TÜRK-ALMAN ÜNİVERSİTESİ
TÜRKISCH-DEUTSCHE UNIVERSITÄT

Emotionaler Gesichtsausdrücke Erkennen

Facial Emotion Recognition

Türkisch-Deutsche Universität
Fakultät für Ingenieurwissenschaften

TMS_3
Mohamad Eyad Abras
Matrikelnummer: 170501104
E-mail: e170501104@stud.tau.edu.tr

TMS 3
Muhammed Alobayd
Matrikelnummer: 170501109
Email: e170501109@stud.tau.edu.tr

Abgabedastun: 26.06.2022
MEC324- Bildgestützte Automatisierung-II, Dr. Naime Özben ÖNHON, **WiMi:** M.Sc.Onur Akgün

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

FER	Facial Emotion Recognition
MTCCN	Multi-Task Cascaded Convolutional Netzwerk
NN	Neurale Netzwerk
CNN	Convolutional Neurale Netzwerk
ROI	Region of Interest

1. Einführung

Gesichter sind ein unvermeidlicher Bestandteil der zwischenmenschlichen Kommunikation. Es wird allgemein angenommen, dass der emotionale Zustand einer Person leicht aus ihren Gesichtsbewegungen geschlossen werden kann, typischerweise emotionale Ausdrücke oder Gesichtsausdrücke genannt.

1.1 Motivation

Die Emotional von Gesichter können in vielen verschiedenen Formen ausgedrückt werden, die mit bloßem Auge beobachtet werden können oder nicht. beispielsweise begrüßen die Menschen begrüßen sich mit einem Lächeln oder Nicken. Sie haben Tägliche Gespräche von Angesicht zu Angesicht, ob in Person oder über Computer. Sie nehmen Gesichter mit Smartphones und Tablets auf und tauschen Fotos von sich aus und voneinander auf Instagram, Snapchat und anderen Social-Media-Plattformen. Daher war die Gesichtsausdruckerkennung in den letzten Jahrzehnten ein aktives Forschungsgebiet, und es ist immer noch eine Herausforderung aufgrund der hohen Variation innerhalb der Klassen.

Traditionelle Ansätze dafür stützen sich auf handgefertigte Funktionen wie SIFT, HOG und LBP, gefolgt von einem trainierten Klassifikator eine Datenbank mit Bildern oder Videos, aber der Einsatz von Deep Learning und insbesondere Convolutional Neural Networks hat in den letzten Jahren zunehmende Interesse.

Bemerkenswert ist jedoch, dass bei Gesichtsausdrücken Viele der Hinweise stammen aus einigen Bereichen des Gesichts, z. B. Mund und Augen, während andere Teile, wie die Ohren und Haare, spielen in der Ausgabe kleine Rollen. Das bedeutet, dass, Idealerweise sollte sich das Framework für maschinelles Lernen nur auf wichtige Teile des Gesichts konzentrieren und sollte für andere Gesichtsregionen weniger empfindlich sein.

Die Erkennung von Gesichtsausdrücken ist ein wichtiger Studiengegenstand in den Bereichen menschliche Entwicklung, psychologisches Wohlbefinden und soziale Anpassung. Tatsächlich spielt die Emotionserkennung eine zentrale Rolle beim Erleben von Empathie für Roboter. Als beispiel dazu hilft diese Technologie die Robotern, intelligenter mit Menschen zu interagieren und ihre Effektivität und Anpassungsfähigkeit zu verbessern, also die Roboter zu humanisieren. Außerdem kann dies die Ärzte helfen, psychische Störung zu analysieren oder ob ihre Patienten Medikamente benötigt oder die Schwierigkeiten haben, also Patienten besser zu verstehen und behandeln und es gibt noch viel zu viel Anwendungen.

1.2 Ziele

Es wird System für Erkennen emotionaler Gesichtsausdrücke basieren auf Deep-Learning-Algorithmen entwickeln und Diese Arbeit zeichnet sich durch drei zentrale Schritte aus. Zur erst verwenden wir die Haarcascades-Methode, um festzustellen, ob ein Gesicht in den Bildern vorhanden ist, und ob es kein Gesicht gibt, Kehren Sie dann zum Anfang zurück und geben Sie die Bildrahmen ein. Wenn es ein Gesicht gibt, das Vorhandensein von Augen und Mund und Mundbereiche müssen getrimmt werden.denn, Bei einem Gesichtsbild ist klar, dass nicht alle Teile des Gesichts für die Erkennung wichtig sind eine bestimmte Emotion, und in vielen Fällen müssen wir nur auf bestimmte Regionen achten. Darüber hinaus wird es Im Rahmen dieses Projektes erheblich grosser Datensatz mit etwa 20000 Bildern, um Genauigkeit vom System zu erhöhen, Der dritte Schritt ist Gesichtsausdruck-Klassifizierung, für diese Schritt wird es

Deep-Learning-Verfahren verwendet, das auf einem Aufmerksamkeitsfaltungsnetzwerk basiert, um die zugrunde liegende Emotion in Geschichtsbildern zu klassifizieren.

Ein tiefes neuronales Netzwerk beruht auf dem Hinzufügen weiterer Schichten/Neuronen, wodurch der Gradientenfluss erleichtert wird im Netzwerk (z. B. durch Hinzufügen von Skip-Layern) oder bessere Regularisierungen (z. B. spektrale Normalisierung), insbesondere bei Klassifikationsproblemen mit einer großen Anzahl von Klassen. Jedoch, für die Gesichtsausdruckerkennung zeigen wir aufgrund der geringen Anzahl von Klassen, dass die Verwendung von ein Faltungsnetzwerk mit weniger als 8 Schichten ist in der Lage, bessere Ergebnisse zu erzielen , In diesem Projekt werden wir verschiedene Methode untersuchen um die effiziente , schnellste und hochpräzise System zur Erkennung der Gesichter zu entwickeln. und am Ende natürlich werden wir ganze System mit unterschiedlichen Eingabe oder Bilder testen.

2. Stand der Technik

Gesichtsemotionen Erkennung ist eine in Software verwendete Technik, Die es einem Programm ermöglicht Emotionen auf einem menschlichen Gesicht mit fortschrittlicher Bildverarbeitung zu „lesen“.

Große Unternehmen haben damit experimentiert, ausgeklügelte Algorithmen mit Bildverarbeitungstechniken zu kombinieren um besser zu verstehen, was uns ein Bild oder ein Video des Gesichts einer Person über ihre Gefühle verrät.

Um die menschliche Gesichtaausdrucke zu erkennen, es gibt drei Hauptschritte:

1. Gesichtserkennung
2. Merkmale Extraktion
3. Gesichtsausdruck-Klassifizierung

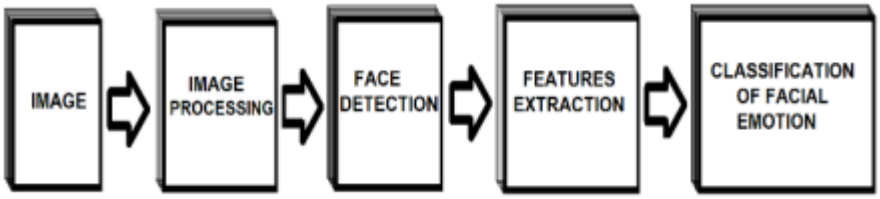


Abbildung 1: *Schritte der Gesichtsemotionen Erkennung*

Gesichtserkennung

Die Gesichtserkennung ist die Aufgabe, ein oder mehrere Gesichter in einem digitalen Bild oder Video zu lokalisieren. Die bestehenden Techniken zur Detektion des Gesichts in einem Bild können grob in vier Kategorien eingeteilt werden:

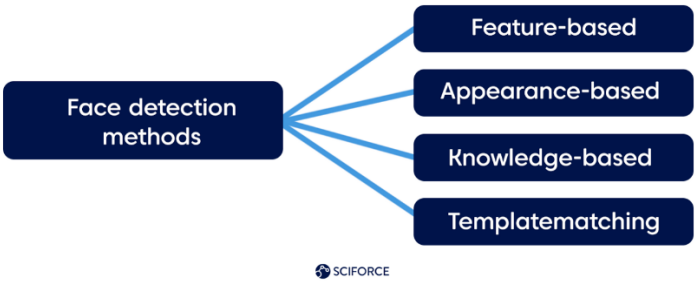


Abbildung 2: *Klassifikation von Gesichtsdetektion*

(a) Templatematching Methode
(b) Aussehensbasierte Methode (Appearance based Methode)
(c) Wissensbasierte Methode (Knowledge-Based Methode):

Diese Methode stützt sich auf das Regelwerk, das der Mensch nach unserem Wissen entwickelt hat. Wir wissen, dass ein Gesicht Nase, Augen und Mund in bestimmten Abständen und Positionen zueinander haben muss.

