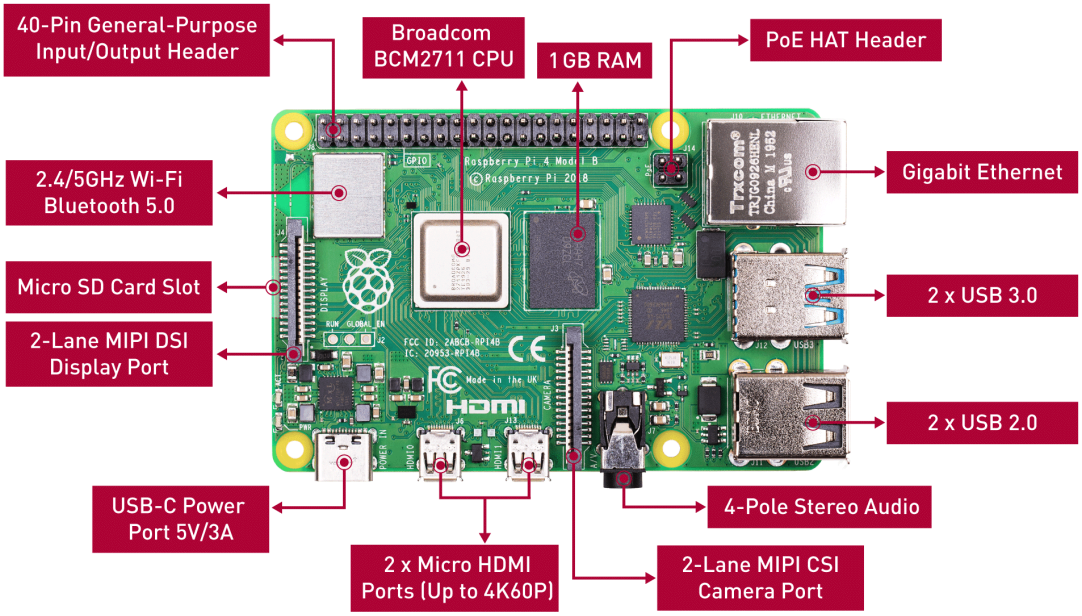

Figure 1: Raspberry Pi Hardware Overview

2.2 Kamera

Als Kamera kommt das Kamera Modul V2 von der Raspberry Pi Foundation zum Einsatz, siehe Abbildung 2. Es setzt auf einen IMX219PQ Bildsensor. Die Anbindung erfolgt via 15-polige serielle MIPI-Schnittstelle (CSI-2). Da Embedded System und Kamera vom gleichen Hersteller stammen, ist eine hohe Kompatibilität zu erwarten.

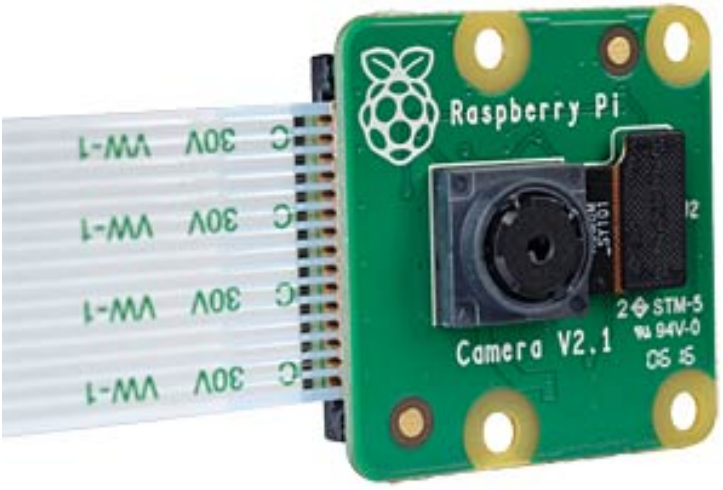

Figure 2: Raspberry Pi Kamera Modul V2

2.3 LED-Matrix

Als LED-Matrix kommt ein Verbund von 4 kleineren Matrizen zum Einsatz, siehe 3. Jedes Modul besteht aus einer LED Matrix und einem MAX7219 LED-Treiber. Dieser unterstützt das SPI Protokoll und erlaubt es mehrere Module hintereinander zu verketten. Dadurch lassen sich Matrizen einfach generieren und ansprechen.

