Technische Hochschule Deggendorf, Fakultät für Angewandte Informatik
Bachelor : **Interactive Systeme**

Grundlagen der Künstlichen Intelligenz
Studienarbeit

Thema: Gesichts- und Emotionserkennung

Name: Aboudou Koffitse

Mtr.Nr.: 00774763

Betreuer: Prof. Dr. Andreas Fischer

Abgabedatum: 31. 01. 2021

Inhalt

Abstract

Deep Learning-Algorithmen haben enorm dazu beigetragen, die Performance von Gesichts- und Emotionserkennung zu verbessen. Wir stellen einige Anwendungsbeispiele vor und schildern die typischen neuronalen Netzwerkarchitekturen, die zum Aufbau dieser Systeme eingesetzt werden. In diesem Projekt wenden wir Deep Learning Algorithmen an, um Personen auf Bildern zu erkennen und Emotionen zu identifizieren. Wir werden einen allgemeinen Überblick über den generischen Prozess der Gesichts- und Emotionserkennung geben. Im Folgenden werden wir sehen, wie die Gesichtserkennung in einem Bild funktioniert, welche Methoden es gibt und wie sie in OpenCV und anderen wichtigen Bibliotheken eingesetzt wird. Es wird die Wahl des Algorithmus zur Identifizierung der Person kurz erläutert. Abschließend wird die Implementierung und dessen Ergebnis erläutert.

1- Einführung: Gesichts- und Emotionserkennung

Zunächst ist es wichtig, zwischen dem Gesichtserkennung von Emotionserkennung zu unterscheiden. Gesichtserkennung[1] ist das automatisierte Erkennen eines Gesichts in der Umwelt bzw. in einem Bild (das bereits vorliegt oder zum Zwecke der Gesichtserkennung erzeugt wird) oder das automatisierte Erkennen, Vermessen und Beschreiben von Merkmalen eines Gesichts, um die Identität einer Person ("face recognition") oder deren Geschlecht, Gesundheit, Herkunft, Alter, sexuelle Ausrichtung festzustellen. Emotionserkennung [2] dagegen ist der Prozess der Abstraktion und Klassifikation von audio-visuellen Signalen und ihre Entschlüsselung als Zeichen für Emotionen. Sie findet vor allem in der Emotionspsychologie, der Psychotherapie, der Humanethologie und der Robotik Anwendung. In der Robotik unterscheidet man zwischen visueller Emotionserkennung und akustischer Emotionserkennung... So werden hier der Ausdruck und die Darstellung von Emotionen häufig mit dem Empfinden von Emotionen gleichgesetzt. Folgendes Beispiel soll dies verdeutlichen. Sobald eine Person seine Mundwinkel nach oben zieht oder z. B. den Satz „ich bin fröhlich“ spricht, deutet das die Robotik als Freude. Die Psychologie hingegen würde hier zunächst nur den Ausdruck eines fröhlichen Satzes sehen – aber noch lange keine echte Freude. Damit ein Gesicht sofort erkannt werden kann, arbeiten Gesichtserkennungssysteme mit künstlicher Intelligenz. Mit Deep Learning werden Algorithmen darauf trainiert, menschliche Gesichter aus zahlreichen Fotos und Videos zu erkennen. Viele Unternehmen trainieren ihre neuronalen Netzwerke auf den Milliarden von Gesichtsfotos, die von Instagram, Facebook oder Google im Internet gespeichert werden. Auf Basis dieser Definitionen können wir sagen, dass es Algorithmen zur Gesichts- und Emotionserkennung gibt. Diese beiden Algorithmen haben die gleichen Ziele, unterscheiden sich aber im Detail. Es ist daher wichtig, diese Details zu kennen und wenn möglich die Vorteile des einen zu nutzen, um die Leistung des anderen zu erweitern. Es ist entscheidend, den richtigen Algorithmus für die spezifische Anwendung zu verwenden.

[1].
[2].

2. Stand der Technik

2.1 Gesichtserkennungsfunktion

Ein Gesichtserkennungsprogramm ist eine Softwareanwendung zur Verifizierung einer Person und deren Identifizierung anhand eines Videos oder Bildes aus einer Quelle[3]. Die Gesichtserkennung prüft, ob zwei Gesichter gleich sind. Der Einsatz von Gesichtserkennung ist in den Bereichen Sicherheit, Bio-Metrik, Unterhaltung, Personenschutz usw. enorm. Die **Python-Bibliothek** "face_recognition" bietet eine sehr gute Leistung bei der Erkennung, ob zwei Gesichter miteinander übereinstimmen und gibt das Ergebnis als Wahr oder falsch aus. Die Schritte bei der Gesichtserkennung sind : -Gesichtserkennung - Vergleich mehrerer Gesichter miteinander, um zu erkennen, welche Gesichter zur selben Person gehören.

