

Facial Emotion Recognition with Convolutional Neural Networks

Ibrahim Alkurdi

Advisor: Prof. Dr. Stephan Pareigis

Hochschule für Angewandte Wissenschaften Hamburg
`ibrahem.alkurdi@haw-hamburg.de`

Abstract. Die Erkennung von Gesichtsemotionen ist in den meisten Bereichen der künstlichen Intelligenz, Videospielen, dem Gesundheitswesen und Marketing ein Thema von großer Bedeutung. Das Ziel dieses Papers ist es Bilder von Menschen in eine von fünf Emotionen zu klassifizieren. Es wurde mit verschiedenen Modellen von konvolutionären neuronalen Netzen (CNN) experimentiert.

CNNs funktionieren mit Bilderkennungsaufgaben deutlich besser als einfache neuronale Netze, da sie wegen der höheren Anzahl von Filtern in der Lage sind, viele wichtige Merkmale der Eingabebilder zu extrahieren. Nachdem mit verschiedenen CNN Modellen mehrmals experimentiert wurde, hat das beste Modell am Ende eine Validierungsgenauigkeit (validation accuracy) von 0.78 und einen Validierungsverlust (validation loss) von 0.64 erreicht.

1 Einführung

Menschen kommunizieren nicht nur durch die Rede miteinander, sondern auch durch Gestik und Mimik. Systeme, die diese drei Arten der Kommunikation simulieren und damit umgehen können, sind in vielen Bereichen von großer Bedeutung. Die Rechner wirken für Menschen natürlicher, wenn sie in der Lage sind Emotionen zu verstehen, zu bilden und anhand dessen mit dem Benutzer zu interagieren. Solche Systeme werden auch bei der Beratung oder bei Bereichen des Gesundheitswesens verwendet. Vor allem kann es während der COVID-19-Pandemie bei dem Fernunterricht sehr hilfreich sein, wenn die Zufriedenheitsbewertung der Teilnehmer während der Konferenz anhand der Erkennung ihrer Emotionen durch die Kameravideos in Echtzeit automatisch geschieht.

Erkennung von Gesichtsemotionen in Echtzeit kann in zahlreichen Bereichen eingesetzt werden. In der Autoindustrie, wenn der Fahrer des Autos wegen seiner Emotionen nicht fahren kann, soll das Selbstfahren die Kontrolle des Autos übernehmen. Die Benutzer von Applikationen oder die, die Produkte testen, geben normalerweise ihr Feedback entweder mündlich oder schriftlich. Aber wenn das Feedback als Gesichtsemotion gegeben werden könnte, würde die Anzahl der Bewerter steigen, da die Bewertung im Vergleich zu der mündlichen oder schriftlichen Bewertung schneller ablaufen kann.

In unserem Zeitalter, welche von Smart-Homes, Robotik, Siri und Alexa, dominiert ist, ist es mittlerweile undenkbar, ohne solch eine Technologie den Alltag zu bewältigen. Es ist eine Technologie, die unsere Gefühle erkennt und uns dann zum Lachen bringt, oder passende Musik zur Stimmung spielt. Dank der künstlichen Intelligenz kann ein Modell die Gesichtsemotionen bis zu 78% richtig erkennen. Dies war vor einigen Jahren noch unmöglich. Es stimmt, dass 78% keine 100% sind, aber selbst ein Mensch kann nicht immer zu 100% die Gefühle eines anderen nur anhand der Gesichtszüge erkennen.

Eine große Herausforderung ist ebenfalls, dass die Datensätze aus Bildern bestehen, die in einer stabilen Umgebung wie einem Labor aufgenommen wurden. Es ist deutlich einfacher in solchen Situationen die Emotionen richtig zu erkennen, da die Modelle bei unkontrollierter Umgebung unzuverlässig arbeiten. Eine andere Herausforderung ist, dass die Datensätze nicht gleichmäßig über alle Klassen verteilt werden. Beispielsweise gibt es für die Klasse Happy viel mehr Bilder als für die Klasse Surprise. Außerdem spielt die Beleuchtung eine große Rolle bei der Erkennung der Gesichtsemotionen. Ein Modell kann bei einer Emotion, die normalerweise richtig erkannt wird, fehlschlagen, wenn die Beleuchtung schlecht ist.