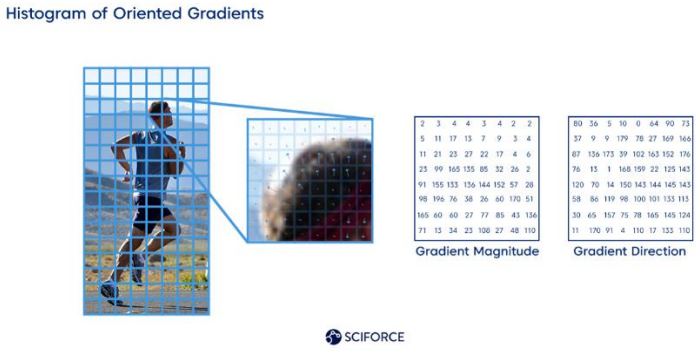


Abbildung 3: *Beispiel des Funkionsweise von HOG*

(d) Merkmalsbasierte Methode (Feature based Methode):

Das merkmalsbasierte Verfahren extrahiert strukturelle Merkmale des Gesichts. Es wird als Klassifikator trainiert und dann verwendet, um Gesichts- und Nicht-Gesichtsregionen zu unterscheiden. Als Beispiele dieses Method:

- Haarcascade Allgorithmus
- Multi-Task Cascaded Convolutional Networks

Haarcascade Allgorithmus

Haarcascade Algorithmus ist die maschinelle Lernmethode, bei der sich ein Klassifikator befindet aus vielen positiven und negativen Fotos gebohrt, also Es ist ein Objekterkennungsalgorithmus, der verwendet wird, um Gesichter in einem Bild oder einem Echtzeitvideo zu identifizieren. . Der Algorithmus wird von Paul Viola und Michael Jones vorgelegt und Dieses Algorithmus werden viele positive Bilder gegeben, die aus Gesichtern bestehen, und viele negative Bilder, die nicht aus einem Gesicht bestehen, um darauf zu

trainieren. Das ausdiesem Training erstellte Modell ist im OpenCV GitHub-Repository verfügbar Allgemein kann man diese verfahren kann in vier Stufen erklärt werden.

1. Haar-Feature-Auswahl

Der erste Schritt besteht darin, die Haar-Merkmale zu sammeln. Ein Haar-ähnliches Merkmal besteht aus dunklen Regionen und hellen Regionen, die kontinuierlich von oben links im Bild nach unten rechts verschieben werden, um nach dem bestimmten Merkmal zu suchen.

Es erzeugt einen einzigen Wert, indem es die Differenz der Summe der Intensitäten der dunklen Bereiche und der Summe der Intensitäten der hellen Bereiche bildet, wie in der Abbildung deutlich dargestellt.

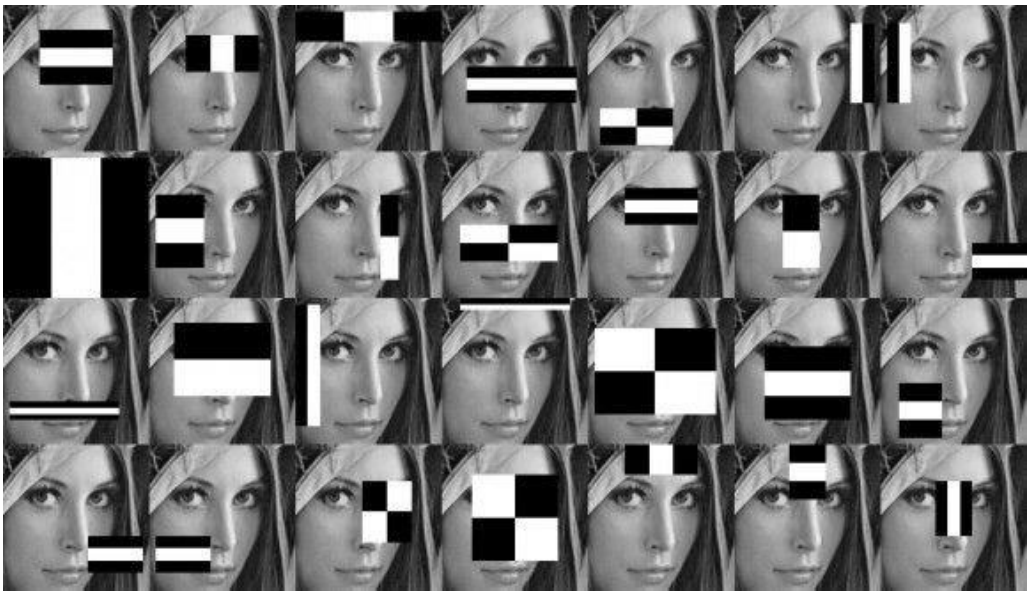


Abbildung 4: *Haar Merkmale Auswahl*

2. Erstellung integraler Bilder

Das Berechnen der Rechteck-Features in einem Convolutional Kernel Stil kann rechenaufwendig sein. Aus diesem Grund haben die Autoren Viola und Jones eine Zwischendarstellung für das Bild vorgeschlagen, ziemlich Erstellung integrale Bilder. man kann einen integralen Bildansatz verwenden, um eine O(1)-Laufzeit zu erreichen. Ein bestimmter Pixel im integralen Bild ist die Summe aller Pixel auf der linken Seite und aller Pixel darüber. Mit den anderen worten, Die Summe aller violetten Kästchen im Originalbild ist gleich der Summe der grünen Kästchen im integralen Bild minus (subtrahiert) von den violetten Kästchen im integralen Bild.wie in der Abbildung deutlich dargestellt

64	2	3	61	60	6	7	57
9	55	54	12	13	51	50	16
17	47	46	20	21	43	43	24
40	26	27	37	36	30	31	33
32	34	35	29	28	38	39	25
41	23	22	44	45	19	18	48
49	15	14	52	53	11	10	56
8	58	59	5	4	62	63	1

Original Image

64	66	69	130	190	196	203	260
73	130	187	260	333	390	446	520
90	194	297	390	484	584	683	780
130	260	390	520	650	780	910	1040
162	326	491	650	808	976	1145	1300
203	390	577	780	983	1170	1357	1560
252	454	655	910	1166	1364	1561	1820
260	520	780	1040	1300	1560	1820	2080

Integral Image

Abbildung 5: *Integraler Bilder Verfahren*

3. AdaBoost Training

Für ein Fenster mit 24 x 24 Pixeln kann es etwa 162.336 mögliche Merkmale geben, deren Auswertung sehr teuer wäre. Daher wird der AdaBoost-Algorithmus verwendet, um den Klassifikator nur mit den besten Merkmalen zu trainieren, denn Adaboost verwendet eine Kombination aus „schwachen Klassifikatoren“, um einen „starken Klassifikator“ zu erstellen, den der Algorithmus verwenden kann, um Objekte schnell zu erkennen.

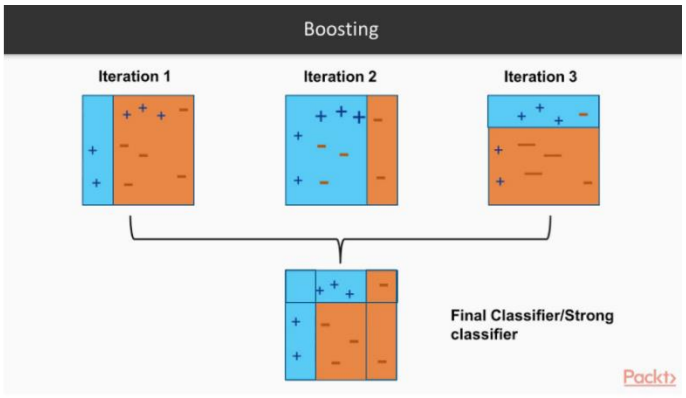


Abbildung 6: *Adaboost Training Verfahren*

4. Cascade_klassifizierer-Architektur

Es ist eine Methode zum Kombinieren von immer komplexeren Klassifikatoren wie AdaBoost in einer Cascade. Es ermöglicht, negative Eingaben (keine Gesichter) schnell zu verwerfen, während mehr Berechnungen für vielversprechende oder positive gesichtsähnliche Regionen aufgewendet werden. mit einem anderen worten, die Hauptaufgabe besteht darin, Unterfenster abzulehnen, die keine Gesichter

enthalten, während Regionen identifiziert werden, die Gesichter enthalten. Da die Aufgabe darin besteht, das Gesicht richtig zu identifizieren. Es reduziert die Rechenzeit erheblich und macht den Prozess effizienter.

