

**COLLEGE OF BIOLOGICAL AND
PHYSICAL SCIENCES SCHOOL OF
COMPUTING AND INFORMATICS
USING ARTIFICIAL INTELLIGENCE
TO CREATE A FACIAL EMOTION
DETECTOR**

By:

CHIRCHIR EMMANUEL KIBET

P15/140389/2020

CHAPTER 1

PROBLEM DEFINITION

1.0 INTRODUCTION

This document provides an overview of the Facial Emotion Recognition System, it designed to detect and interpret emotions based on facial expressions. The system utilizes advanced computer vision algorithms and machine learning models to accurately identify emotions such as being happy, sad, angry, fear, surprise and disgust in real-time. This documentation will provide a comprehensive guide to the system's functionality, including instructions, usage guidelines, and performance evaluation metrics. Whether you're a researcher, developer, or end-user, this document will provide the information you need to get started with the Facial Emotion Recognition System.

1.1 PROBLEM DEFINITION

A facial emotion recognition system is a piece of technology that analyzes photos or videos to recognize the emotion on a person's face using machine learning algorithms. This technique seeks to reliably categorize facial emotions into many emotional states, including happiness, sadness, rage, and surprise.

How to teach a machine learning model to recognize patterns and features in facial photos that are indicative of various emotions is a crucial issue that needs to be solved in order to create such a system. A dataset of pictures of faces displaying various emotions is needed for this. The model is trained using this dataset so that it may become familiar with the patterns connected to each emotion.

However, there are a number of issues that must be resolved in order to develop a facial emotion identification system. Facial emotion variety is one of these difficulties. The same feeling can be represented in numerous ways because people express their emotions in various ways. One way to express happiness, for instance, is to smile or laugh. Different emotions of an emotion should be recognizable by a good face emotion recognition system.

The alterations in lighting and position present another difficulty. A face might look different in different lighting and viewing angles, and a successful facial emotion identification system should be resistant to these changes.

Another issue that must be resolved while developing a facial emotion recognition system is the scarcity of data. In conclusion, creating a system for recognizing facial emotions of various emotions necessitates finding a solution to the challenge of teaching a machine learning model to identify patterns and characteristics in facial images that are indicative of various emotions. For the model to be trained, a dataset of labeled photos of faces displaying various emotions is needed. The system must also deal with issues like varying face emotions, poor lighting, awkward poses, and a lack of data availability.

1.2 OBJECTIVES

1.2.1 RESEARCH OBJECTIVES

The specific application and use case will determine the research goals for a facial emotion recognition system, but generally speaking, the main goals are:

- To build a reliable and precise model for identifying emotions from facial emotions: An accurate classification of facial emotions into various emotional categories, such as happiness, sadness, anger, and surprise, is the aim of training a machine learning model.
- To handle variations in facial emotions, lighting, and pose: The model should be resistant to changes in poses, lighting, and facial emotions, which can alter how a face looks and make emotion recognition more challenging.
- To deal with demographic differences, the system must be able to operate with a wide range of users, regardless of their age, gender, or ethnicity.
- Enhancing system performance: To enhance system performance, researchers may investigate various algorithms, architectures, and feature representations.
- Researchers should use established metrics, such as accuracy, to assess the system's performance.

1.2.2 SYSTEM DEVELOPMENT OBJECTIVES

Here are some specific system development objectives for a system that can identify facial emotions of emotion.

- To put into practice a model with high accuracy on the dataset used: The objective is to produce a model that can correctly categorize facial emotions.
- To implement a strong pre-processing pipeline: By utilizing pre-processing methods like face detection, alignment, and normalization, the system should be able to handle variations in lighting and facial emotions.
- To improve system performance, the model's performance should be optimized by tweaking its hyperparameters, as well as by experimenting with various architectures and feature representations.
- To implement an effective and lightweight model: The model must be lightweight in order to be used in embedded systems or other practical applications, such as mobile devices.
- To create a model that is simple to use and understand: For end users, the system should be simple to use and understand, with clear justifications for its predictions.

CHAPTER 2

RELATED WORK

2.1 FACIAL ACTION UNIT RECOGNITION

Similar to facial emotion recognition, facial action unit recognition concentrates on identifying particular facial movements, or "Action Units," as opposed to more general emotional emotions. The Facial Action Coding System, a comprehensive set of facial gestures that can be used to convey a variety of emotions, defines these AUs.

The tightening or loosening of particular facial muscle groups characterizes AUs. As an illustration, AU 1 is the contraction of the muscle that raises the inner brows, which can be used to convey surprise or skepticism. AU 6 causes the cheek raiser muscle to contract, which causes the cheeks to rise and can be used to convey happiness.