Das Ziel dieses Papers ist es, das bestgeeignetste Model zu suchen, welches die Emotionen in Echtzeit erkennen und richtig klassifizieren kann. Das Modell klassifiziert Menschengesichter in eine von fünf wesentlichen Emotionen (Happy, Sad, Angry, Surprise, Neutral). In vergangenen Jahren wurden mehrere Papers zum Thema "Facial Emotion Recognition" veröffentlicht [8] [12] [11]. Diese Modelle wurden jedoch mit sieben Klassen von Emotionen trainiert.

2 Verwandte Arbeiten

Bei dem Online-Kurs "Facial Expression Recognition with Keras" [5] von Coursera hat Kekre ein Modell mit vier CNN Schichten, zwei vollständig verbundene Schichten und am Ende eine Softmax Schicht vorgeschlagen (siehe Fig. 1). Nach jeder Schicht, außer der Softmax Schicht (Ausgabeschicht), wurde Batch-Normalization und Dropout von 0.25 eingesetzt. Zudem wurde nach jeder CNN Schicht Max-Pooling mit pool_size von (2,2) verwendet, damit die Anzahl der Dimensionen reduziert wird und somit der Grad der Abstraktion steigt. Dropout, Batch-Normalization und Max-Pooling sind im Fig. 1 nicht abgebildet.

Das Modell wurde mit dem FER-2013 Datensatz [2] trainiert und hat am Ende eine 0.70 Genauigkeit erreicht. Die Eingabedaten sind Bilder, die 48*48 Pixel groß sind und in fünf verschiedene Klassen gelabelt wurden. Die Aktivierungsfunktionen der Schichten sind ausschließlich RELU-Aktivierungsfunktion. Mit der Optimierungsfunktion Adam wurde es mit der Lernrate 0.0005 trainiert. Die Kostenfunktion Categorical-Crossentropy wurde eingesetzt.

Im Fig. 2 ist es zu sehen, dass die Genauigkeit sich in den letzten 40 Epochen nicht verbessert hat. Daher könnte man mit dem Training früher aufhören.

Dieses Modell hat Graphics Processing Unit (GPU)-Berechnung genutzt, um den Trainingsprozess zu beschleunigen.

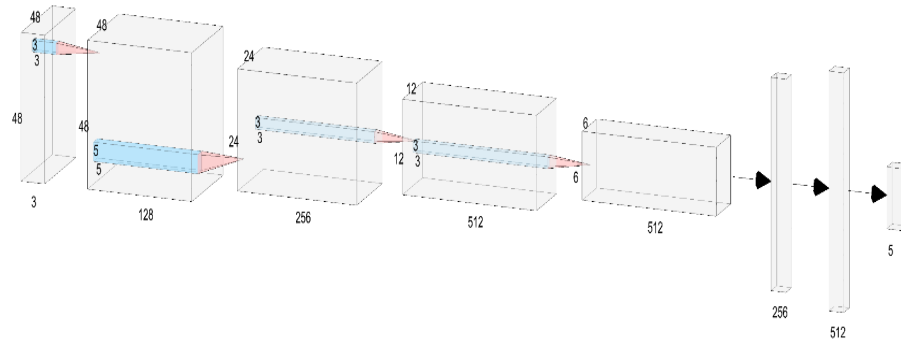


Fig. 1. CNN Model von Kekre erstellt mithilfe von alexlenail Webseite [1]