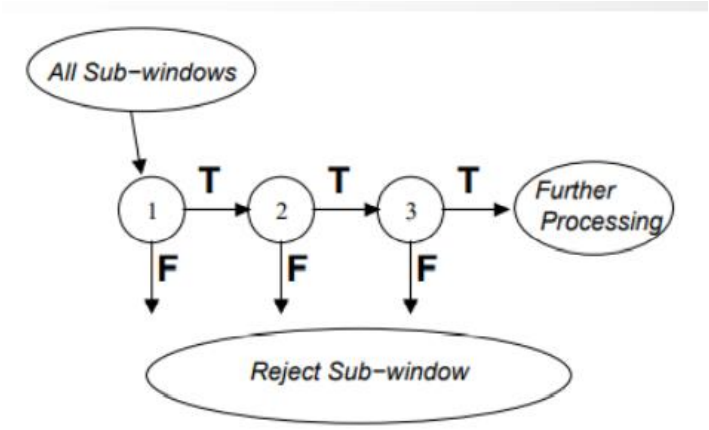


Abbildung 7: *Kaskadenklassifizierer-Architektur*

MTCCN

MTCCN ist ein tief kaskadiertes Multi-Task-Framework, das die inhärente Korrelation zwischen Erkennung und Ausrichtung nutzt, um ihre Leistung zu steigern. Diese Verfahren besteht im Wesentlichen aus drei CNN mit unterschiedlichen Strukturen zusammen für die Gesichtserkennung, ziemlich Proposal Network (P-Net),) ein Refine Network (R-Net), O-Net (Output)

Jedoch bevor wir das Testbild in den Klassifikator einfügen, müssen wir die Größe des Bildes in verschiedenen Maßstäben ändern und es in einer Bildpyramide stapeln. Durch diese Schritte können wir dasselbe Gesicht in verschiedenen Maßstäben erzeugen, was die Leistungsfähigkeit des Netzwerks erhöht. Danach wird ein gleitendes Fenster angewendet zur Pyramide, und brechen Sie das Bild in Regionen, die sind die Eingänge zum Netzwerk. Also verschiedene Kopien desselben Bildes in unterschiedlichen Größen erstellen, Um innerhalb des Bildes nach Gesichtern unterschiedlicher Größe zu suchen.

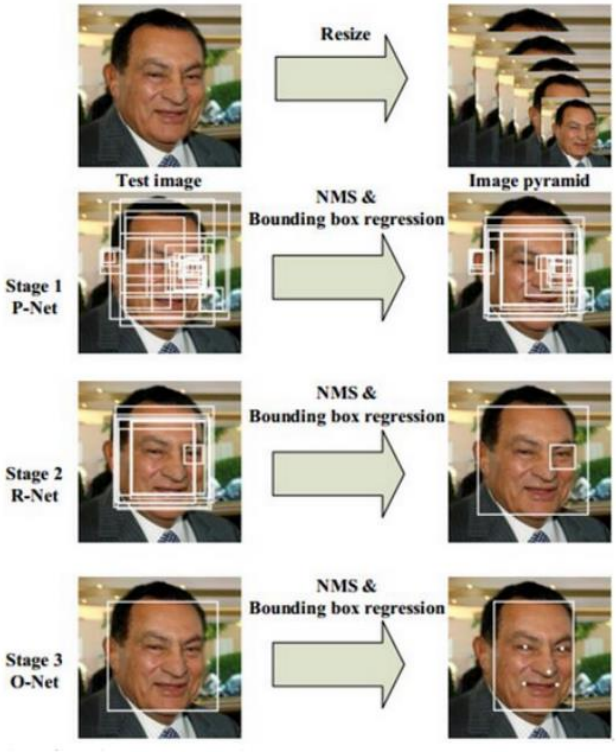


Abbildung 8: *MTCNN-Netzwerkstruktur (Multitask Cascade Convolutional Neural Network), einschließlich P-Net, R-Net und O-Net*

- (P-Net)

Wir nutzen ein vollständig gefaltetes Netzwerk, genannt Proposal Network (P-Net), um das Gesicht des Kandidaten zu erhalten Fenster und ihre Bounding-Box-Regressionsvektoren. Dann Kandidaten werden basierend auf den geschätzten Bounding-Box-Regressionsvektoren kalibriert.. Für jede vorgeschlagene Begrenzung Box, es werden drei verschiedene Klassifikationen darauf angewendet, Dies sind Gesichtsklassifizierung, Bounding-Box-Regression und Lokalisierung von Gesichtsmerkmalen.

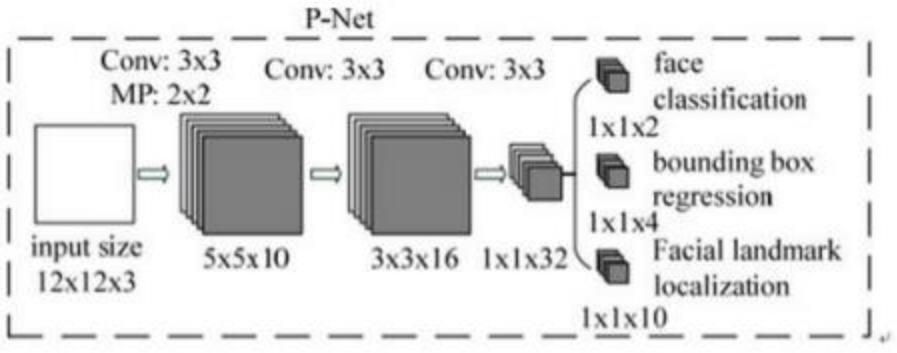


Abbildung 9: *P-Net (Proposal)*

- R-Net (Refine):

Dieses Netz hat eine Faltung mit größerem Kernel und einer vollständig verbundenen Schicht, die leistungsfähiger ist als (P-Net).

Das Hauptziel dieses Netzes ist es, die Ergebnisse des vorherigen Netzes (P-Net) zu verfeinern. Diese Netzwerkstruktur eliminiert hauptsächlich Fehlalarme Proben durch Boundary-Box-Regression und Nicht-Maximum Unterdrückung, und führt immer noch nur Gesichtserkennung und Boundary-Box-Regressionsaufgaben.

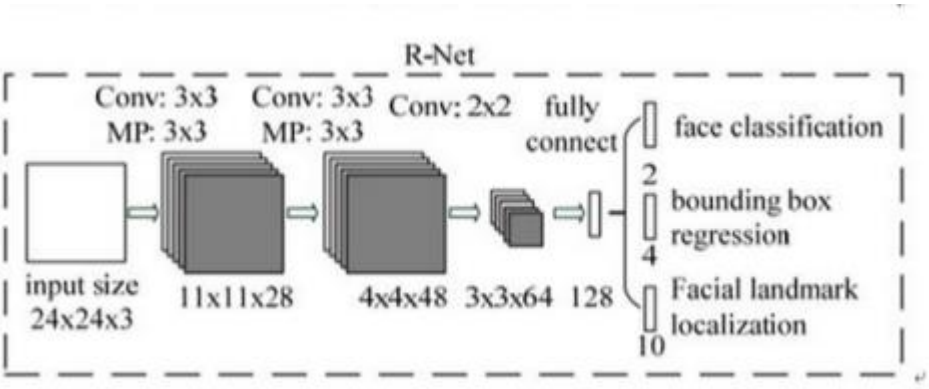


Abbildung 10: *R-Net (Refine)*

- O-Net (Output)

Dieses Netzwerk ist tiefer mit einem größeren Faltungskern im Vergleich zu früheren Netzen und dies passt die Größe der Eingabebild auf 150 × 150 und berechnet die Position des Merkmals Punkte auf jedem Gesichtsrahmen.