2.2 Die Verfahren kurz beleuchtet

Das Problem verstehen

In der Geschichte der Wissenschaft wurden viele Algorithmen zur Gesichtserkennung entwickelt, aber ihre Genauigkeit und Geschwindigkeit haben die Erwartungen nicht erfüllt. Erfreulicherweise wurden bedeutende Fortschritte gemacht und scheinen vielversprechend zu sein. CNNs wurden in einer Vielzahl von Computer-Vision-Anwendungen, einschließlich der Facial Expression Recognition (FER), eingesetzt. Anfang des 21. Jahrhunderts wurde in mehreren Studien der FER-Literatur festgestellt, dass CNNs gut bei Änderungen der Gesichtslage sowie bei Variationen des Maßstabs funktionieren. Die Wissenschaftler verwendeten CNN, um verschiedene Probleme der Gesichtsausdruckserkennung zu lösen, wie z. B. Translation, Rotation, Subjektunabhängigkeit und Skaleninvarianz [4].

Mit der Menge an Bildern, die sich im Internet ansammeln, ergriffen Wissenschaftler, die auf dem Gebiet der Computer Vision arbeiten, die Möglichkeit, all diese „Bildern Datenbanken“ zu nutzen, um Bilderkennungsmodelle zu erstellen. Die Leistung der Bilderkennung explodierte mit dem Aufkommen von Deep Learning[4]. Prototypen von Gesichtserkennungs-Apps wurden bereits gebaut und funktionieren sehr gut. So Deep-learning löst das Problem der Gesichtserkennung in realen Überwachungsvideos. Aber es ist festzuhalten, dass das **Convolutional Neural Network** die Hauptrolle in dieser technologischen und wissenschaftlichen Entwicklung gespielt hat. Ein tiefes neuronales Netzwerkmodell ist heutzutage in der Lage, jedes Element einer Szene zu erkennen, sofern es dafür trainiert wurde. Basierend auf der Semantik dieser Erkennung kann er sogar eine „Legende“ zu dieser Szene generieren.

Ein anschauliches Beispiel ist Facebook, wo man mit nur wenigen Trainingsbildern Freunde mit einer Genauigkeit von 95-98% finden und markieren können.
Wie die Deep-Learning-Methode funktioniert, um die Bilderkennung zu erleichtern?
Um es zu verstehen, müssen wir zunächst wissen, was ein Bild für einen Computer ist.

---

[3].
[5].

Für einen Menschen wird ein Bild, ob digital, gedruckt oder auf einem anderen Medium, auf die gleiche Weise wahrgenommen.
Ein Computer weiß nicht, was ein Bild ist .Wir wissen, dass der Computer auf seiner untersten Ebene nur mit Binärwerten, also 0 und 1, umgeht. Der Computer zerlegt ein Bild in Pixel ("Bildelement"), es ist der kleinste Bestandteil eines Bildes, es ist ein einfarbiges Quadrat.
Wenn ein Bild aus genügend Pixeln aufgebaut ist, haben wir den Eindruck, ein zusammenhängendes Bild zu sehen, aber wenn wir auf dieses Bild zoomen, sehen wir die verschiedenen Pixel, aus denen es aufgebaut ist.

Adam Geitgey [6] hat ein Gesichtserkennungssystem entwickelt und hat durch das folgende Konzept gesagt : Die Gesichtserkennung ist eine Reihe von verschiedenen Problemen:
Zuallererst: Der Algorithmus sieht sich ein Foto an und findet alle Gesichter darauf.

Zweitens: Der Algorithmus muss sich auf jedes Gesicht konzentrieren und in der Lage sein, zu verstehen, dass es sich immer um dieselbe Person handelt, auch wenn ein Gesicht in eine andere Richtung oder bei schlechter Beleuchtung gedreht ist.

Drittens : Er muss in der Lage sein, eindeutige Gesichtsmerkmale zu erkennen, anhand derer er sich von anderen Personen unterscheiden kann - wie Augengröße, Gesichtslänge usw. Vergleicht schließlich die einzigartigen Merkmale dieses Gesichts mit denen aller Personen, die er bereits kennt, um den Namen der Person zu bestimmen.

Genau auf dieses Prinzip und Methoden setze ich bei der Umsetzung unseres Gesichtserkennungssystems.

3.Eingesetzte Methoden

3.1 Plattform

Unser Code funktioniert also auf einem klassischen Laptop, nur dank des Prozessors (CPU). Es wurden keine grundlegenden Änderungen am Betriebssystem vorgenommen. Es mussten lediglich viele Pakete oder Programme installiert werden.

Der Installationsprozess für dieses Projekt ist ein wenig mehr als üblich. Zuerst habe ich einen C++-Compiler heruntergeladen. Nach der Installation habe ich es ausgeführt und "Desktop-Entwicklung mit C++" ausgewählt.

Für dieses Projekt habe ich Python 3 verwendet, weil es eine Sprache ist, die perfekt an unsere Bedürfnisse angepasst ist und die über die Werkzeuge zur Verarbeitung der Bilder, der Kamera und der künstlichen Intelligenz verfügt.