2.2 FACIAL LANDMARK DETECTION

In computer vision and image processing, a technique known as facial landmark detection is used to identify particular facial features, such as the corners of the mouth, the nose's tip, and the edges of the eyes. These landmarks on the face, also known as facial keypoints, can be used as features for a number of processes, such as face tracking, face alignment, and facial emotion recognition.

Feature extraction, feature matching, and model fitting are common techniques used in facial landmark detection algorithms. In order to locate important areas of an image, such as edges,

feature extraction techniques are used. Model fitting techniques, like an Active Shape Model, are used to fit a model and feature matching techniques are used to match keypoints across multiple images (ASM).

Traditional computer vision methods for facial landmark detection include Haar cascades and Viola-Jones, while deep learning methods include convolutional neural networks (CNNs). The Multi-task Cascaded Convolutional Networks (MTCNN) and the Facial Landmark Detection using Multi-task Deep Learning are two of the most widely used facial landmark detection algorithms (FAN).

In many computer vision and image processing tasks, such as facial emotion recognition, face alignment, and face tracking, facial landmark detection is a crucial first step. It aids in the accuracy of the process and aids in the extraction of the crucial information from the facial images.

CHAPTER 3

ANALYSIS AND DESIGN

3.1 DEVELOPMENT METHODOLOGY

The application was created utilizing the Agile development technique and Feature Driven Development. This made sure that only the features required for the application to function were developed and released at regular intervals.

Why employ a Feature Driven Development approach?

- i) More complex activities are divided up into simpler ones, making it simpler to find faults and mistakes.
- ii) Less time is spent developing the application.
- iii) It can adjust to changes in functionality more readily without suffering significant interruption.

3.2 REQUIREMENTS ANALYSIS

Functional Requirements

- Image Preprocessing: The system should be able to preprocess the images, such as resizing, normalizing, and converting them to grayscale.

- **Convolutional Neural Network:** The system should use a Convolutional Neural Network (CNN) architecture for feature extraction and classification of facial emotions.
- **Training and Validation:** The system should be able to train the CNN on the FER 2013 dataset and validate its performance using a separate validation dataset.
- **emotion Classification:** The system should be able to classify the facial emotions into one of the following categories defined in the dataset: angry, disgust,, happy, neutral, sad, and surprise.
- **Accuracy Metrics:** The system should measure and report its accuracy using metrics such as precision, recall, F1 score, and confusion matrix.
- **Real-time Detection:** The system should be able to detect facial emotions in real-time, using live video or static images.
- **User Interface:** The system should have a user-friendly interface for selecting images or videos, displaying results, and configuring settings.