Figure 3: LED-Matrix aus MAX7219 Modulen

3 Projekt Ablauf

3.1 Datensatz Aufbereitung

Für dieses Projekt stellte sich die Frage, ob ich ein eigener Datensatz mit Bildern verschiedener Emotionen gesetzt werden soll. Ein solches Szenario wäre das fotografieren von verschiedenen Person die aufgefordert werden bestimmte Emotionen zu zeigen. Es wird erwartet, dass eine Vielzahl von Personen hier ein eine hohe Zuverlässigkeit im späteren Betrieb ergeben. Aufgrund der Corona-Pandemie und der damit verbunden Kontaktbeschränkungen zum eigenen und zum Schutz anderer wird allerdings auf einen bestehenden Datensatz gesetzt.

Zum Einsatz kommt der FER-2013 Datensatz (*FER2013 Dataset*, 2018). Der Datensatz besteht aus 28709 Bildern die dem Trainieren des Neuronalen Netzes dienen sollen und 3589 Bilder die anschließend dieses Testen sollen. Jedes Bild ist in Graustufen und hat eine Auflösung von 48x48 Pixeln. Zu sehen sind sowohl Frontansichten von Köpfen als auch im Portrait und weiteren Abstufungen. Jedes Bild verfügt zu dem über ein Label der darin gezeigten Emotion. Der Datensatz verfügt über die Emotionen:

- *fear*
- *surprise*
- *neutral*
- *happy*
- *angry*
- *disgust*
- *sad*

Figure 4: Beispiele der im Datensatz enthaltenden Fotos und ihrer Label

Abbildung 4 zeigt Beispiele aus dem Datensatz zu allen enthaltenen Emotionen.

Aufgeteilt sind die Emotionen in Training und Test Datensatz gleichmäßig wie in Abbildung 5 zu sehen. Die signifikant niedrige Zahl von Bildern der Emotion *Disgust* ist zu vernachlässigen, da diese Emotion in diesem Projekt keine Rolle spielen soll.

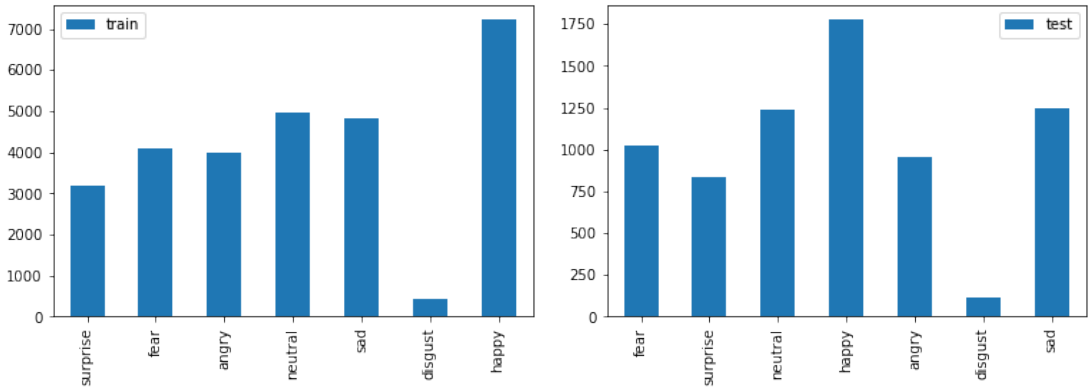

Figure 5: Verteilung der Emotionen in den Datensätzen

Um die Erkennung noch flexibler zu machen, werden alle Bilder die zum Trainieren des Neuronalen Netzwerks genutzt werden noch zusätzlich künstlich Erweitert sog. *Image Augmentation*. Hierzu wird der `ImageDataGenerator` des Keras Framework (*Image data preprocessing*, n.d.) genutzt.

```
from keras.preprocessing.image import ImageDataGenerator

aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
```

Figure 6: Image Augmentation mit dem des Kera Frameworks bereitgestellten ImageDataGenerator

Das Listing 6 sorgt nur dafür, dass die Bilder beispielsweise gedreht, vergrößert oder gedreht werden. Was genau welcher Parameter soll an dieser Stelle nicht weiter vertieft werden. Weitere Informationen und Beispiele sind hier zu finden.

3.2 Neuronales Netzwerk Modellieren und Trainieren

In diesem Projekt wird ein Convolutional Neural Network eingesetzt. Das eingesetzte besteht aus drei Convolutional Blöcken zur Erfassung von von Strukturen innerhalb der Bilder. Jeder Block beginnt mit Convolutional Layern die auf der doppelten Anzahl von Neuronen im Vergleich zur vorherigen setzt. Versuche mit verschiedenen Anzahl an Blöcken haben gezeigt, dass die 3 Blöcke das beste Verhältnis an Komplexität und Genauigkeit liefern, siehe Abbildung 7 zu sehen.