Nachdem dieser Schritt abgeschlossen war, ging es weiter mit der Pycharm-Installation. Pycharm ist einfach zu bedienen und ermöglicht einen leichten Zugriff auf den Code. Das Verbinden mit anderen Tools wie Git ist sehr einfach. Ich habe die folgenden Pakete installiert: *Cmake, dlib, Gesichts_Erkennung, numpy, opencv-python.*

Auf der anderen Seite muss ich die folgende 1 GB große Datei herunterladen. Dies ist ein bereits trainiertes Modell der künstlichen Intelligenz, genannt VGG Face, das in der Lage ist, die Vektoren zu bilden, die die Bilder repräsentieren.

Die Vorhersagezeit, um jemanden zu erkennen, beträgt etwa 2 bis 10s, aber da ich "Echtzeit" haben möchte, verwende ich meine Grafikkarte (GPU), um die Berechnungen durchzuführen. Anstatt Tensorflow zu installieren, habe ich also Tensorflow-GPU installiert. So beträgt die Vorhersagezeit etwa 0,1 bis 0,3s.

3.2 Datensatz Allgemein

Die Suche nach einem geeigneten Datensatz erwies sich als schwierig. Zu Beginn wurde daher ein Test mit Bildern aus dem Internet durchgeführt, um die Funktion des Codes zu überprüfen. Um ein signifikantes Ergebnis zu erhalten, müsste jedoch ein ausreichend großer Datensatz gefunden werden. Verschiedene Kriterien sind wichtig, wie z. B. Formatierung, Bildformat. Am Anfang habe ich versucht, Bilder aus dem Internet zu verwenden, ohne deren Größe zu berücksichtigen, ich habe auch mein eigenes Bild verwendet, aber das hat keinen Erfolg gehabt. Schließlich wurde ein vielversprechendes Bild gefunden, das die Kriterien weitgehend erfüllt. Ich habe sogar Fotos von Prominenten wie Angela Merkel im Internet genommen und in PNG konvertiert. In unserem Code und in der Datei finden Sie den Namen dieses Bildes unter (merkel_m.png, merkel1.png, merkel2.png).

Im gleichen Scenario war das Erkennen der Emotion nicht so einfach. Ich habe zuerst versucht, Google Colab zu verwenden. Also ich habe verfügbaren Bilddatenbanken aus Internet geklont. Die erzielten Ergebnisse haben mir nicht zufrieden gestellt. Weil sie nicht genau die Ergebnisse liefern, die ich mir erhoffe. Insbesondere eine Erkennung über die Kamera war schwierig. Dazu habe ich Datensätze auf Gaggle [7] heruntergeladen, die Bildpakete enthalten, mehr als 7800 Bilder. Die Bilder sind in 2 Ordnern gruppiert. Der erste Ordner ist benannt: Train und die zweite ist die Validation. Die beiden enthalten 7 „Emotionen“ Bildordner.

Dieses Mal habe ich mir für Pycharm entschieden. Mit dem Tutorial [8], dem ich im Internet gefolgt habe, und den Links zum Gesichtserkennungs-Tutorial, die uns der Dozent gegeben hat, konnte ich ein mehr oder weniger zufriedenstellendes Ergebnis erzielen.

3.3 KI-Methoden und deren Vorbereitung zum Einsatz

Am Ende der Vorbereitung der technischen Umgebung, begann die Recherche im Internet. Die Themen Gesichts- und Emotionserkennung wurden einzeln betrachtet und auf Unterschiede und Vergleichspunkte der beiden Verfahren geachtet. Im Internet ergaben sich viele Ergebnisse zu den beiden Verfahren, jedoch wenig relevante und entscheidende Aussagen.

3.3.1-Theorie zum Gesichts-und Emotionerkennung Algorithmus.

Für die Implementierung der Gesichts- und Emotionserkennung wurde die Python-Version von OpenCV, einer Open-Source-Computer-Vision-Bibliothek, verwendet, da sie einfach zu installiert ist.

Die Algorithmen zur Gesichts-/Emotion- Körpererkennung sind auf OpenCV [9] basiert und dem Viola-Jones-Objekterkennungs-Framework [10], das ebenfalls auf Haarkaskaden basiert ist. Dabei handelt es sich um einen Ansatz des maschinellen Lernens, bei dem eine Kaskadenfunktion aus einem großen Satz positiver und negativer Bilder trainiert wird. Danach wird diese Funktion verwendet, um Objekte in neuen Bildern zu erkennen. OpenCV enthält sowohl einen „Trainer“ als auch einen Detektor. OpenCV bietet jedoch vordefinierte Klassifikatoren für die Erkennung von häufig verwendeten Objekten, wie z. B. menschliches Gesicht, ganzer Körper, Körper- und Gesichtsteile (bei einigen davon sowohl Vorder- als auch Rückseite). Daher nutzen wir in dieser Arbeit die vorhandenen Klassifikatoren, die von der OpenCV-Bibliothek zur Verfügung gestellt werden, da sie ausreichend waren, um die Anforderungen der implementierten Lösung zu erfüllen.