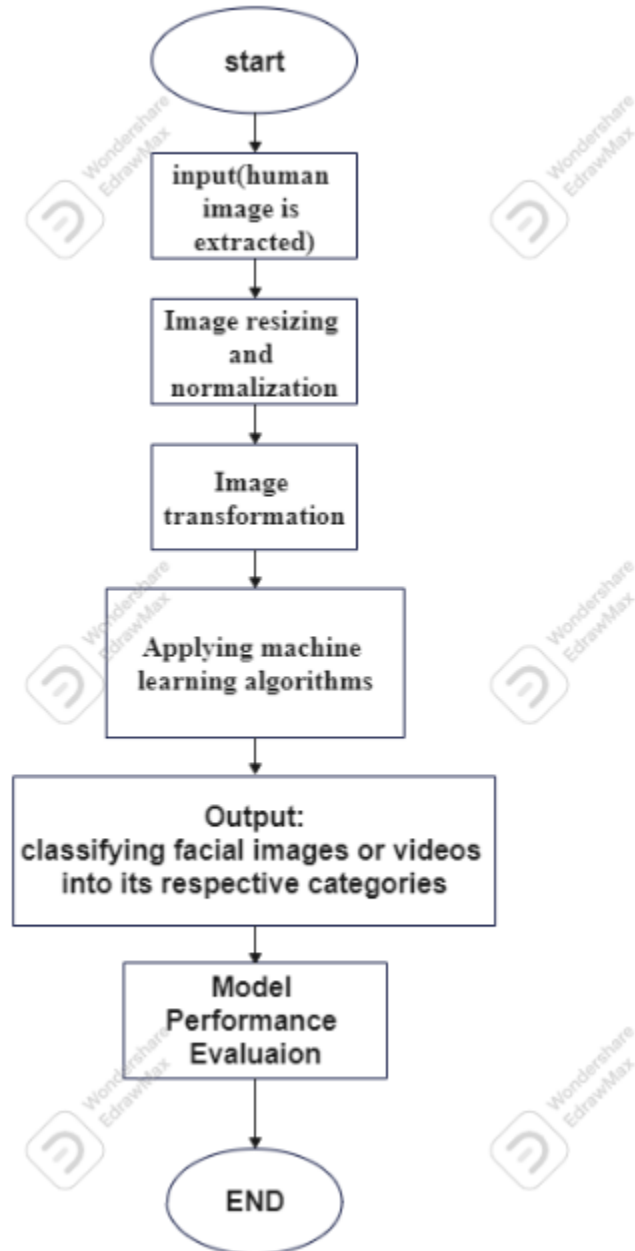
Non-Functional Requirements

- **Accuracy:** The system should have a moderate to high level of accuracy in detecting different facial emotions.
- **Robustness:** The system should be able to handle different lighting conditions, face orientations, and other factors that can affect image quality.
- **User-friendly interface:** The system should be easy to use and understand, with a simple and intuitive user interface.
- **Reliability:** The system should be reliable, with minimal downtime and errors.
- **Compatibility:** The system should be compatible with different devices and platforms, such as smartphones, laptops, and web browsers.
- **Cost:** The system should be affordable and accessible, with minimal costs for development, maintenance, and usage.

3.3 DESIGN.

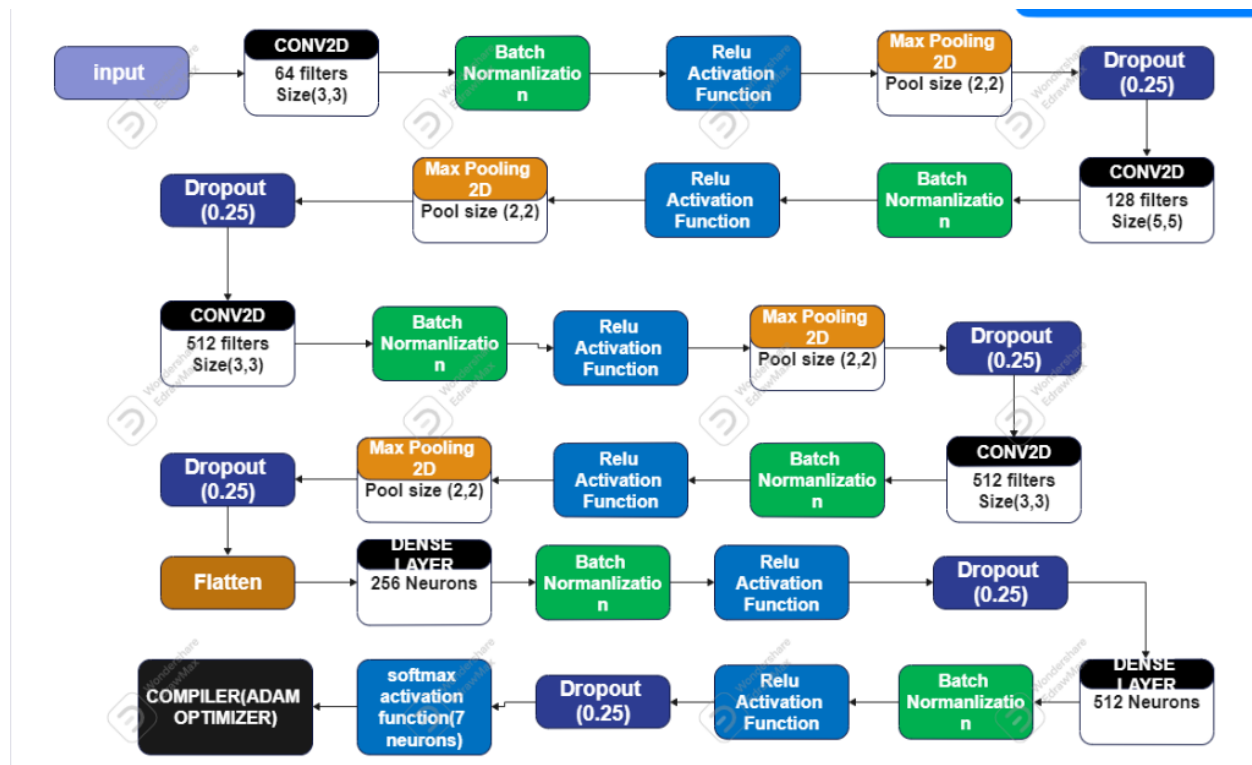
i)Data flow Diagram

The Data flow Diagram below will help visualize the flow of the data, from capturing the input image of a face, to processing the image through the machine learning techniques, to outputting a prediction of the emotional state and finally carrying out a performance evaluation of the model.



i)The Model

Below is a graphical representation of how the model will work.



UNDERSTANDING THE MODEL

The model architecture below is a convolutional neural network (CNN) with four Conv2D layers, each followed by a BatchNormalization layer and an activation layer (ReLU), a MaxPooling2D layer, and a Dropout layer. The input shape of the model is (48, 48, 1), indicating the input image is 48x48 with 1 channel (grayscale).