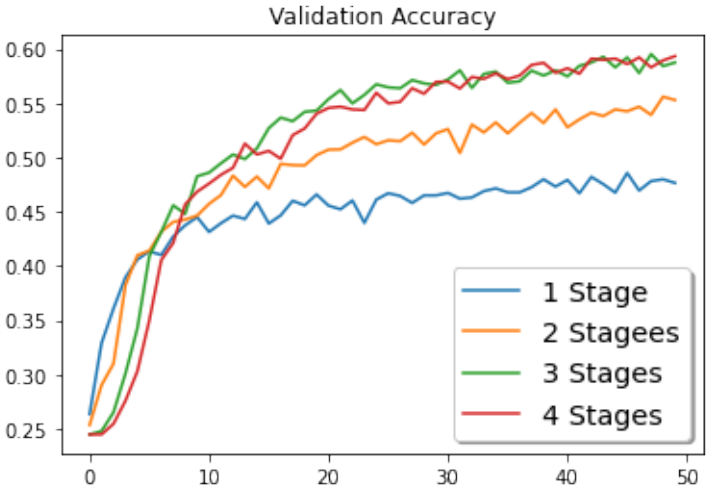

Figure 7: Genauigkeit des Netzes mit verschiedener Anzahl von Convolutional Blöcken. Jeder Conv Block mit zwei aufeinander folgenden Convolutional Layern. Erster Convolutional Layer besteht aus 32 Neuronen.

Jeder Block besteht aus zwei aufeinander folgenden Convolutional Layern. Die Layer bestehen aus der selben Anzahl von Neuronen. Zwei Layer erwies sich als optimale Zahl, siehe Abbildung 8. Der Faltungskskern hat eine Größe von $3 * 3$. Darauf folgt ein ein MaxPooling-Layer (*MaxPooling2D Layer*, n.d.) um markante Teil der Bilder hervorzuheben.

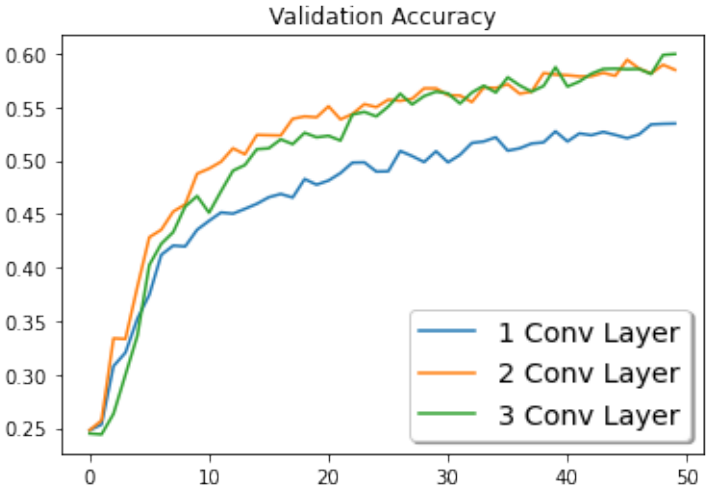

Figure 8: Genauigkeit des Netzes mit verschiedener Anzahl von aufeinander folgenden Convolutional Layern pro Block. Erster Convolutional Layer besteht aus 32 Neuronen.

Die Anzahl der Neuronen pro Convolutional Block multipliziert sich pro Schritt um den Faktor 2. Es hat sich gezeigt, dass 32 Neuronen im ersten Block, 64 im zweiten und 128 im dritten eine gutes Ergebnis liefert, siehe Abbildung 10.

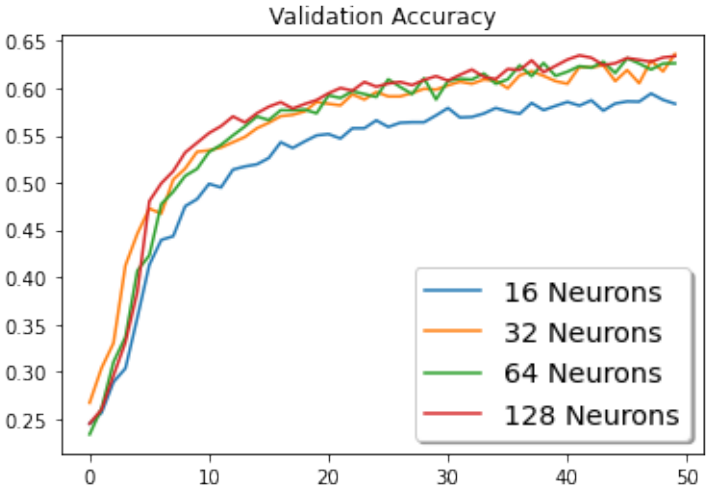

Figure 9: Genauigkeit des Netzes mit verschiedener Anzahl von Neuronen im ersten Block.