Als Grundlage der Arbeit mit dem Gesicht Erkennung-Algorithmus diente ein Tutorial im Internet, welches vom Professor vorgeschlagen wurde. Dort wurde ein fertiges Beispiel in Code über GitHub [11] bereitgestellt und einzelnen Code-Fragmente erklärt. Für das Verständnis dieser Arbeit musste sich im Detail mit dem Code beschäftigt werden, sowie Hintergrundwissen zum Verfahren Gesicht/Emotionerkennung erarbeitet werden. Anfangs wurde das bereitgestellte Tutorial durchgearbeitet. Hier mussten viele „Python Pakete“ installiert werden.

Nachdem dies auf den Computer ausgeführt werden konnte, wurde ein Plan zur weiteren Vorgehensweise erstellt. So war es sehr wichtig den vorbereiteten Datensatz in den Code einzuarbeiten. Nach längerem Probieren war es möglich, unser Projekt mit Python zu durchzuführen.

3.3.2- Datensatz Gesichtserkennung

Für unser Projekt haben wir jedoch eine einfachere Methode gewählt. Unser Gesichtserkennungsalgorithmus ist folgendermaßen unterteilt. Zunächst das Bild in Graustufen umwandeln. Dann wird der HOG-Algorithmus (Histogram of Oriented Gradients) zur Vereinfachung des Bildes angewendet. Dieses Bild wird an den Haar-Kaskaden-Algorithmus gesendet, der sich um das Finden der Gesichter kümmert.

Das Gesichtserkennungspaket besteht aus einer Funktion zum Laden von Bildern. Nach dem Import muss das Bild in RGB konvertiert werden. Diese werden dazu in einem bestimmten Verhältnis gemischt. So sind rund 16 Millionen verschiedene Farben möglich. So sieht diese Funktion aus: *face_recognition.load_image_file().*

Wenn wir jedoch mehrere Bilder haben, kann das Importieren der einzelnen Bilder unübersichtlich werden. Wir werden also ein Programm schreiben, das alle Bilder auf einmal in einen bestimmten Ordner importiert. Dazu benötigen wir die *os-Bibliothek(hilft bei der Interaktion mit dem Betriebssystem)*, die ich also zuerst importieren werde.

Gesichter vergleichen und Distanz finden?

Jetzt, nachdem wir die Codes für die verschiedenen Gesichter haben, können wir sie für Gesichter vergleichen, um Ähnlichkeiten zu finden. Um das Paket zu vergleichen, wird einer der gebräuchlichsten maschinellen Lernmethoden, der lineare *SVM-Klassifikator*, verwendet. Wir können die Funktion *compare_faces* verwenden, um festzustellen, ob die Gesichter übereinstimmen. Diese Funktion gibt *True* oder *False* zurück. In ähnlicher Weise können wir die Funktion *face_distance* verwenden, um herauszufinden, wie wahrscheinlich die Übereinstimmung der Gesichter in Bezug auf die Anzahl ist. Dies ist besonders hilfreich, wenn es mehrere Gesichter zu erkennen gibt. Es gab viele Algorithmen zur Gesichtserkennung. Aber diese Methode ist einfacher, in dem Sinne, dass sie das Bild innerhalb des Datensatzes lokal charakterisiert und wenn ein neues unbekanntes Bild auftritt, führen wir einen äquivalenten Algorithmus durch und vergleichen das Ergebnis mit jedem der Bilder innerhalb des Datensatzes[5].

In unserem Beispiel, wenn wir dies mit dem Testbild durchführen, erhalten wir den **Wert True**, was anzeigt, dass das gefundene Gesicht von Merkel ist(Abbildung 1&2). Der Is-Abstand zwischen den Gesichtern beträgt 0,44. Je geringer der Abstand, desto besser die Übereinstimmung.

3.3.3 - Datensatz Emotion Erkennung

Im Vorfeld dieses Teils des Projekts wurden umfangreiche Forschungen über das Convolutional Neuron Network (CNN) [13] durchgeführt.

CNNs sind neuronale Netze, die für räumliche Informationen empfindlich sind. Sie sind in der Lage, komplexe Formen und Muster in einem gegebenen Bild zu erkennen und werden für eine Vielzahl von Bildklassifizierungsalgorithmen verwendet. Wie auch beim Gesicht Erkennung Algorithmus wurde auch hier ein Tutorial vorgeschlagen, welches als Grundlage vom Emotion Erkennung Verfahren gelten soll. Es handelte sich hier um eine „google Colab“ aber auch um „Kaggle“ Datei. Nach langer Arbeit konnte das Tutorial mit dem dort vorgesehen Datensatz ausgeführt werden. Hier bestand die Hauptschwierigkeit darin, die Kamera zu öffnen und die Gesichter mit den verschiedenen Emotionen, die mit google Colab implementiert wurden, zu beobachten. Am Ende machte es keinen Sinn, damit weiterzumachen, da die Forschung mir keine brauchbare Lösung bot. Nach viel Mühe war es mir jedoch möglich, mit Pycharm zu arbeiten. Das hat mir wirklich geholfen. In diesem Teil miener Implementierung versuche ich, ein CNN zu bauen, das in der Lage ist, Sieben Gesichtsemotionen (neutral, Wut, Ekel, Angst, Glück, Traurigkeit und Überraschung) genau zu klassifizieren. Wie bei der Gesichtserkennung sind auch für dieses Projekt einige Bibliotheken zwingend zu installieren. So wurde der in OpenCV verfügbare Local Binary Patterns (LBP) Cascade Classifier zur Gesichtserkennung verwendet, um Gesichter in Bildern zu erkennen. Das CNN für die Emotionserkennung im Gesicht