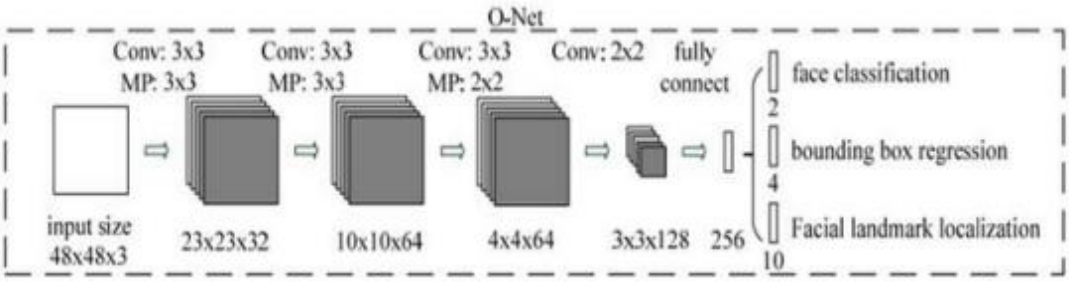


Abbildung 11: *O-Net (Refine)*

Merkmale Extraktion

Die Merkmalsextraktion ist ein Teil des Dimensionsreduktionsprozesses, bei dem ein anfänglicher Satz von Rohdaten aufgeteilt und auf besser handhabbare Gruppen reduziert wird. das hilft dabei, das beste Feature aus großen Datensätzen herauszuholen, indem Variablen ausgewählt und zu Features kombiniert werden, wodurch die Datenmenge effektiv reduziert wird. es vereinfacht die Verarbeitung, und ist in der Lage, den eigentlichen Datensatz mit Genauigkeit und Originalität zu beschreiben.

Warum man die Dimension die Datensatz reduzieret, da mit diesem Prozess kann man die Genauigkeitsrate verbessern, Die Risiko von Überanpassung verringern. Außer beschleunigt die Training und hat eine verbesserte Datenvisualisierung.

Abbildung 12: *Gesicht merkmlae region*

Gesichtsausdrucke Klassifizierung

Es ist der letzte und wichtigste Schritt, bei dem Gesichtsausdrücke mithilfe biometrischer Indikatoren erkennt werden. Emotionale Ausdrücke anhand von Gesichtsmerkmalen (Auge, Mund, Nase ...uzw) definieren. Die Klassifizierung erfolgt gemäß den Ähnlichkeiten zwischen den Gesichtsmerkmalen nach dem Sammeln und Analysieren von Informationen aus Bildern.

Eine Rolle eines CNN besteht darin, Bilder in eine Form zu bringen, die einfacher zu verarbeiten ist, ohne dass Merkmale verloren gehen, die für eine gute Vorhersage entscheidend sind

- CNN

Erstens NN sind eine Teilmenge des maschinellen Lernens und bilden das Herzstück von Deep-Learning-Algorithmen. Der Name und ihre Struktur sind vom menschlichen Gehirn inspiriert bestehen aus:

- Eingabenschicht: Es liefert Informationen (Features) von der Außenwelt an das Netzwerk (an die verborgene Schicht).
- Verborgene Schicht: Führt alle Arten von Berechnungen an den über die Eingabeschicht eingegebenen Features durch und überträgt das Ergebnis an die Ausgabeschicht.
- Ausgabenschicht: Diese Schicht bringt die vom Netzwerk gelernten Informationen nach außen

Also ein CNN ist ein Deep-Learning-Algorithmus, der am häufigsten zur Analyse visueller Bilder verwendet wird. Das haupte Ziel ist für die Bild zu erkennen und -klassifizieren. Es hat Anwendungen auch in Bild- und Videoerkennung, Beratungssystemen, medizinischer Bildanalyse uzw. Erforderliche Vorverarbeitung ist viel geringer als bei anderen Klassifizierungsalgorithmen.

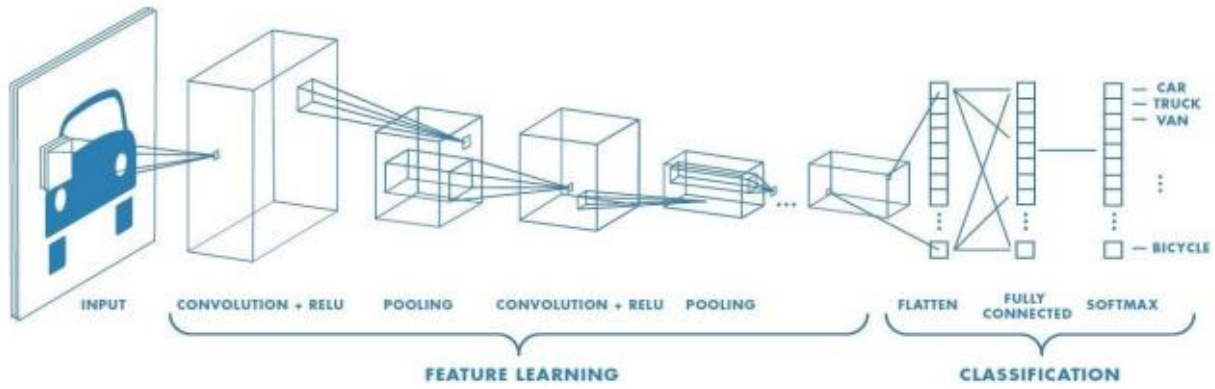


Abbildung 13: *CNN Konzept*

CNN hat viele Operation hilft bei Merkmal des Gesichts zu extrahieren und sie zu klassifizieren, die Operationen sind:

- Convolution Operationen :

Das Ziel ist High-Level-Merkmale, wie Kanten, aus einem Eingabebild mit hilfe von mehreren Faltungsschichte zu extrahieren.

  - Die Funktionen der Faltungsschicht:
    - Die erste(n) Faltungsschicht(en) lernen Merkmale wie Kanten, Farbe, und einfache Texturen.
    - Die nächste(n) Faltungsschicht(en) lernen Merkmale, die komplexen Texturen sind.
    - Die letzte(n) Faltungsschicht(en) lernen Merkmale wie Objekte oder Teile von Objekten.

Es hat auch ein Kernel, der alles filtert, was für die Feature-Map nicht wichtig ist, und konzentriert sich nur auf bestimmte Informationen. Der Filter bewegt sich mit einer bestimmten Schrittlänge nach rechts, bis er die gesamte Breite geparst hat. Dann geht es mit der gleichen Schrittlänge zurück zur linken Seite des Bildes und wiederholt den Vorgang, bis das gesamte Bild durchquert ist.

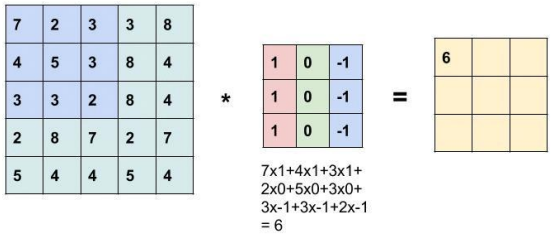


Abbildung 14: *Convolution Operationen*

- Pooling Operationen

Die Pooling-Layer reduziert die räumliche Größe eines gefalteten Features. Es gibt zwei Arten von Pooling, nämlich Max Pooling und Average Pooling. Max Pooling gibt den Maximalwert aus dem Teil des Bildes zurück, der vom Kernel abgedeckt wird, während Average Pooling den Durchschnitt der entsprechenden Werte zurückgibt.

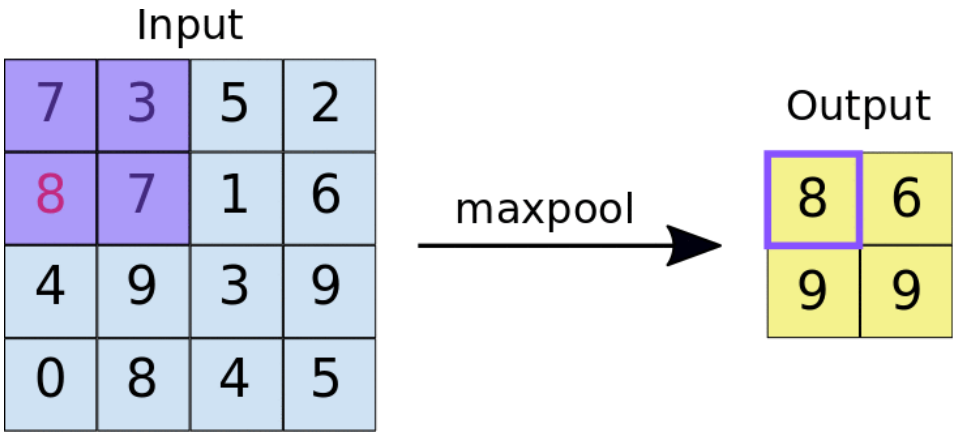


Abbildung 15: *Pooling Operationen*

- Vollständig verbundene Schicht :

Neuronen in dieser Schicht haben Verbindungen zu allen Neuronen in der vorherigen Schicht Diese Schicht befindet sich am Ende eines CNN, Die Eingabe aus der vorherigen Schicht in einen eindimensionalen Vektor abgeplatt Eine Aktivierungsfunktion angewendet, um die Ausgabe zu erhalten.