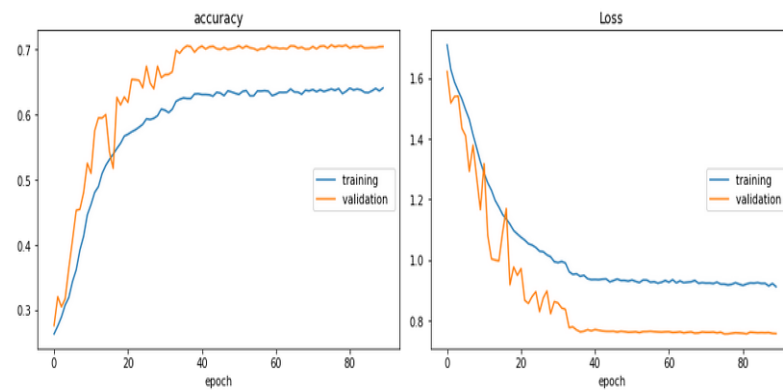


Fig. 2. Genauigkeit und Verlust des Modells von Kekre in 90 Epochen

3 Experimental Setup

Dieser Abschnitt beschreibt die Datensätze, mit denen trainiert und getestet wurde, und wie die Daten vorverarbeitet wurden. Außerdem zeigt es die Bewertung der verschiedenen Modelle.

3.1 Datensatz

Um die Ergebnisse vergleichen zu können, habe ich alle Modelle mit demselben Datensatz trainiert und getestet. Ich habe den Datensatz aus dem Coursera-Online-Kurs "Facial Expression Recognition with Keras" [5] benutzt. Dieser Datensatz ist auf einer Webseite [3] in Kaggle verfügbar. Die Daten stammen aus dem FER-2013 Wettbewerb ("Facial Expression Recognition Challenge 2013") [2]. Diese Daten bestehen aus Graustufenbildern von Gesichtern (siehe Fig. 3). Die Anzahl der Graustufenbilder beträgt insgesamt 30218. Sie sind alle $48 * 48$ Pixel groß. Davon sind 24175 Bilder als Trainingsdaten und die restlichen 6043 Bilder als Validationsdaten. Jedes Bild aus den Validations- und Trainingsdaten muss mit einer der fünf Emotionen kategorisiert sein. Die ursprünglichen Daten enthalten sieben verschiedene Klassen von Emotionen, aber aus Vereinfachungsgründen habe ich mich auf fünf Klassen beschränkt und sie trainiert und getestet. In Fig. 3 sieht man Beispiele dieser fünf Klassen mit der jeweils dazu gehörigen Emotion.

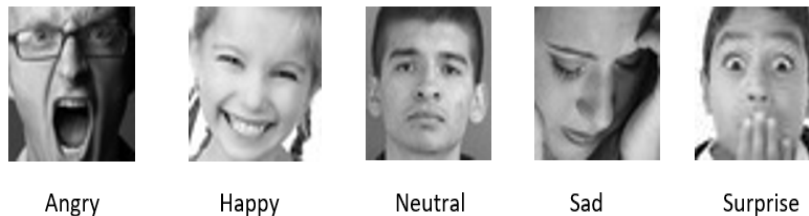


Fig. 3. Beispielbilder aus dem Datensatz, beschriftet mit ihren entsprechenden Emotionen

3.2 Vorverarbeitung der Daten

Bei dem Diagramm 1 (Fig. 4) kann man sehen, dass die Anzahl der Bilder nicht gleichmäßig über die fünf Klassen verteilt wurden. Die Klasse "Happy" hat 7214 und ist damit ca. doppelt so groß wie die Klasse "Surprise". Das könnte unsere Modelle beim Training beeinträchtigen und dazu führen, dass die Modelle die Klasse "Happy" bevorzugen und eher zu dieser Emotion tendieren als anderen.