[5] [12].

wurde mit Keras erstellt. Die in diesem Projekt verwendeten Bilder stammen aus der Kaggle-Datenbank.

4. Versuchsaufbau

4.1 Gesicht Erkennung

Nach Erstellung der Bedingungen konnten Tests mit beiden Methoden durchgeführt werden. Zunächst wurde der Algorithmus zur Gesichtserkennung analysiert. Die Grundidee des Experiments war es, den gleichen Datensatz in der Emotionserkennung zu verwenden, um die Ergebnisse zu vergleichen. Dies konnte jedoch aufgrund von Problemen im Zusammenhang mit der Komplexität des Algorithmus, der Bibliotheken sowie der spezifischen Vorgehensweise in Bezug auf die verwendeten Daten nicht erreicht werden.

Wir haben eine Liste von Bildern und können diese Bilder verwenden, um eine Gesichtserkennungsfunktion zu erstellen. Wir haben nun einfach diese Funktion mit der Liste der Bilder als Eingangsargumente aufgerufen. Mithilfe die "while"-Schleife :

```
while True:
    success, img = cap.read()
    imgS = cv2.resize(img, (0, 0), fx=0.25, fy=0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB).
```

Die Zeitschleife wird erstellt, um die Webcam zu betreiben. Aber um die Schleife auszuführen, haben wir ein Video-Capture-Objekt erstellt, damit wir die Bilder von der Webcam abrufen können (*cap = cv2.VideoCapture(0)*). Zu diesem Zweck versuchen wir zunächst, das Webcam-Bild einzulesen und dann auf ein Viertel seiner Größe zu reduzieren. Dies geschieht, um die Geschwindigkeit des Systems zu erhöhen. Auch wenn das verwendete Bild 1/4 der Originalgröße hat, wird bei der Anzeige immer die Originalgröße verwendet. Dann werde ich es in RGB umwandeln. Und nun konnte die Funktion der Webcams Öffnung kodiert werden.

Sobald wir den Webcam-Rahmen haben, finden wir alle Gesichter auf unserem Bild. Dazu verwenden wir die Funktion :
FacesCurFrame = face_recognition.face_locations(imgS)
encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
Anhand des Indexwertes können wir nun den Namen der Person ermitteln und auf dem Originalbild anzeigen. Und dies ist möglichst mithilfe der *if matches[matchIndex]: name = className[matchIndex].upper().*

Wir können die „datetime-Klasse“ im „datetime-Paket“ verwenden, um die aktuelle Zeit zu erhalten. Dabei wird lediglich geprüft, ob der Abstand zu unserem Gesicht weniger als 0,5 beträgt oder nicht. Wenn dies nicht der Fall ist, bedeutet dies, dass die Person unbekannt ist, so dass wir den Namen in unbekannt ändern und die Anwesenheit nicht markieren.

4.1 Emotion Erkennung

Mit der Erkennung von Emotionen konnte der Test auch in guter Form durchgeführt werden. Hier hat die Videokamera das Auslesen der Emotion ermöglicht und auf dieser Basis können verschiedene Emotionen beobachtet werden. Die erzielten Ergebnisse sind signifikant. Aber ich war nicht in der Lage, die Funktion zu implementieren, die es mir ermöglicht, z. B. den Erkennungsgrad zu vergleichen oder die geeigneten Messwerte und Grafiken zu zeigen. Ich hätte auch den Test gerne mit mehreren Personen gemacht, um herauszufinden, ob das System mehr als ein Gesicht erkennt. Sicherlich zeigt mir ein Test mit meinem eigenen Foto(Abbildung 4 & 5) die Erkennung von 2 Gesichtern mit unterschiedlichen Emotionen. Dies ist bereits ermutigend. Was für mich nicht ermutigend ist, ist, dass die Erkennung der Emotion nicht stabil ist. Es ändert sich schnell.