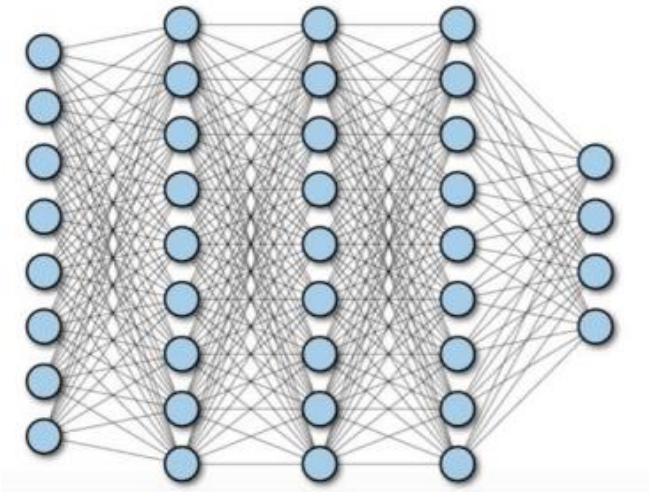


Abbildung 16: *Vollständig verbundene Schicht*

- Dropout

Dropout wird verwendet, um eine Überanpassung zu vermeiden, durch Ignorierung von zufälligen Neuronen während des Trainings.

Sie während eines bestimmten Vorwärts- oder Rückwärtsdurchgangs nicht berücksichtigt werden

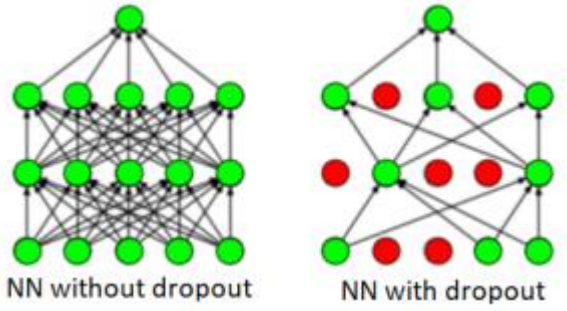


Abbildung 17: *Dropout Beispiel*

- Aktivierungsfunktion

Der Zweck ist Nichtlinearitäten in das Netzwerk einzuführen, da die meisten realen Daten nichtlinear sind. Die Verwendung einer nichtlinearen Funktion ermöglicht es NNs auch, komplexe Funktionen zu approximieren, dadurch sie in der Lage ist, komplexere Aufgaben zu lernen und auszuführen

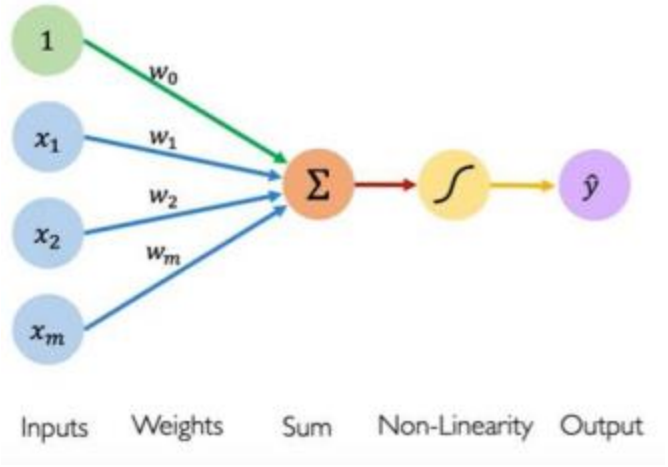


Abbildung 18: *NN Struktur*

3. Arbeitspakete und Gantt Chart

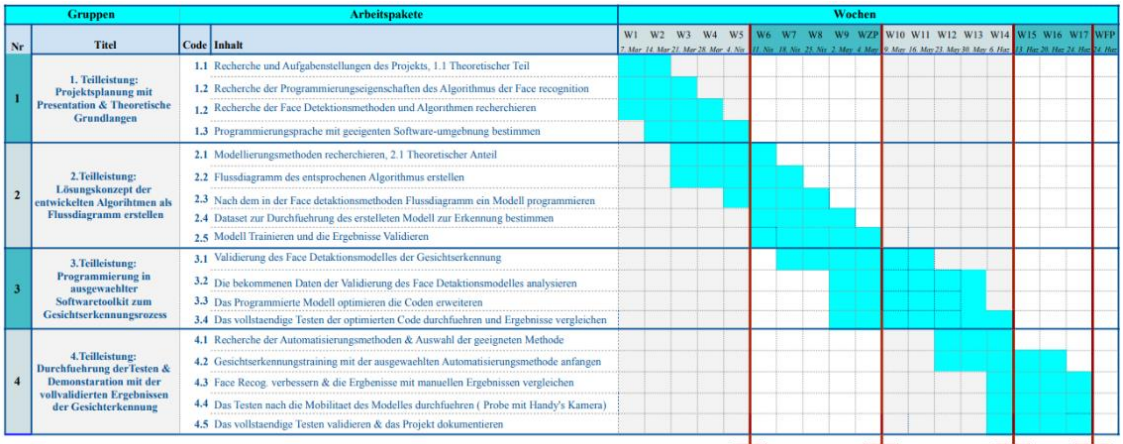
3.1 Arbeitspakete

Arbeitspakete	Input	Funktion	Output	Verantwortlicher	Abgabetermin
1. Theoretischer Teil 1.1 Theoretische Grundlagen Die Herausforderungen des Themas	der Literatur, Code recherchieren, z.B. Google Scholar, Github, Kaggle	Recherche sowie die Überprüfung verschiedener Forschung, Studien und Projekte. Auf diesen Quellen basierend die **Aufgabestellungen bestimmen.** Studien, die zu diesem Thema geeignet sind,, werden untersucht, wie prezis und effizient die Aufgabe mit den gleichenAlgorithmus Struktur erfüllt worden sind,.	Die Aufgaben mit höherer Realisierungswahrscheinlichkeit bestimmen.	Alle Mitglieder	14.03.2022
2. Theoretischer Teil 2.1 Nach Stand der Technik recherchieren	Face Detektion Methoden und Algorithmen recherchieren	Literature Recherche nach die **Funktionlität der Gesichtserkennung**, was gibt es als **Gesichtserkennungstechnologie** suchen	Ausblick über die Anwendungen und funktionsweise haben	Alle Mitglieder	21.03.2022
2.2 Face detektion und erkennung funktionsweise recherchieren	Gesichtserkennung Methoden und Algorithmen recherchieren	Nach der **Methoden** und angewendete **Algorithmen** tiefe recherchen machen, die **moderne Modelle** verstehen	schnelle Algorithmen bestimmen und höhe Genaue Methode auswählen	Alle Mitglieder	28.03.2022
1.2.Theoretischer Teil. Lösungstrategie zur Realisierung klaeren 1.3 Die möglichst geeignete Software tool untersuchen	der Literatur nach der Software tool recherchieren, z.B. Google Scholar, Github, Kaggle	**Software-toolkit,** Durch die Untersuchung der Eigenschaften spezifizierte Aufgaben wird es entschieden,was für die Programmierungssprache und mit welcher Software umgebung das Projekt realisiert wird.	**Programmierungsprache mit geeigenten Software-umgebnung bestimmen**, die zur bestimmten Aufgaben geeignet werden.	Alle Mitglieder	04.04.2022

Arbeitspakete	Input	Funktion	Output	Verantwortlicher	Abgabetermin
3.Implementierung 3.1 Den **Flussdiagramm** des Algorithmus und das Modell erstellen	gewünschte Algorithmus	Nach Bestimmung der Algorithmus auf der geeignete Software-umgebung wird **das grundlegende Modell des Projekts** erstellen sowie Programmieren	**Gesicht detektion Modell**	Alle Mitglieder	11.04.2022
3.2 **Datensatz suchen** und bestimmen	Von Google, gethub oder eigene private Bilder nutzen	nach wir geeignete datensatz gefünden haben werden wir unsere Modell auf **einfache Art und Weise vollständig vorbereiten**	saubere und vorbereitete Datensatz	Alle Mitglieder	18.04.2022
3.3 **Modell Trainieren und die Ergebnisse bekommen**	ausgewählte Bilder basierend	**Das Modell mit Bildern der verschiedenen Personen wird manuell trainiert**	**Das vollständige trainierte Modell**	Alle Mitglieder	25.04.2022
3.4 **Optimierung das Modell**	kontrollieren die Trainingsphase	kontrollieren ob es einige mögliche Fehler durch die Programmierung oder das Trainieren und Optimierung sowie **Entwicklung das Modell**	ausgewählte und **geeignete Modell**	Alle Mitglieder	16.05.2022