The first Conv2D layer has 64 filters of size (3,3) and uses 'same' padding, meaning that the output shape of this layer will have the same size as the input. After this layer, a BatchNormalization layer is applied, followed by a ReLU activation function. The next MaxPooling2D layer has a pool size of (2,2), meaning that it will down-sample the output by a factor of 2 in both dimensions. The Dropout layer with a rate of 0.25 randomly drops out some neurons during training to prevent overfitting.

The second Conv2D layer has 128 filters of size (5,5), followed by the same set of layers (BatchNormalization, ReLU activation, MaxPooling2D, and Dropout).

The third and fourth Conv2D layers are similar to the second, with 512 filters of size (3,3) each, followed by the same set of layers.

The model then flattens the output of the last MaxPooling2D layer into a 1D array.

Two dense (fully connected) layers are added, each with 256 and 512 neurons respectively, followed by a BatchNormalization layer and a ReLU activation function.

The output of the last dense layer is connected to 7 neurons, with a softmax activation function to produce 7 class probabilities.

The model is compiled with the **Adam optimizer** and a categorical cross-entropy loss function. The accuracy of the model is also monitored during training.

CHAPTER 4

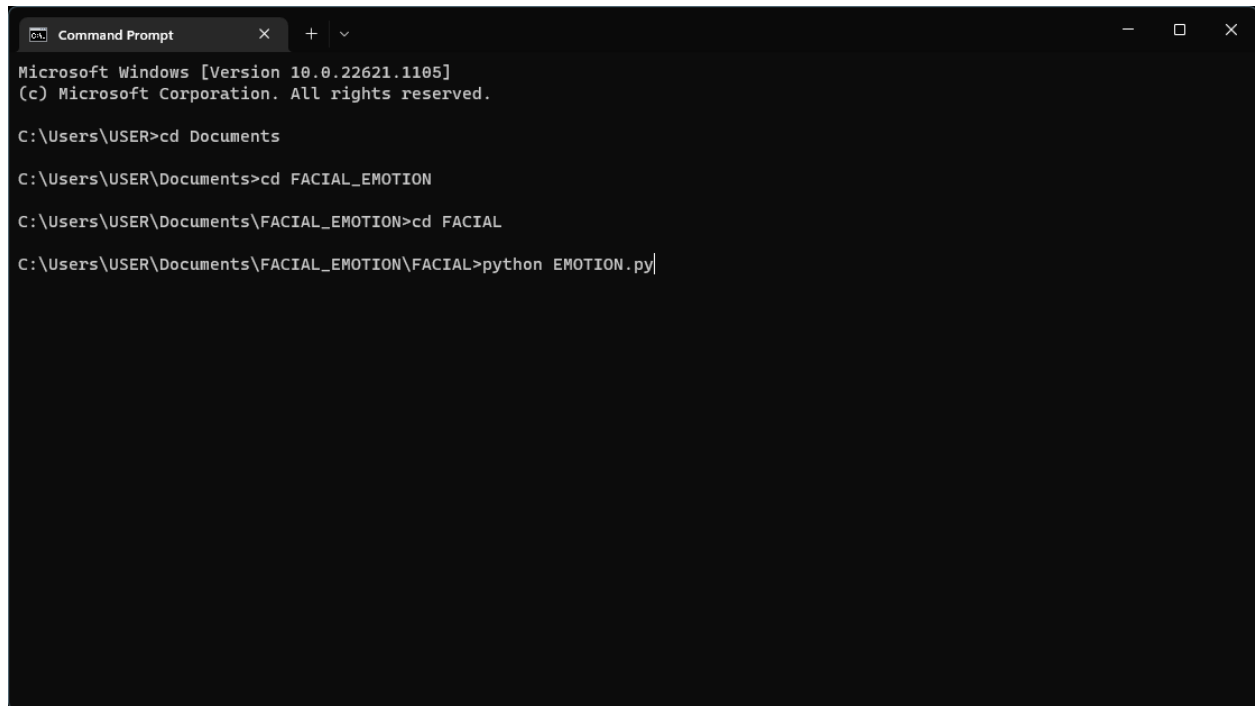
IMPLEMENTATION AND TESTING

4.1 PROGRAMMING TOOLS AND TECHNOLOGIES USED

1. Windows Operating System.
2. Python, Keras, OpenCV, numpy
3. Jupyter Notebook based in kaggle.
4. Visual studio code text editor.

4.2 DEMONSTRATION

The Facial emotion recognition system is run by running the python file.