5. Evaluation und Diskussion der Ergebnisse

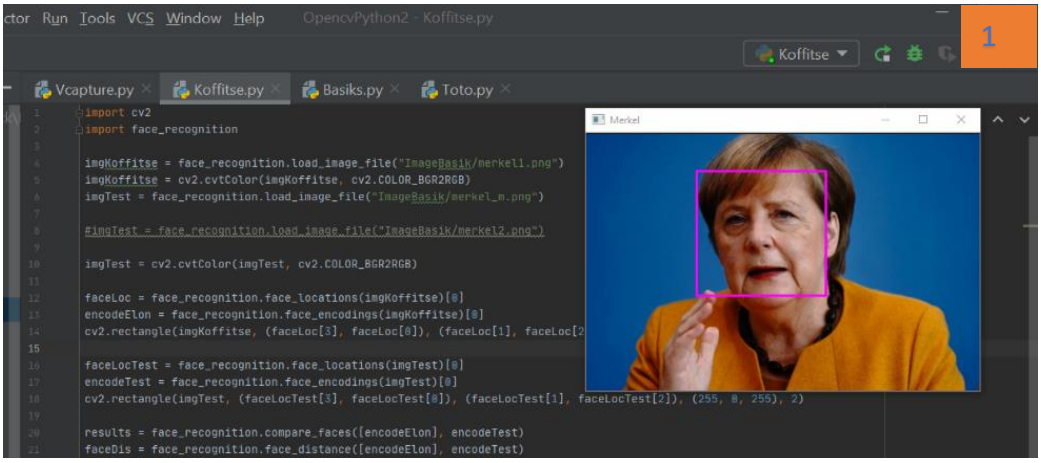
5.1 Gesichtserkennung

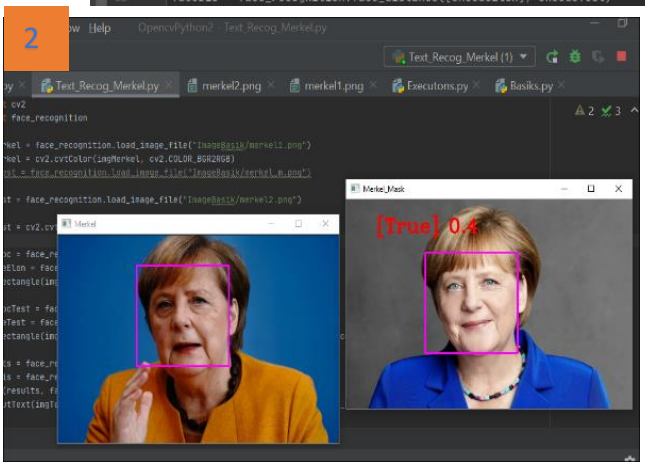
Hier sehen wir, dass der Algorithmus Merkels Gesicht auf dem Foto erkennt.

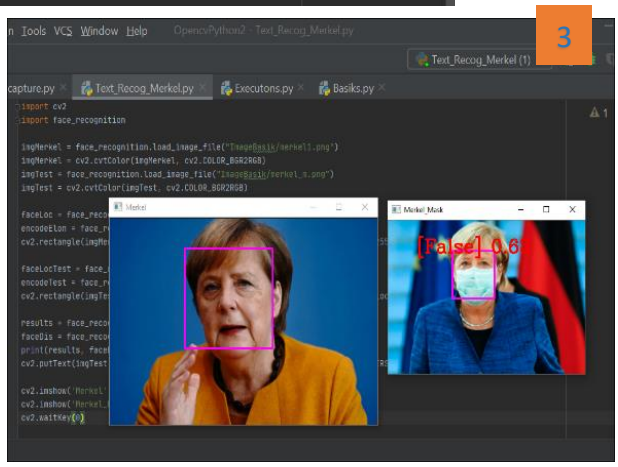
1.Das erste Foto zeigt die Gesichtserkennung.

2.Der zweite erkennt, dass es sich um das Gesicht von Merkel handelt (**True**).

3.Die dritte: Keine Merkel-Gesichtserkennung, also (**False**). Der Punkt ist, dass der Gesichtserkennungsalgorithmus auf bestimmten Teilen des Gesichts basiert, wie Nase, Mund, Augen und Haare... Da Merkels Gesicht maskiert ist, erkennt der Algorithmus nach Durchführung des Bildvergleichs nicht, dass es sich um Merkel handelt. Wenn das Gesicht nicht maskiert war, würde es automatisch erkannt.