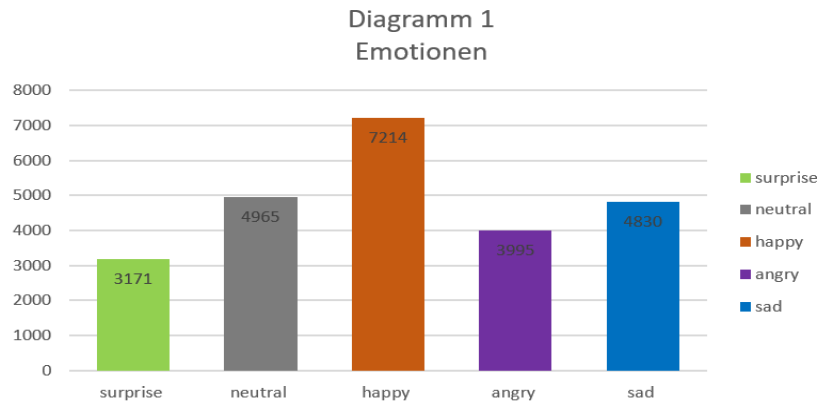


Fig. 4. Anzahl der Bilder in jeder Klasse

Da ich Keras [6] als Framework benutzt habe, hatte ich die Möglichkeit lediglich die Rohbilder der Klassen zu erweitern (Data Augmentation). Mithilfe von Image-Data-Generator kann man aus den Rohbildern gespiegelte Bilder produzieren, indem man den Generator die Bilder horizontal umkippen lässt. Man kann auch mit diesem Generator die Rohbilder rotieren, vergrößern und normalisieren, um mehr Trainingsdaten zu erschaffen.

Ein weiterer Grund der Generator zu nutzen, ist dessen Methode "flow-from-directory". Durch diese Methode ist es nicht mehr nötig, alle Daten auf einmal im Arbeitsspeicher zu platzieren, sondern die Bilder werden stapelweise aus der Festplatte geholt, in den Arbeitsspeicher geschoben und an das neuronale Netz gegeben. Darüber hinaus könnte man die Bilder anders skalieren. Beispielsweise skalierte ich wegen des vortrainiertem Mobilenet Modells die Bilder von $48*48*1$ zu $224*224*3$.

Die Trainingsdaten werden auch gemischt, um die Überanpassung zu vermeiden.

3.3 Hyperparameter

Alle Modelle wurden mit der Aktivierungsfunktion Adam mit der Lernrate von 0,0005 trainiert. Die Aktivierungsfunktion besteht für alle Categorical-Crossentropy. Mithilfe von der Callback-Methode "ReduceLROnPlateau" wird nach 5 Epochen die Lernrate um 0,1 reduziert, wenn der Validationsverlust sich in diesen 5 Epochen nur sehr leicht reduziert hat. Eine sehr wichtige Callback-Methode ist EarlyStopping. Durch diese Methode wird das Training eines Modells früher abgebrochen, wenn es in den letzten 14 Epochen keine Verbesserung im Validationsverlust gab. Wenn das frühere Abbrechen nicht eingesetzt worden wäre, hätte es höchstwahrscheinlich eine Überanpassung gegeben. Wird das Training früher angehalten, was in den meisten Fällen eintrat, werden die Gewichte wiederhergestellt, mit denen der beste Validationsverlust erbracht wurde.

3.4 Beschreibung und Vergleich der Modelle

Ich habe mich hauptsächlich mit drei verschiedenen Modellen beschäftigt. Zwei vortrainierte Modelle (MobileNets [9] und MobileNetV2 [10]) und ein benutzerdefiniertes Modell. Sie wurden alle in Kaggle mit der Benutzung von GPU trainiert und getestet.

Das benutzerdefinierte Modell ähnelt dem, welches beim Coursera-Online-Kurs vorgeschlagen wurde. Es besteht aus vier CNN-Blocks, zwei vollständig verbundene Schichten und am Ende eine Softmax Schicht (siehe Fig. 5). Jeder CNN-Block hat zwei nacheinander geschaltete CNN-Schichten, dann Max-Pooling und Dropout von 0.2. Nach jeder Schicht gibt es BatchNormalization, um die Performance zu verbessern. Allerdings ist Max-Pooling im Fig. 5 nicht abgebildet. Die Aktivierungsfunktionen von allen Schichten sind LeakyReLU. Für dieses Modell müssen die Eingabedaten nicht anders skaliert werden, da wir selbst die Eingabeschicht definieren können. Das heißt, die Größe der Daten bleibt hier $48 \times 48 \times 1$. Am Ende hat das Netz eine Genauigkeit von 0.70 und einen Verlust von 0.76 erreicht (siehe Fig. 6).