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

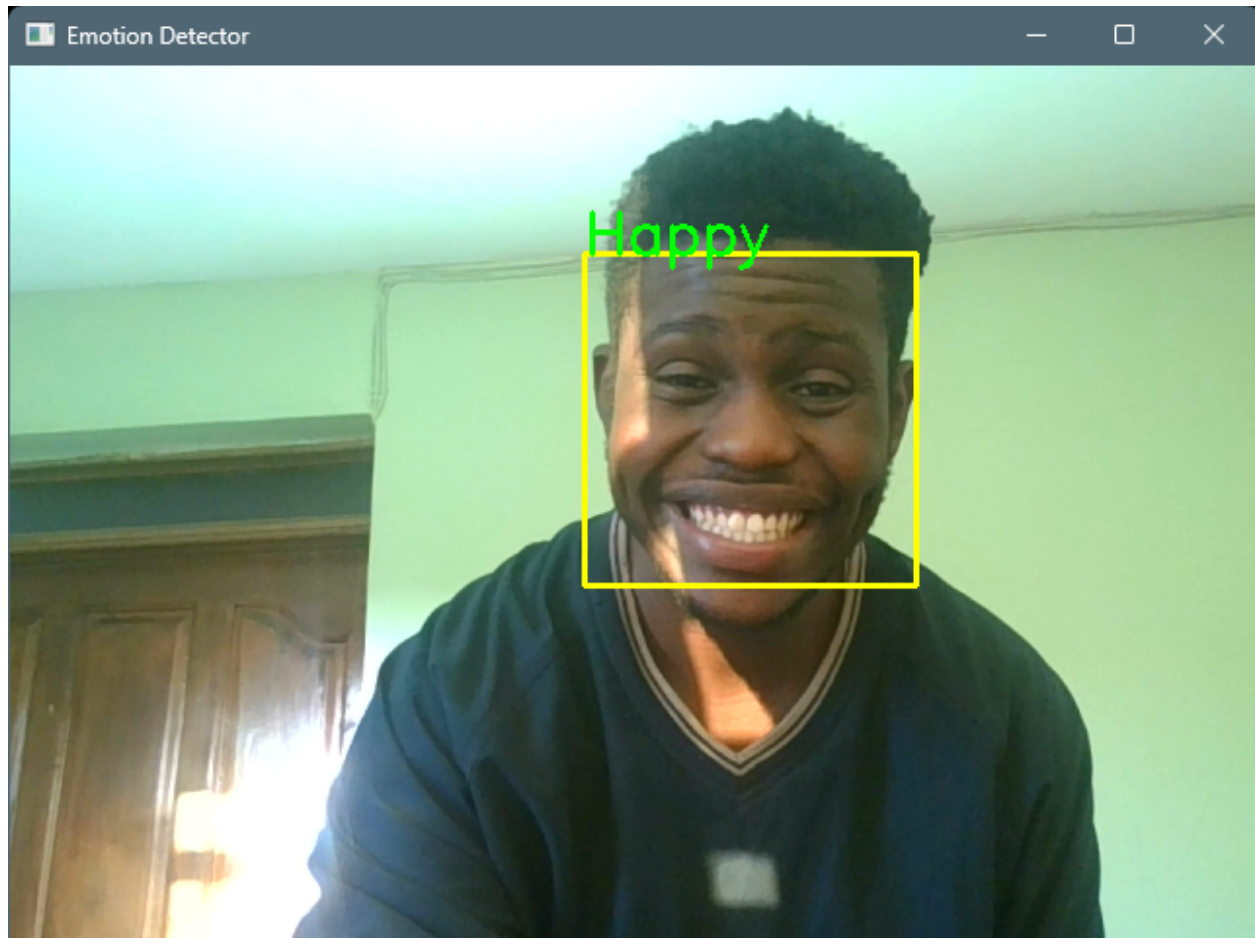
C:\Users\USER>cd Documents