1
Koffitse
Vcapture.py
Koffitse.py
Basiks.py
Toto.py
import cv2
import face_recognition
imgKoffitse = face_recognition.load_image_file("ImageBasik/merkel1.png")
imgKoffitse = cv2.cvtColor(imgKoffitse, cv2.COLOR_BGR2RGB)
imgTest = face_recognition.load_image_file("ImageBasik/merkel_m.png")
#imgTest = face_recognition.load_image_file("ImageBasik/merkel2.png")
imgTest = cv2.cvtColor(imgTest, cv2.COLOR_BGR2RGB)
faceLoc = face_recognition.face_locations(imgKoffitse)[0]
encodeElon = face_recognition.face_encodings(imgKoffitse)[0]
cv2.rectangle(imgKoffitse, (faceLoc[3], faceLoc[0]), (faceLoc[1], faceLoc[2
faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest, (faceLocTest[3], faceLocTest[0]), (faceLocTest[1], faceLocTest[2]), (255, 0, 255), 2)
results = face_recognition.compare_faces([encodeElon], encodeTest)
faceDis = face_recognition.face_distance([encodeElon], encodeTest)
Merkel


2
Text_Recog_Merkel (1)
Text_Recog_Merkel.py
merkel2.png
merkel1.png
Executons.py
Basiks.py
Merkel
Merkel_Mask
[True] 0.4


3
Text_Recog_Merkel (1)
Text_Recog_Merkel.py
Executons.py
Basiks.py
Merkel
Merkel Mask
[False] 0.6

5.2 Emotion Erkennung

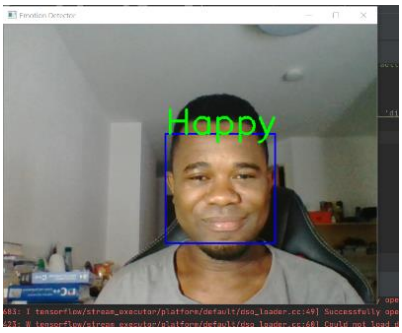


Abbildung 4

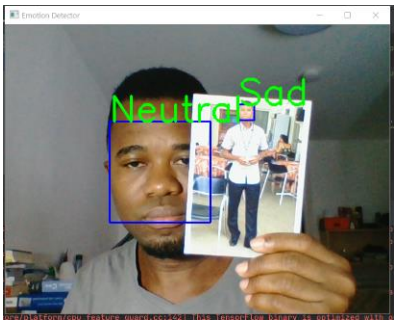


Abbildung 5

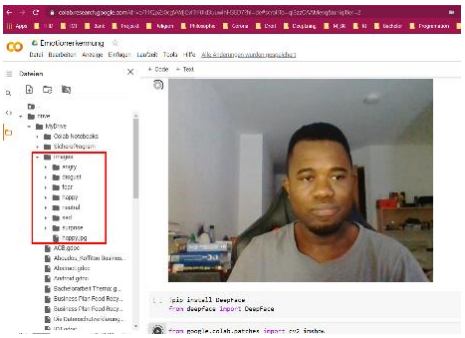


Abbildung 6

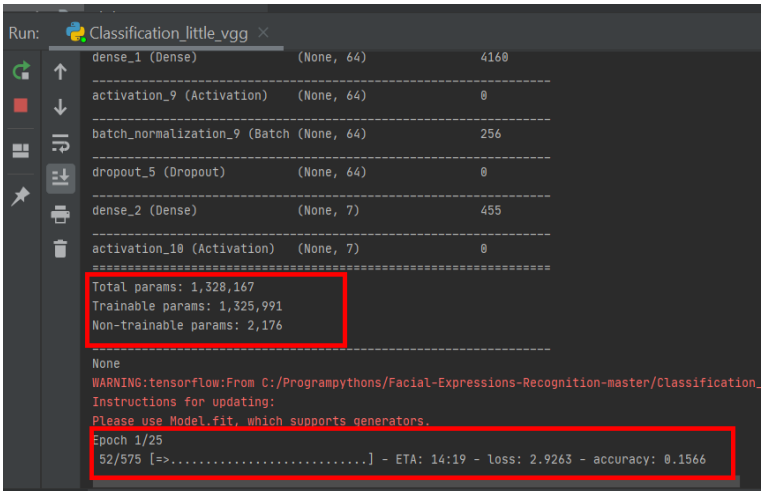


Abbildung 7

Abbildung 4 : Emotion Happy wird gezeigt

Abbildung 5 : Emotion Neutral und Sad in 2 Bilder werden gezeigt

Abbildung 6 : Test mit Google Cobab(Notebook). Kamera öffnet sich aber keine Reaktion

Abbildung 7 : Training Daten werden gezeigt

6.Fazit

Eigentlich war das Ziel, zuerst eine Gesichtserkennung mit der Kamera durchzuführen. Mit den oben beschriebenen Methoden kann unser System durch einfaches Training auf verschiedenen Klassen einzelne Gesichter erkennen und identifizieren. Der Algorithmus lernt automatisch die Eigenschaften des Gesichts und kann so einzelne Gesichter erkennen. Es hat bei mir nicht gut funktioniert, weil ich keine Erkennung durch die Kamera bekommen konnte. Meine Kamera schaltet sich ein, aber wenn ich versuche, sie mit dem Erkennungsfoto zu konfrontieren, schaltet sie sich aus. So stürzte teilweise sogar mein Computer ab. Dies zeigt selbstverständlich ein offensichtliches Problem im Code, das ich auch nach weiteren Tests nicht lösen konnte. Ich hatte eigentlich das Gefühl, dass nach langen Tests das Ergebnis immer das gleiche ist oder dass sich nur das Detail unterscheidet. Darüber hinaus war es aufgrund der großen Datenmenge der Ergebnisse nicht möglich, eine Interpretation zu geben, so dass ich die Verfahren gezielt weiterbearbeiten konnte.