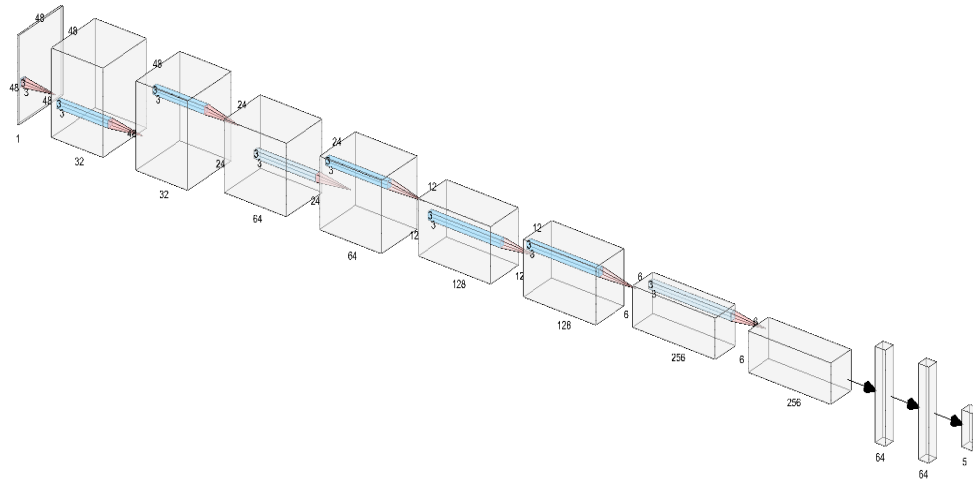


Fig. 5. Das benutzerdefinierte CNN-Modell erstellt mithilfe von alexlenail Webseite [1]

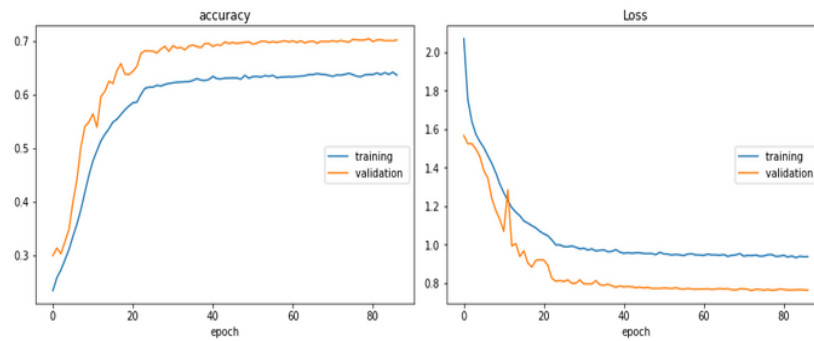


Fig. 6. Genauigkeit und Verlust des benutzerdefinierten Modells in 87 Epochen

Ich habe mit den beiden anderen vortrainierten Mobilenet-Modelle mit verschiedenen Graden von Feintuning experimentiert. Keras hat schon vortrainierte MobileNets und MobliNetV2, die mit ImageNet-Datensatz [4] trainiert wurden. Die folgenden Graden von Feintuning vom vortrainierten MobileNets wurden erfolgreich experimentiert:

1. Die letzten 20 Schichten auf trainierbar schalten und mit drei vollständig verbundenen Schichten (siehe Fig. 7) erweitern, die die Aktivierungsfunktion LeakyReLU haben.
2. Alle Schichten auf trainierbar schalten und mit drei vollständig verbundenen Schichten (siehe Fig. 7) erweitern, die die Aktivierungsfunktion LeakyReLU haben.
3. Genau wie bei 2, aber die Aktivierungsfunktion RELU statt LeakyReLU einsetzen.