Arbeitspakete	Input	Funktion	Output	Verantwortlicher	Abgabetermin
4.Verbesserung des Modelles mit dem Handy verbindend	Nach der Praktische Anwendung das Face erkennungmodell mobiler machen	**Die Coden optimieren,** indem **das Handy's Kamera fuer live video basierende Gesichtserkennung** benutzt wird. Dadurch wird die hohere Erkennungsqaulitaet des Modelles auch erheoht, mit dem guten Kamera	**Das Handy's Kamera fuer live** video basierende Gesichtserkennung wird **mit dem Modell verbunden**	Alle Mitglieder	30.05.2022
das ganz Projekt testen und optomieren und die Dokomente vorbreiten	verschieden Bildern **automatische tranieren**	nach wir alle Zielle erreicht haben werden wir mehr Bilder automatisch mit **zusaetlichen ausseren Softwaretoolkit tranieren, um die Funktionprinzip schneller und unfehlerhaft und mit der vollstandig Erfuellung der Aufgabe zu** testen	**Fehlerfrei Ergebnisse bekommen** und die Dekomente ist vollstandig vorbreitet	Alle Mitglieder	06.06.2022
Das Projekt dokumentieren & die Ergebnisse presentieren	**Die Modellergebnisse vergleiched validieren**	Das vollstaedig funktionierende Modell mit den Ergebissen vergleichend live demonstrienen. **Validiereung der Ergebnisse** der automatisierten Gesichtserkennung **mit Teachable Machine**	Die **Finale Presentation** mit die **fertigte Dokumentation** zur Verfuegung stellen	Alle Mitglieder	24.06.2022

3.2 Gantt Chart und Meilensteine



M1 - Theoretische Grundlagen & Programmierungsprache mit Softwaretoolkit bestimmen

M2 - Erstellung des Modelles & Trainieren und die Ergebnisse Validieren

M3 -Das vollstaendige Testen der optimierten Code mit Ergebnisse vergleichen, das Modell zur Automatisierung breit stellen

M4- Validierung der Ergebnisse der automatisierten Face recognition & Ergebnisse zur Verfuegung stellen

4. Arbeitperizip

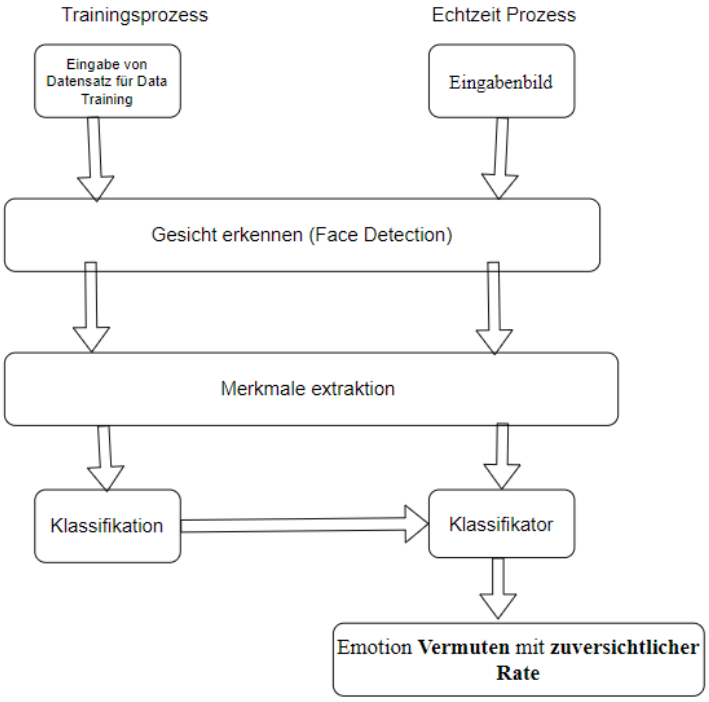


Abbildung 19: *Flussdiagramm des Arbeitperinzip*

Im Rahmen dieses Projekt haben wir eine erbliche große Datensatz, die mehr als 4000 Bilder mit der guten Qualität enthaltet, um erhoher ergebnisse zu erreichen, danach haben wir dieses Datensatz trainiert. unser Projekt funktioniert wie folgenden:

Es wird als Eingabe (Bild) angegeben, danach wird Gesicht mit hilfe Haar Cascade Algorithmus erkennen und dann wird die wichtigste Merkmale ziemlich die Gesicht extrahieren und diese Merkmale wird mit dem trainierten Datensatz vergleichen. Am Ende wird das System Emotion Vermuten.

Es gibt verschieden Verfahren, um die Genauigkeit von System zu verbessern, als beispiel dazu haben wir mehr als 99. Epoch verwendet. Wir haben „Colab“ verwendet, denn Es ermöglicht, beliebigen Python-Code über den Browser zu schreiben und auszuführen, Außerdem ist es besonders geeignet für maschinelles Lernen, Datenanalyse und zusätzlich ist es kostenlos nutzbar.

5. Implementierung

In diesem Unterkapitel werden wir sehen, wie das FER-System funktioniert. Wie wir im zweiten Unterkapitel erwähnt haben, konzentrieren wir uns auf 3 Aufgaben, um das gesamte System zu vervollständigen.

Die Aufgaben sind: Entwicklung einer Gesichtserkennung mit maschinellem Lernalgorithmus, Suche nach geeignetem Datensatz und Erstellung und Trainierung eines Modells mit CNN.

5.1 Gesichtserkennung-Aufgabe

Wir haben Haar Cascades in OpenCV und Python verwendet, um das Gesicht oder das vordere Gesicht zu erkennen. Zunächst bestimmt das entwickelte Gesichtserkennungsmodell den interessierenden Bereich, dieser „ROI“ sagt uns, wo sich das Gesicht im Bild befindet. Jetzt zeigen wir das Kode:

- Die Bibliotheken

```
import cv2
import numpy as np
import pandas as pd
from cv2 import imread
from google.colab.patches import cv2_imshow
from google.colab import files
from cv2 import rectangle
```

Zelle 01

- Das HauptKode

```
face_cascade= cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_frontalface_alt2.xml'))
pixels = cv2.imread('Image_Path ')
bboxes = face_cascade.detectMultiScale(pixels)
for box in bboxes:
    print(box)
```

Zelle 02

Die Ausgabe der Zelle 02 ist die Koordinate des ROI. Wir möchten ein Rechteck über die Pixeln durch verwendung die erhaltene Koordinat

```
for box in bboxes:

    x, y, width, height = box
    x2, y2 = x + width, y + height

    rectangle(pixels, (x, y), (x2, y2), (0,0,255), 4)

cv2_imshow(pixels),
print( 'number of detected faces :' ,len(bboxes))
```

Zelle 03

Die Ausgabe der Zelle 03 zeigt uns wo das Menschliche Gesicht in dem Eingabebild, das von dem Rechteck umgeben ist.

5.2 Datensatz Suche-Aufgabe

In unserer Recherche haben wir uns zum Ziel gesetzt, den am besten geeigneten Datensatz zu finden, der unserem Systemzweck dient.

Am Ende fanden wir nach stundenlanger Recherche einen Datensatz, der von NICK NGUYEN DAI TRUONG THANH gesammelt wurde. Dieser Datensatz enthält insgesamt mehr als 35.000 Fotos für 6 verschiedene Emotionsklassen. Wir haben 20.000 Fotos davon ausgewählt, die zu 3 Gefühlsklassen gehören, diese ausgewählten Gefühlsklassen sind: „Angry“, „Happy“ und „Sad“. Die Bilder des Datensatzes wurden in Training, Validierung und Testen unterteilt.