Aufgrund der *Image Augmentation* kann auf die oft verwendeten Dropout-Layer (*Dropout Layer*, n.d.) verzichtet werden. Diese sollen verhindern, dass es zu einem *Overfitting* kommt. Dropout Layer verwerfen in einem bestimmten Maße Inputs damit die Generalisierungsfähigkeit nicht verloren geht. Im schlimmsten Fall erkennt das Netzwerk all die Trainingsbilder, aber keine anderen Bilder. Abbildung **??** zeigt, dass ohne Dropout-Layer die Erkennungsrate höher ist und es keine Anzeichen für ein Overfitting gibt.

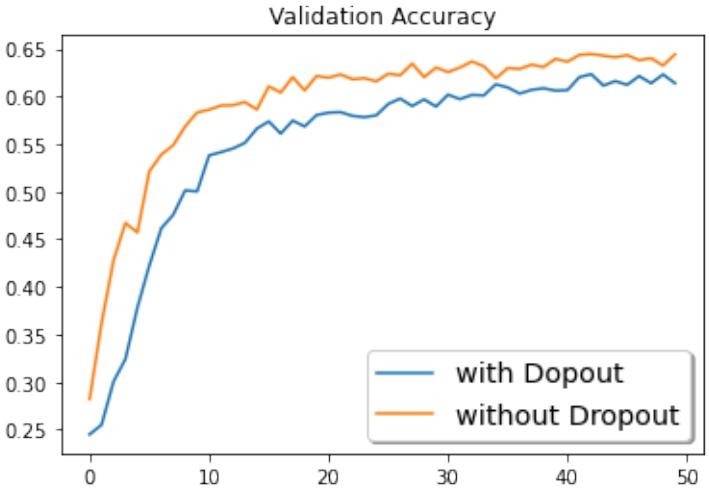

Figure 10: Genauigkeit des Netzes mit und ohne Dropout

Die Outputs des letzten MaxPolling Layers gehen dann in einen Flatten-Layer (*Flatten Layer*, n.d.). Dieser sorgt dafür dass die Outputs zu einem Vector umgeformt werden. Zuletzt folgen dann zwei Dense-Layer (*Dense Layer*, n.d.). Der erste Dient der Erfassung der Features und hat eine breite von 32 Neuronen. Der zweite dient der Klassifizierung. Er besitzt so viele Neuronen wie Emotionen in den Bildern stecken. Mit Hilfe der *SoftMax* Aktivierungsfunktion gibt dieser letzte Layer dann Aufschluss auf die wahrscheinlichste Emotion.

```
model = Sequential()

#conv block 1
model.add(Conv2D(32, kernel_size=(3,3),
                 activation="relu",
                 padding="same",
                 input_shape=trainDataPixel[0].shape,
                 name="conv1_1"))
model.add(Conv2D(32, (3, 3), activation='relu', padding="same", name="conv1_2"))
model.add(MaxPooling2D(pool_size=(2,2), name="pool1_1"))
# conv block 2
model.add(Conv2D(64, kernel_size=(3,3),
                 activation="relu",
                 padding="same",
                 input_shape=trainDataPixel[0].shape,
                 name="conv2_1"))
model.add(Conv2D(64, (3, 3), activation='relu', padding="same", name="conv2_2"))
model.add(MaxPooling2D(pool_size=(2,2), name="pool2_1"))
# conv block 3
model.add(Conv2D(128, kernel_size=(3,3),
                 activation="relu",
                 padding="same",
                 input_shape=trainDataPixel[0].shape,
                 name="conv3_1"))
model.add(Conv2D(128, (3, 3), activation='relu', padding="same", name="conv3_2"))
model.add(MaxPooling2D(pool_size=(2,2), name="pool3_1"))

#identification
model.add(Flatten(name="flatten"))
model.add(Dense(32, activation="relu", name="dense_features"))

# last layer to categories
model.add(Dense(7, activation="softmax", name="dense_classification"))
```

Figure 11: Das Convolutional Neural Network

Listing 11 zeigt das entstandene Netzwerk.