Insgesamt hat das MobileNets-Model 87 Schichten. Alle Netze enden mit einer Softmax-Schicht.

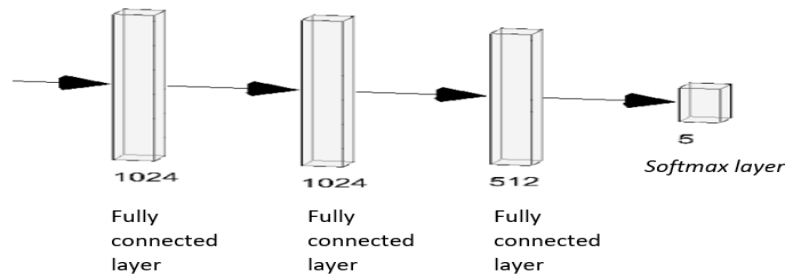


Fig. 7. Drei vollständig verbundenen Schichten gefolgt von Softmax-Layer

Nach Vergleichen der drei Modellen (siehe Table 1) ist das zweite Model von MobileNets mit der Aktivierungsfunktion LeakyReLU das Beste. Die Anzahl der Epochen ist bei den drei Netzen fast identisch.

Table 1. Vergleich von MobileNets-Modellen

Anzahl der trainierbaren Schichten	letzten 20	Alle mit der Aktivierungsfunktion LeakyReLU	Alle mit der Aktivierungsfunktion ReLU
Anzahl der Epochen	31	32	29
Max. Genauigkeit	0.70	0.78	0.77
Min. Verlust	0.80	0.64	0.64

Mit dem vortrainierten MobileNetv2 habe ich weitere Modelle trainiert. Folgende Graden von Feintuning wurden erfolgreich experimentiert:

1. Alle Schichten sind nicht trainierbar.
2. Nur die letzten 50 Schichten sind trainierbar.
3. Die letzten 100 Schichten sind trainierbar.
4. Alle Schichten sind trainierbar.

Alle Modelle sind gefolgt von drei vollständig verbundenen Schichten (wie bei Fig. 7), die die Aktivierungsfunktion LeakyReLU haben, und mit einer Softmax-Schicht enden. Es gibt noch eine fünfte Variante wie bei 4. aber ohne die drei Schichten am Ende.

MobileNetV2 hat insgesamt 155 Schichten, welches als Deep Netz bezeichnet wird.

Nach Vergleichen dieser fünf Varianten (siehe Table 2) hat sich herausgestellt, dass die Varianten vier und fünf gleich gut und die besten sind. Allerdings unterscheiden sich die beiden Varianten von der Anzahl der trainierbaren Parameter. Die fünfte Variante hat am Ende keine vollständig verbundenen Schichten und somit ist die Anzahl der Parameter um ca. 2 Millionen und 900 tausend gesunken. In diesem Fall muss man sich entscheiden, ob die Größe des Netzes oder die Anzahl der Epochen wichtiger ist. Da das Trainieren vom Netzes, das mit drei vollständig verbundenen Schichten endt, um 10 Epochen schneller ist als das ohne.

Table 2. Vergleich von MobileNetV2-Modellen

Anzahl der trainierbaren Schichten	Keine, nur die drei vollständig verbundenen	Nur die letzten 50	Die letzten 100	Alle	Alle ohne die drei vollständig verbundenen
Anzahl der Epochen	54	32	54	32	42
Max. Genauigkeit	0.52	0.58	0.75	0.77	0.77
Min. Verlust	1.17	1.03	0.71	0.67	0.75