C:\Users\USER\Documents>cd FACIAL_EMOTION

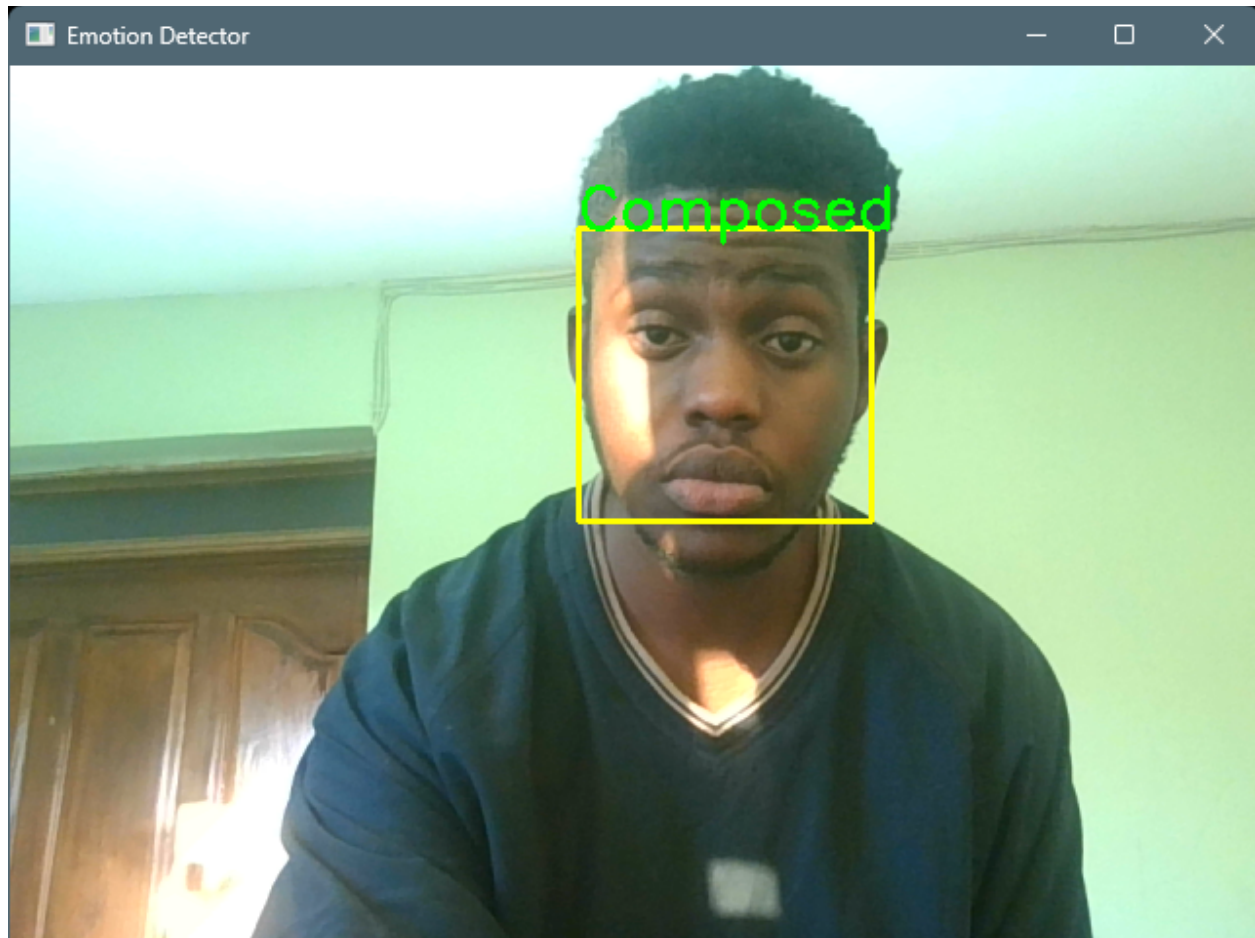
C:\Users\USER\Documents\FACIAL_EMOTION>cd FACIAL