3.3 Verarbeitung Feedback

Auf dem Raspberry Pi läuft im Loop ein Programm das durch Open-CV (*OpenCV*, n.d.) auf die Kamera zugreift. Diese Bibliothek bietet die Möglichkeit einer Objekterkennung bereits fertige Filter für Gesichter. Erscheint nun ein Gesicht vor der Kamera wird der Ausschnitt des Gesichts ausgeschnitten. Anschließend wird das Foto des Gesichts in Graustufen konvertiert und auf die Input-Größe des Neuronalen Netzes skaliert (48x48). Basierend auf der Prediction des Netzes wird danach eine Animation über die LED-Matrix ausgegeben, wenn die wahrscheinlichste Stimmung nicht *Happy* ist.

3.4 Visuelles Feedback

Die LED Matrix die in diesem Projekt verwendet wird, ist eigentlich eine Matrix aus 4 einzelnen Matrizen. Vier miteinander verbundene Platinen auf denen sich jeweils eine 8x8 LED Matrix und ein 8-Digit LED Display Driver (*MAX7219 Datasheet*, n.d.), siehe Abbildung 12.

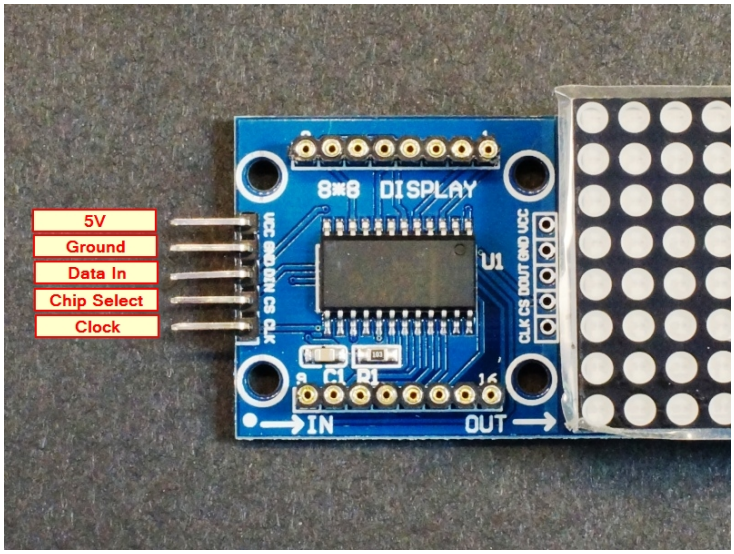

Figure 12: Platine mit abgenommener LED Matrix. Zu sehen ist der LED Display Driver.

Dieser Display Driver lässt sich mit anderen seiner Sorte hintereinander Ketten (Daisy Chain) wodurch über eine SPI-Schnittstelle große Mengen an LED Matrizen gesteuert werden können.

Da in diesem Projekt der Fokus auf dem Neuronalen Netzwerk liegt, wird die Luma Bibliothek (*Luma Library*, 2017) für dieses Projekt verwendet. Diese bietet bereits zahlreiche Funktionen wie `draw.rectangle()`, `scrollText()`, `draw.point()` und `draw.ellipse()`. Mit diesen Funktionen läßt sich ohne großen Aufwand eine Animation erstellen die den Fokus des Benutzenden auf sich zieht.

Die Animation besteht aus der Abfolge:

- Blinkende LED Matrix
- Auf- und Abblenden von Kästen
- Laufender Text:"Watch out!"
- 3 lachende Smileys erscheinen
- Auf- und Abblenden von Elipsen

4 Auswertung

Der *Classification Report* 13 zeigt, dass die Stimmung *Happy* am besten erkannt wird. Alle anderen Stimmungen werden teilweise hingegen deutlich seltener richtig erkannt. Da in diesem Projekt primär die Stimmung *Happy* eine Rolle spielt ist das aber ausreichend.

	precision	recall	f1−score	support
Angry	0.57	0.59	0.58	491
Disgust	0.60	0.55	0.57	55
Fear	0.48	0.38	0.42	528
Happy	0.86	0.90	0.88	879
Sad	0.50	0.58	0.54	594
Surprise	0.77	0.69	0.73	416
Neutral	0.63	0.63	0.63	626
accuracy			0.65	3589
macro avg	0.63	0.62	0.62	3589
weighted avg	0.65	0.65	0.65	3589

Figure 13: Classification Report for Test Data

Die *Confusion Matrix* 14 zeigen wie die Stimmung *Happy* beinah immer richtig erkannt wird, wohingegen *Disgust* sehr oft verwechselt wird.