3.5 Ergebnisse

Das beste Endergebnis ist das des vortrainierten MobileNets-Models (Fig. 8), dessen drei vollständig verbundenen Schichten die Aktivierungsfunktion Leaky-ReLU haben und dessen Schichten trainierbar sind. Überraschenderweise ist die alte Version von MobileNet in diesem Projekt besser als die neue, wobei die neuere tiefer als die alte ist. Normalerweise erwartet man, dass die tieferen Netze bessere Ergebnisse liefern, da sie in der Lage sind mehr Merkmale aus den Eingabedaten zu extrahieren, speichern und zu erkennen. Ich bin zu der Schlussfolgerung gekommen, dass alle vortrainierten Netze einen Vorteil haben, da sie sehr schnell ein gutes Ergebnis erzeugen, auch wenn sie eventuell neu trainiert werden. Das liegt daran, dass sie schon gewisse Informationen und wichtige Merkmale gespeichert haben, als sie mit vorherigen Datensätzen trainiert wurden. Nicht zu vergessen ist, dass die leichte Überanpassung, die am Ende bei Fig. 8 zu sehen ist, dank des früheren Abbrechens vermieden wird und die besten Gewichte wiederhergestellt werden.

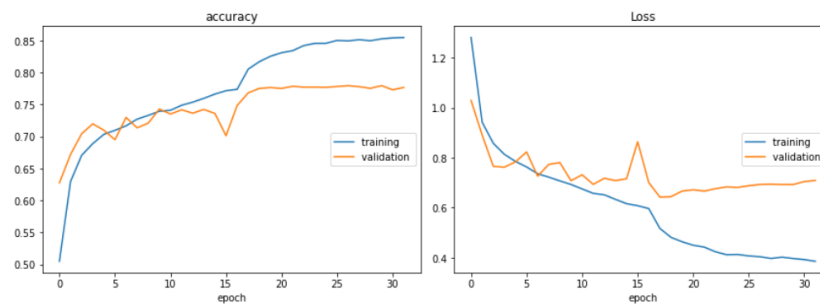


Fig. 8. Genauigkeit und Verlust vom vortrainierten MobileNets-Model, was nochmal neu trainiert und mit drei vollständig verbundenen Schichten erweitert wurde.

4 Das trainierte Netz auf einem eingebetteten Gerät laufen

Mit dem Framework Keras kann man nach dem Trainieren sowohl die Gewichte des Netzes als .h5 Datei speichern als auch die Architektur des Modells. Auf einem Raspberrypi müssen die folgenden Pakete Keras, tensorflow, picamera[array] und opencv installiert werden. Durch picamera kann man auf dem Raspberrypi ein Video Streaming starten. Bildweise werden die Daten aus dem Video genommen und an CV2 weitergegeben, um die Positionen der Gesichter zu ermitteln. Erkennt das CV2 die Gesichter im Bild, werden diese an das gespeicherte Netz gegeben. Das Netz macht anhand der Eingabebilder eine Vorhersage, um die Emotion zu schätzen. Diese Vorhersage wird als eine Beschriftung an das jeweilig dazugehörige Gesicht in dem Video gezeigt (siehe Fig. 9).

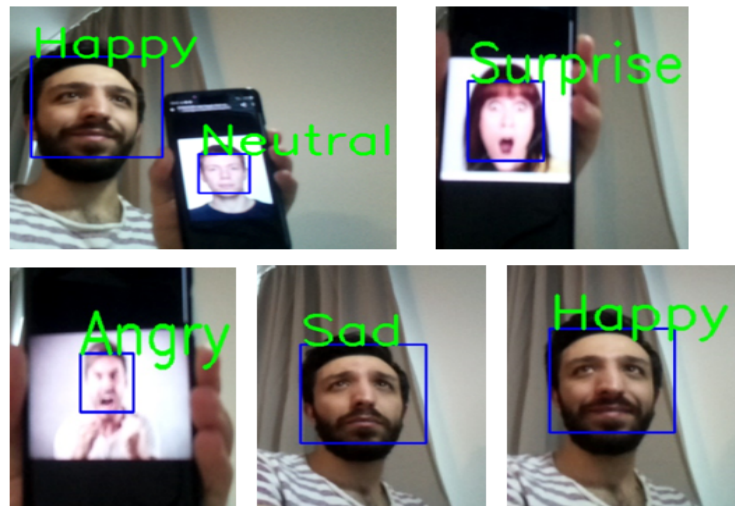


Fig. 9. Beispiele aus dem Video auf Raspberrypi mit Vorhersage der Emotion des trainierten Netzes, hier wurde das beste Modell eingesetzt (MobileNets, die zweite Variante (siehe Table 1))