	Angst	*Happy*	*Sad*
Trainieren	3995	7215	4830
Testen	958	1774	1247

5.3 Erstellung und Trainierung des Modells- Aufgabe

Zur Erstellung des Modells haben wir CNN angewendet. Mit CNN können wir die merkmale des Gesichts extrahieren danack klassifizieren. Außerdem zur Trainierung des Modells haben wir die Bibliotheken Tenserflow und Keras benutzt. Zur Erfüllung dieser Aufgabe, sollten wir:

- Die Bibliotheken

```
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras import Model
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import GlobalMaxPooling2D, MaxPooling2D
from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activatio
n,MaxPooling2D
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.optimizers import Adam,SGD,RMSprop
from tensorflow.keras.models import Model
from tensorflow.keras.models import load_model
import os
from os import listdir
```

Zelle 04

- Datenaufbereitung

wir vorbereiten die Daten, mit den das Modell/der Klassifikator trainiert wird.

```
train_dir = 'Training_file Path'
test_dir = 'Testing_file Path'

train_angry_dir = 'Angry _file Path'
train_happy_dir = 'Happy_file Path'
train_sad_dir = 'Sad_file Path"

test_angry_dir = 'Angry_file Path"
test_happy_dir = 'Happy_file Path"
test_sad_dir = 'Sad_file Path"

dir_list = [train_angry_dir, train_happy_dir,train_sad_dir,
        test_angry_dir, test_happy_dir, test_sad_dir]

for d in dir_list:
    print(d, len(os.listdir(d)))
```

Zelle 05

- Datenerweiterung

```
train_datagen = ImageDataGenerator(rescale=1.0/255.0,
                    rotation_range=2.5,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    zoom_range=0.1,
                    horizontal_flip=True,
                    fill_mode='nearest')

train_generator = train_datagen.flow_from_directory(train_dir,
                                target_size=(150,150),
                                batch_size=128,
                                class_mode='categorical')

validation_datagen = ImageDataGenerator(rescale=1.0/255.0)

validation_generator = validation_datagen.flow_from_directory(test_dir,
                                        target_size=(150,150),
                                        batch_size=128,
                                        class_mode='categorical')
```

Zelle 06

- Aufbau des Modells

```
model = tf.keras.models.Sequential([

    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])


model.compile(optimizer=RMSprop(learning_rate=0.001 ),
              loss='categorical_crossentropy',
              metrics = ['accuracy'])

model.summary()
```

Zelle 07

- Trainieren des Modells

```
path = r"Modul_Path"

checkpoint = ModelCheckpoint(path, monitor = "val_loss",
                             mode = "min" ,
                             save_best_only = True , verbose = 1)
earlystop = EarlyStopping (monitor = "val_loss" , min_delta = 0 ,
                           patience = 5 , restore_best_weights = True ,
                           verbose = 1)
callbacks = [earlystop , checkpoint]
history = model.fit(train_generator,
                    epochs=100,
                    verbose=1,
                    validation_data=validation_generator,
                    callbacks=callbacks)

#speichern wir die beste Ergebnisse des Traianierens
model.save(path)
```

Zelle 08

- Plotten der Trainierensverlauf

Um den Modelltrainingsverlauf zu visualisieren, können wir Diagramme(Plot) aus den gesammelten Verlaufsdaten erstellen. Die Diagramme können uns einen Hinweis auf nützliche Informationen über das Training des Modells geben, wie zum Beispiel:

- Die Geschwindigkeit der Konvergenz über Epochen (Steigung).
- Ob das Modul möglicherweise bereits konvergiert ist (Plateau der Linie).
- Ob das Modul die Trainingsdaten möglicherweise überlernt (Wendepunkt für Validierungslinie).

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

# accuracy

plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'b--', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.show()

# loss

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'r--', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```

Zelle 09

6. Auswertung und Evaluierung

In diesem Unterkapitel diskutieren wir die Ergebnisse bzw. Die Ausgabe jedes oben gezeigten Zelle, aucg die Falsche oder die unerwartete Ausgabe und die möglische Lösungen bevor wir die ganzen System testen.

- Probleme im Datensatz

Wir haben einige Probleme mit unserem Datensatz festgestellt, die ein Problem durch die Trainingsphase verursachen könnten. diese sind:

- **Ungleichgewichtsproblem**

Einige Gesichtsausdrucke Klasse hat mehr Daten als die Anderen. Als Lösung dieses Problem ist die Daten zu erweitern.

- **Klasseninterne Variation**

In mancher Klasse beinhaltet animated Bilder oder Text Bilder. Die Problemslösung ist die Überanpassung zu vermeiden

- **Abdeckung** (mit Hände)
- **Kontrastvariation**

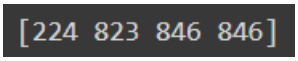
Manche Bilder sind dunkler als die andere Bilder inerhalb einer Klasse

- **Sonnenbrille**

- Die Ausgabe von Zelle 02:

Wir haben schon erwähnt dass das Ergebnisse dieser Zelle ein Koordinate von ROI ist. Als Beispiel haben wir Abbildung(a) als Eingabebild des System und wir möchten das Gesicht detektieren.

(a)



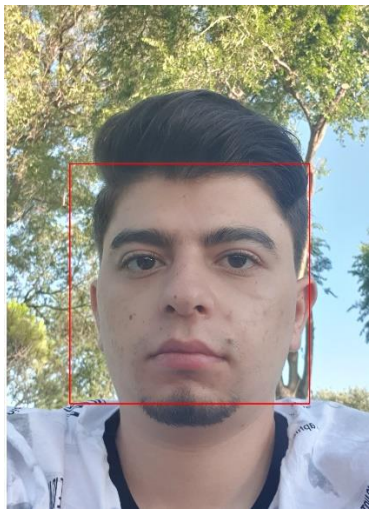
(b)

Als Ausgabe bekommen wir die Koordinat (b), die wie später benutzen werden können um das Gesicht zu detektieren.

- Die Ausgabe vo Zelle 03:

Wir nehmen jetzt die Koordinate (b), um ein Rechteck um den Gesichtsbereich zu zeichnen. Außerdem können wir die Farbe und die Dicke des Rechtecks ändern.

(b)



(c)

Erwartungsgemäß erhalten wir die gewünschte Ausgabe(c).

- Problem:

Manchmal erhalten wir mehr als eine Gesichtsbereichskoordinate im Eingabebild, aber es ist sicher, dass es nur ein Gesicht gibt. In diesem Fall sollten wir die Dimensionen des Eingabebilds überprüfen, je großer ist, desto nedriger ist die Detektionsgenaugikeit, weshalb erhalten wir falche Ausgabe. Um dieses Problem

zu lösen, sollten wir also Dimension des Eingabebilds ändern, das wird durch die Größenänderungsfunktion (Zelle 10).

```
def resize(img):
print('Original Dimensions : ',img.shape)
scale_percent = 60
width = int(img.shape[1] * scale_percent / 300)
height = int(img.shape[0] * scale_percent / 300)
dim = (width, height)

resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)

print('Resized Dimensions : ',resized.shape)

return resized
```

Zelle 10

- Die Ausgabe von Zelle 05& Zelle 06:

Die Ausgaben dieser Zellen sind für Überprüfung wie viel Trainierensklassen und Testsklassen haben wir und wie viel Bildern hat jeder klasse.

```
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Training/Training/Angry 3995
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Training/Training/Happy 7215
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Training/Training/Sad 4830
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Testing/Testing/Angry 958
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Testing/Testing/Happy 1774
/content/drive/MyDrive/jdc/Bildgestutzte/Dataset/Testing/Testing/Sad 1247
Found 16040 images belonging to 3 classes.
Found 3979 images belonging to 3 classes.
```

- Die Ausgabe von Zelle 07:

In dieser Zelle erstellten wir das Klassifier mit 4 CNN-layer für die Gesicht- Merkmale zu extrahieren und ein Flatten-layer um die extrahierte Merkmale in Merkmale Vektor zu wandeln und 5 NN-layer um die Gesichtsausdrucke zu klassifizieren.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 64)      1792

max_pooling2d (MaxPooling2D  (None, 74, 74, 64)        0
)

conv2d_1 (Conv2D)            (None, 72, 72, 128)       73856

max_pooling2d_1 (MaxPooling  (None, 36, 36, 128)       0
2D)

conv2d_2 (Conv2D)            (None, 34, 34, 128)       147584

max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)       0
2D)

conv2d_3 (Conv2D)            (None, 15, 15, 64)        73792

max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)          0
2D)