Mit den Ergebnissen speziell im Bereich Emotion Erkennung bin ich sehr zufrieden. Die Vorbereitung beanspruchten sehr viel Zeit und man musste viele Probleme lösen, die an sich nichts mit dem Verfahren zu tun hatten. Jedoch konnte man später sehr gute Ergebnisse mit der Bearbeitung der Parameter erzeugen. In dem Tutorial gibt es einen Part Code. Auf der Basis dieses Codes konnte ich weitere Funktionen schreiben, die zu einem mehr oder weniger perfekten Ergebnis führen können.

Letztendlich konnte ich mir durch diese Arbeit ein sehr gutes Wissen über den Algorithmus Gesichts- und Emotionserkennung erwerben und hatte die Möglichkeit, die Methode z.B. in einem Projekt gezielt anzuwenden. Außerdem habe ich viel über Convolutional Neural Networks (CNNs), Deep Learning und Datensätze gelernt.

Im Netz finden sich zahlreiche Beispiele für die Implementierung von Emotionserkennung. Leider war es für mich nicht einfach,“ Bildpakete“ zu finden. Ich musste meine Suche verstärken und einige Tricks anwenden, um 7 große „Bildpakete“ mit verschiedenen Emotionen zu sammeln. Die Ergebnisse, die ich bei den Versuchen erzielt habe, haben mich dazu motiviert, weiter an dem Thema zu arbeiten. Vor allem, weil das Thema dem Thema meiner Bachelorarbeit, sehr ähnlich ist.

Allerdings sind diese Ergebnisse meiner Meinung nach von geringer Bedeutung. Für die Zukunft wünsche ich mir mehr Forschung, um ein besseres Ergebnis zu erzielen.

Ich selbst habe viel Zeit in dieses Projekt investiert und hätte mir ein noch besseres Ergebnis gewünscht. Das ändert aber nichts an meinem Lerneffekt und der Lust, die „Welt von Deep Learning, CNNs und Gesicht- und Emotion Erkennung...“zu entdecken. Ich konnte mir ein sehr gutes Wissen über beide Algorithmen verschaffen und mich in der Programmierumgebung mit Python noch weiter verbessern. Alles, was ich nach meinen Recherchen zu diesem Thema las, gab mir mehr Trost. Es hat mich gefreut zu sehen, wie aktuell dieses Thema ist und welche vielfältigen Möglichkeiten es in verschiedenen Bereichen bietet.

References

[1] “Gesichtserkennung,” 11/13/2020, https://de.wikipedia.org/w/index.php?title=Gesichtserkennung&oldid=205469892.

[2] “Emotionserkennung,” 4/8/2018, https://de.wikipedia.org/w/index.php?title=Emotionserkennung&oldid=176257424.

[3] M. Khan, S. Chakraborty, R. Astya et al., “Face Detection and Recognition Using OpenCV,” in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 116–119, 2019.

[4] S. Singh and F. Nasoz, “Facial Expression Recognition with Convolutional Neural Networks,” in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 324–328, 2020.

[5] A. Almadhor, “Deep Learning Based Face Detection Algorithm for Mobile Applications,” in *TENCON 2018 - 2018 IEEE Region 10 Conference*, pp. 1158–1162, 2018.

[6] LinkedIn, “Onlinekurs: Deep Learning: Image Recognition | LinkedIn Learning, früher Lynda.com,” 1/30/2021, https://www.linkedin.com/learning/deep-learning-image-recognition.

[7] DataTurks, “Face Detection in Images,” 7/12/2018, https://www.kaggle.com/dataturks/face-detection-in-images.

[8] GitHub, “ageitgey/face_recognition,” 1/30/2021, https://github.com/ageitgey/face_recognition.

[9] F. K. Noble, “Comparison of OpenCV's feature detectors and feature matchers,” in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1–6, 2016.

[10] K. C. Kirana, S. Wibawanto, and H. W. Herwanto, “Redundancy Reduction in Face Detection of Viola-Jones using the Hill Climbing Algorithm,” in *2020 4th International Conference on Vocational Education and Training (ICOVET)*, pp. 139–143, 2020.

[11] GitHub, “ageitgey/face_recognition,” 1/26/2021, https://github.com/ageitgey/face_recognition#face-recognition.

[12] B. Tej Chinimilli, A. T., A. Kotturi et al., “Face Recognition based Attendance System using Haar Cascade and Local Binary Pattern Histogram Algorithm,” in *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pp. 701–704, 2020.

[13] Z. Yan, H. Zhang, R. Piramuthu et al., “HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition,” in *2015 IEEE International Conference on Computer Vision*: *11-18 December 2015, Santiago, Chile : proceedings*, pp. 2740–2748, IEEE, Piscataway, NJ, 2015.