5 Fazit

In diesem Paper habe ich mich mit der Aufgabe "die Erkennung von Gesichtsemotionen" auseinandergesetzt, um gezielt ein Gesicht in einem Bild in eine von fünf Emotionen zu klassifizieren. Ich habe dabei verschiedene Netze in Kaggle mithilfe von GPU trainiert und verglichen, um das beste Netz zu finden und auf einem eingebetteten Gerät (Raspberrypi in meinem Fall) laufen zu lassen. Mehr als 0.78 Genauigkeit konnte in diesem Projekt nicht erreicht werden, obwohl sehr tiefe Netze wie MobileNetV2 eingesetzt wurden. Man könnte es mit anderen Modellen und Architekturen von neuronalen Netzen weiter versuchen. Meiner Meinung nach ist es unmöglich allein durch Bilder die Gefühle der Menschen mit sehr guter Genauigkeit zu erkennen. Wenn man lächelt bedeutet dies nicht, dass man zu diesem Zeitpunkt glücklich ist.

Der Quellcode zu diesem Projekt ist bei meinem Konto in Github [7] zu erreichen.

Bibliography

- [1] Alexander lenail's publication-ready nn-architecture schematics. <http://alexlenail.me/NN-SVG/AlexNet.html>. Accessed: 2020-07-10.
- [2] Challenges in representation learning: Facial expression recognition challenge. <https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge>. Accessed: 2020-07-10.
- [3] Facial expression recognition challenge dataset. <https://www.kaggle.com/debanga/facial-expression-recognition-challenge>. Accessed: 2020-07-10.
- [4] Imagenet database. <http://www.image-net.org/>. Accessed: 2020-07-10.
- [5] Kekre, s. facial expression recognition with keras. <https://www.coursera.org/projects/facial-expression-recognition-keras>. Accessed: 2020-07-10.
- [6] Keras framework. <https://keras.io/>. Accessed: 2020-07-10.
- [7] Link to source code of this project. <https://github.com/Ibrahemalkurdi/EMLFacialEmotionRecognitionWithCNN>. Accessed: 2020-07-10.
- [8] Alizadeh, S. and Fazel, A. (2017). Convolutional neural networks for facial expression recognition. *CoRR*, abs/1704.06756.
- [9] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.
- [10] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.
- [11] Saravanan, A., Perichetla, G., and Gayathri, K. S., D. (2019). Facial Emotion Recognition using Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1910.05602.
- [12] Yu, Z. and Zhang, C. (2015). Image based static facial expression recognition with multiple deep network learning. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, ICMI '15*, page 435–442, New York, NY, USA. Association for Computing Machinery.

Eigenständigkeitserklärung


Hiermit bestätige ich, dass ich die vorliegende Arbeit (mit den dazugehörigen Teilen wie Code etc.) selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Die Stellen in der Arbeit, die anderen Werken (Bücher, Artikel, Internetquellen etc.) im Wortlaut, in Übersetzung oder dem Sinn nach entnommen wurden, sind als Zitat mit Angaben der Herkunft deutlich kenntlich gemacht. Dies gilt auch für Abbildungen (Diagramme, Fotos etc.) sowie Programmcodefragmente.

Ich bestätige hiermit außerdem, dass ich weder diese Arbeit noch Teile daraus bereits an anderer Stelle eingereicht habe.

Hamburg, den 12.07.2020

Name: Ibrahim Alkurdi



(Unterschrift)