flatten (Flatten)            (None, 3136)              0

dense (Dense)                (None, 512)               1606144

dense_1 (Dense)              (None, 256)               131328

dense_2 (Dense)              (None, 128)               32896

dense_3 (Dense)              (None, 64)                8256

dense_4 (Dense)              (None, 3)                 195

=================================================================
Total params: 2,075,843
Trainable params: 2,075,843
Non-trainable params: 0
```

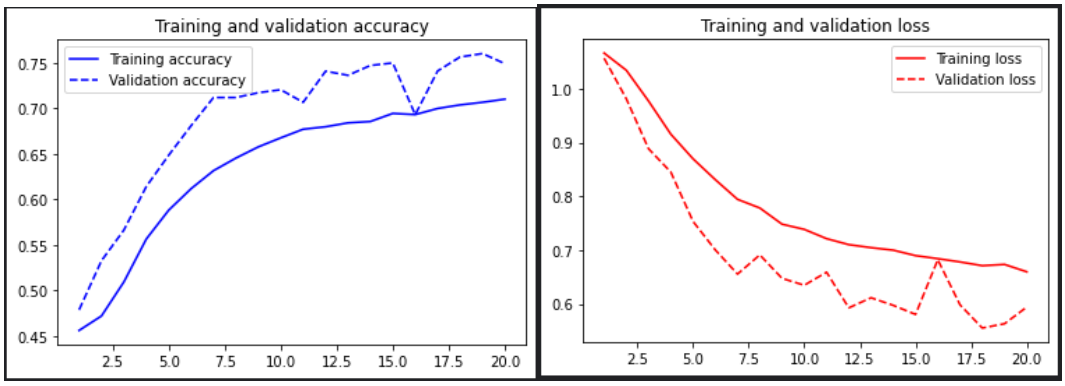
Abbildung 20: *Struktur des Modells*

Wie es zu sehen, das Modell enthält mehr als 2 Milionen Parametern. Diese Prarmetern sind alle Trainierbar.

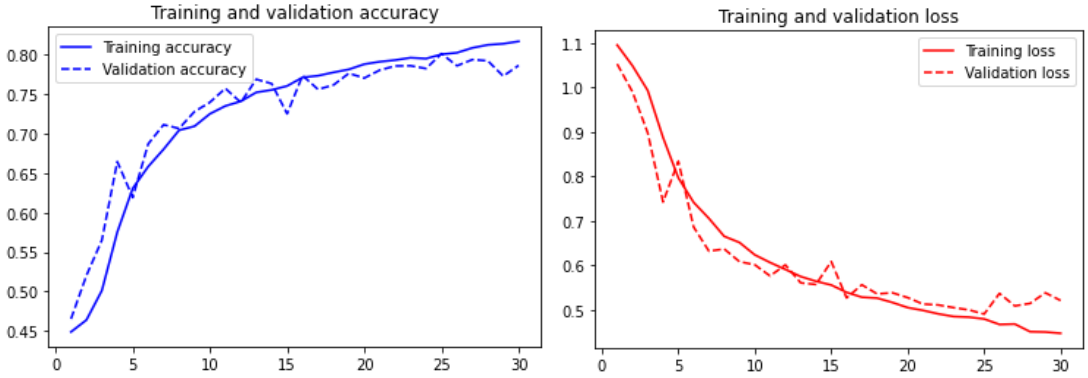
- Ausgabe von Zelle 08:

Wir haben mehr als ein Modul trainiert, um die bestmögliche Genauigkeitsrate zu erreichen. Als Start bekamen wir als Genauigkeitsrate von 70%. diese Erste Rate war gut aber nicht hoch wie wir wünschen. Mit zwiten Versuch erhalten wir ungefähr 60% und sie ist klar niedriger als die erste Rate. Jeden Mal versuchen wir die Rate zu verbessern erhalten wir neue Rate. Am Ende und mit mehreren Versuchen erhalten wir von eine Rate von 81% und das ist sicherlich die beste und höheste Genauigkeitsrate.

- Ausgabe von Zelle 09:



Die oben bereitgestellten Diagramme stammen aus der ersten Trainingshistorie. Wie man sehen kann, ist die Visualisierung der Daten schlecht, obwohl wir eine Rate von 70 % erhalten. Aus der Diagramme vesteht man dass es Überanpassung oder Mangelheit in der daten für Tests gibt.



Nach Lösung der Datensatzprobleme und Verbesserung der Struktur des Modells ist es klar zu sehen nicht nur die Genauigkeitsrate sich auf 81% ansteigt, sondern auch die Visalisierung der Daten besser sind.

- Testen das ganze System:

Alle Teilaufgaben fassen wir zu einem Gesamtsystem zusammen, um es zu testen.

- model = load_model(path)
- model.summary()

Zunächst haben wir das Modell geladen.

- emotion_labels= ['Angry','Happy','Sad']
- emotion_labels

Zelle 12

Danach haben wir in Zelle 12 die Hauptemotionsklassen als Array definiert. Die Ausgabe davon :

```
['Angry', 'Happy', 'Sad']
```

```
import cv2

img = cv2.imread('/content/drive/MyDrive/tests/happy face.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
face_cascade= cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_frontalface_alt2.xml'))
faces = face_cascade.detectMultiScale(gray, 1.1, 4)
for (x, y, w, h) in faces:
cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)
faces = img[y:y + h, x:x + w]
cv2.imwrite('detcted.jpg', img)
cv2_imshow( img)
cv2_imshow(faces)
cv2.imwrite('face.jpg', faces)
cv2.waitKey()


#resize the input image to 150*150
image= faces
def prepareResizedImage(img_path):
new_img=cv2.resize(image,(150,150))
new_img = img_to_array(new_img)
new_img = np.expand_dims(new_img, axis = 0)
return new_img
```

Zelle 13

Dann liest das System das Eingabebild, um das Gesicht zu erkennen, schneidet es zu und speichert es als neues Eingabebild, um es später für die Merkmalsextraktion und Emotionsvorhersage zu verwenden. Die Ausdabe davon ist für eine Testsbild:

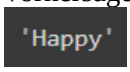
(a) (b)

(c)

- emotion_labels[np.argmax(model.predict(prepareResizedImage(image)))]

Zelle 14

Nun wird das Modul basierend auf den Trainingsergebnissen die Emotion der Person auf dem Bild vorhersagen. Die Ausgabe von Zelle 14 ist:



7. Zusammenfassung

Wir haben viel über die Technologie zur Erkennung von Gesichtsemotionen gelernt, als wir ein System entwickelten, das diese Fähigkeit besitzt, menschliche Emotionen zu lesen und zu verstehen. Dieses System kann mehr verbessern ,weiterentwickeln und in realen Zeit testen. Wenn ich das ganze Projekt in wenigen Sätzen zusammenfassen kann, kann ich sagen:

- Die Gesichtsausdruckerkennungstechnologie besteht hauptsächlich aus 3 Schritten:

    (b) Gesichtserkennung
    (c) Merkmale extrahieren
    (d) Expressionsklassifikation und -erkennung

- Es gibt viele Möglichkeiten, das menschliche Gesicht zu erkennen, z. B. mit dem kaskadierten Klassifikator von Haar, der das Gesicht schneller erkennen kann, und mit CNN können wir Emotionen genauer klassifizieren und erkennen.
- Um besser Genauigkeitsrate (Accuracy) zu erhalten gibt es viele Methoden wie Der CNN-Struktur Verbessern oder Bild Datenerweiterung.

8. Literaturverzeichnis

- Michael Jones, Paul Viola, Cambridge 2001: Rapid Object Detection using a Boosted Cascade of Simple Features
- Sciforce, Jun 17,2021: Face Detection Explained: State-of-the-Art Methods and Best Tools
- Adreeja Bardhan, Mar 12, 2020: Face Detection Comparing MTCNN,RESNET10- Haar Classifiers
- AI Insights, May 27, 2020: Face Detection: Haar Cascade vs. MTCNN
- Sampriti Chatterjee, Oct 29, 2021: What is Feature Extraction? Feature Extraction in Image Processing
- Pier Paolo Ippolito, Oct 10, 2019: Feature Extraction Techniques
- Boddepalli Kiran Kumar, Korla SwaroopaTarakeswara Rao Balaga, 2021: Facial Emotion Recognition and Detection Using CNN
- Fatih Altekin, Hasan Demir, Electronic and Communication Engineering, Çorlu Faculty of Engineering, Tekirdağ Namık Kemal University, Tekirdağ, Turkey, 16.01.2021: Electronic and Communication Engineering, Çorlu Faculty of Engineering, Tekirdağ Namık Kemal University, Tekirdağ, Turkey
- Ninad Mehendale, 18 February 2020: Facial emotion recognition using convolutional neural networks (FERC)
- Mingjie Wang et al 2020 J. Phys.: Conf. Ser. 1601 052027: Facial expression recognition based on CNN
- Sakib Hussain,BRAC University, 2013: Emotion detection from frontal facial image
- Mohammed Adnan Adil, Bachelor of Engineering, Osmania University,2016: FACIAL EMOTION DETECTION USING CONVOLUTIONAL NEURAL NETWORKS
- Sakshi_Tiwari, 08 Oct, 2020: Activation functions in Neural Networks
- IBM Cloud Education, 17 August 2020: Neural Networks
- **Datensatz**: https://www.kaggle.com/code/nguyendaitruongthanh/facial-emotion-detection-with-cnn/data