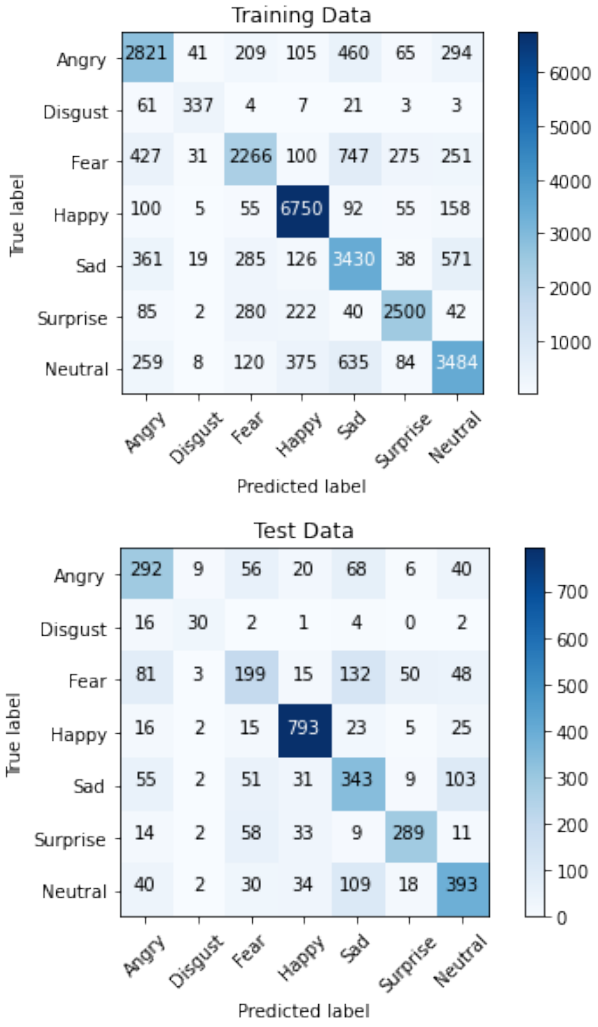

Figure 14: Confusion Matrix des Neuronalen Netzes über die Trainingsbilder und Testbilder

5 Fazit

Das Neuronale Netzwerk liefert für den hier verwendeten Einsatz ausreichende Ergebnisse. Sollten aber in Zukunft auch andere Stimmungen erkannt und gezielt auf sie reagiert werden, so muss das Netzwerk noch verbessert werden. Möglich ist, dass die ungleiche Verteilung von Stimmungen im Datensatz das Netzwerk sich *spezialisiert* hat die Stimmung `Happy` am besten zu erkennen.

References

Dense Layer. (n.d.). https://keras.io/api/layers/core_layers/dense/. ([Online; accessed 28-February-2021])

Dropout Layer. (n.d.). https://keras.io/api/layers/regularization_layers/dropout/. ([Online; accessed 28-February-2021])

FER2013 Dataset. (2018). https://www.kaggle.com/deadskull7/fer2013. ([Online; accessed 28-February-2021])

Flatten Layer. (n.d.). https://keras.io/api/layers/reshaping_layers/flatten/. ([Online; accessed 28-February-2021])

Image data preprocessing. (n.d.). https://keras.io/api/preprocessing/image. ([Online; accessed 28-February-2021])

Luma Library. (2017). https://pypi.org/project/luma.core/. ([Online; accessed 28-February-2021])

MAX7219 Datasheet. (n.d.). https://www.maximintegrated.com/en/products/power/display-power-control/MAX7219.html. ([Online; accessed 28-February-2021])

MaxPooling2D Layer. (n.d.). https://keras.io/api/layers/pooling_layers/max_pooling2d. ([Online; accessed 28-February-2021])

OpenCV. (n.d.). https://opencv.org/. ([Online; accessed 28-February-2021])

OpenCV Cascade Classifier. (n.d.). https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. ([Online; accessed 28-February-2021])

Strack, F., Martin, L., & Stepper, S. (1988, 06). Inhibiting and facilitating conditions of the human smile: A nonobtrusive test of the facial feedback hypothesis. *Journal of personality and social psychology*, *54*, 768-77. DOI: 10.1037/0022-3514.54.5.768