C:\Users\USER\Documents\FACIAL_EMOTION\FACIAL>python EMOTION.py
```

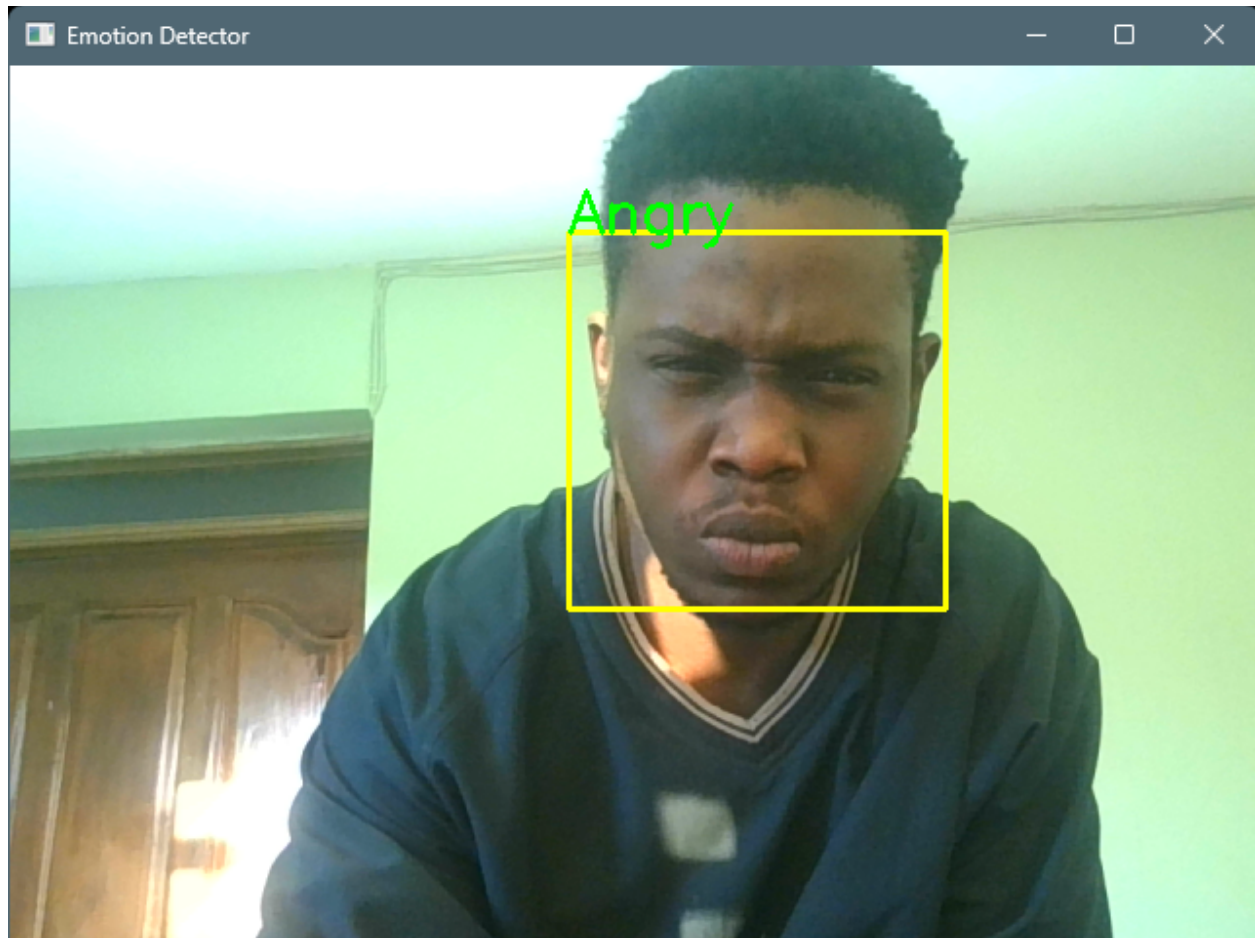
Then a window of size 255 by 255 then pops up in real time.
The model will classify your face based on the emotions made on your face:
For a smiling face it will predict as:



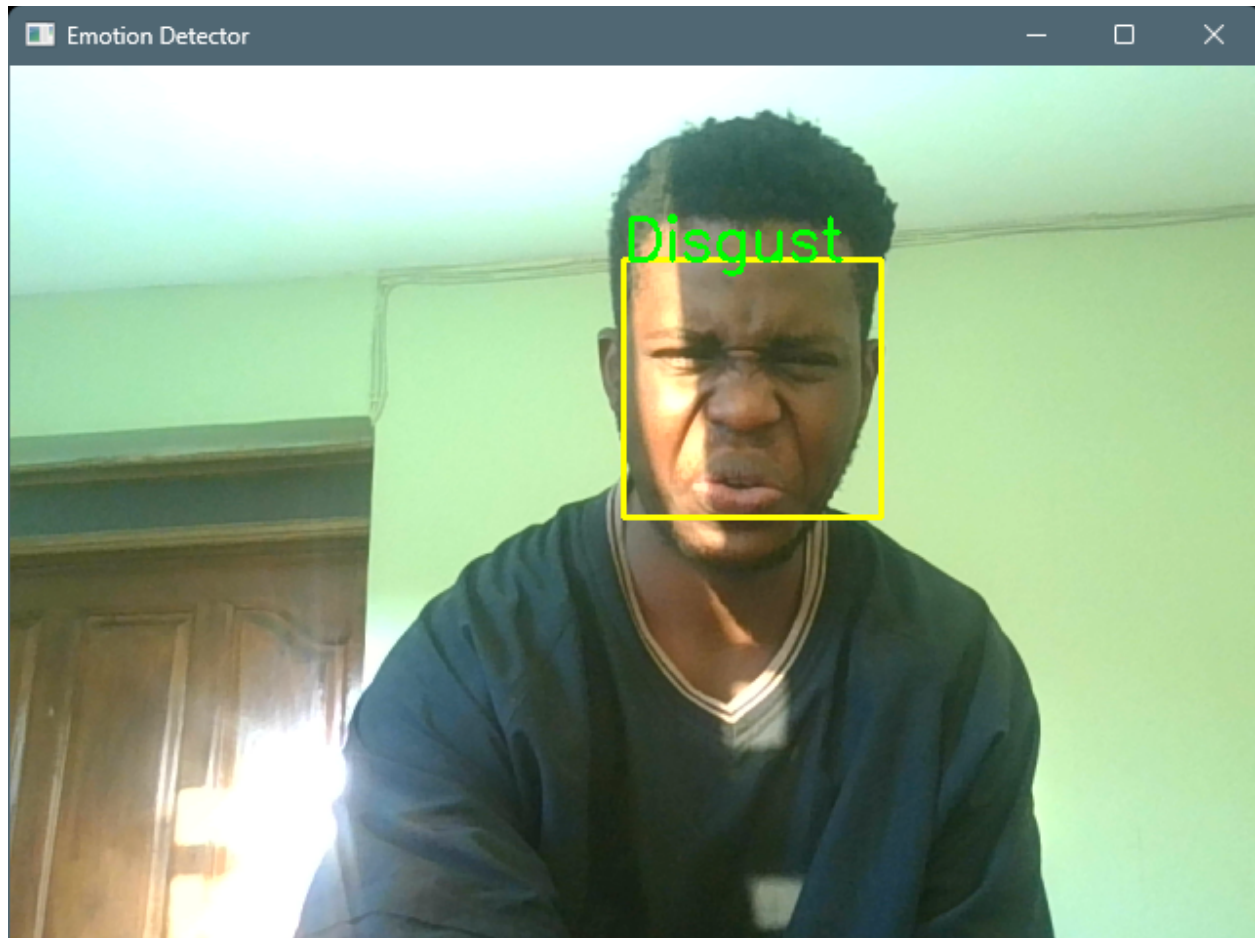
For a normal and neutral face, it will predict:



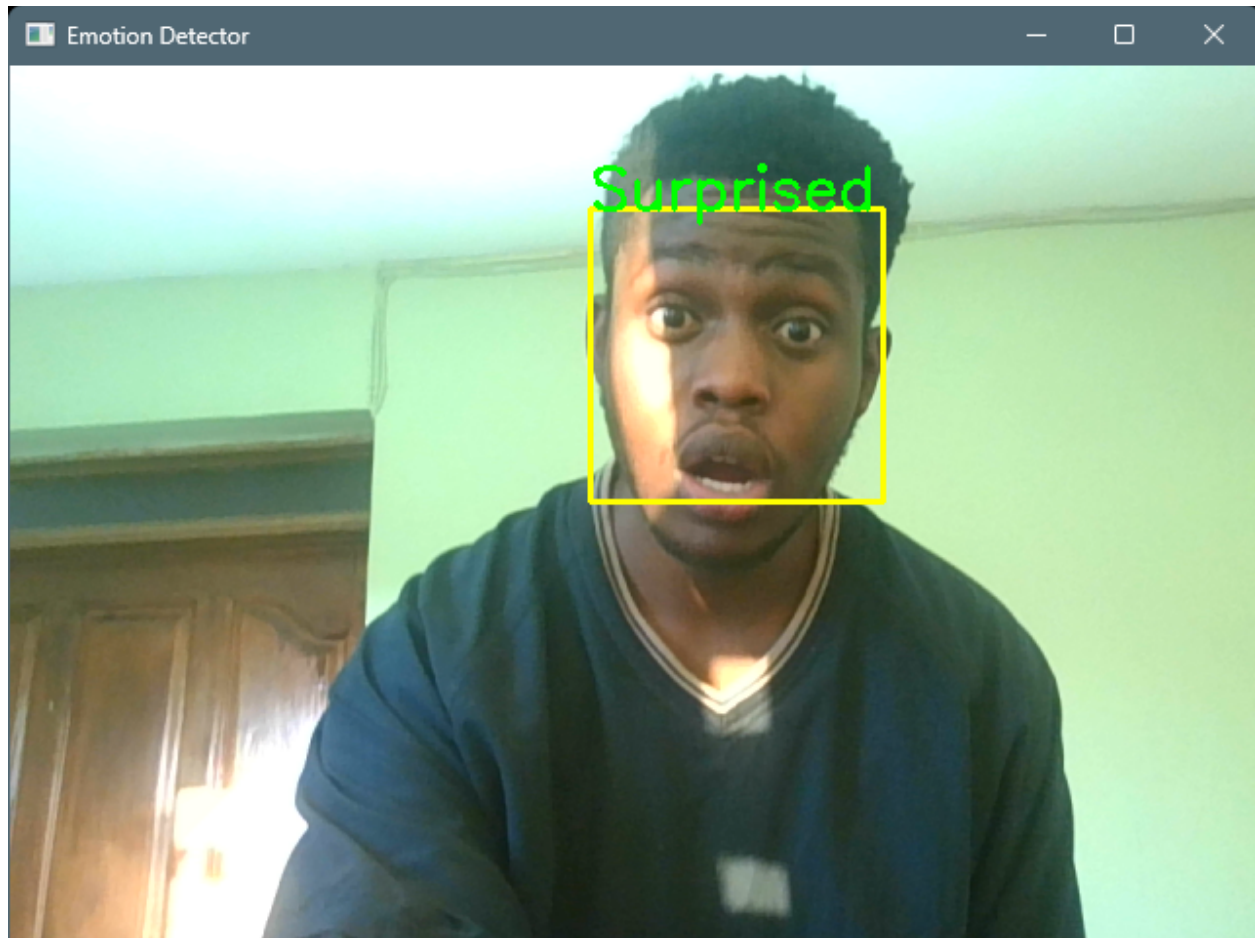
For an angry face, it will predict:



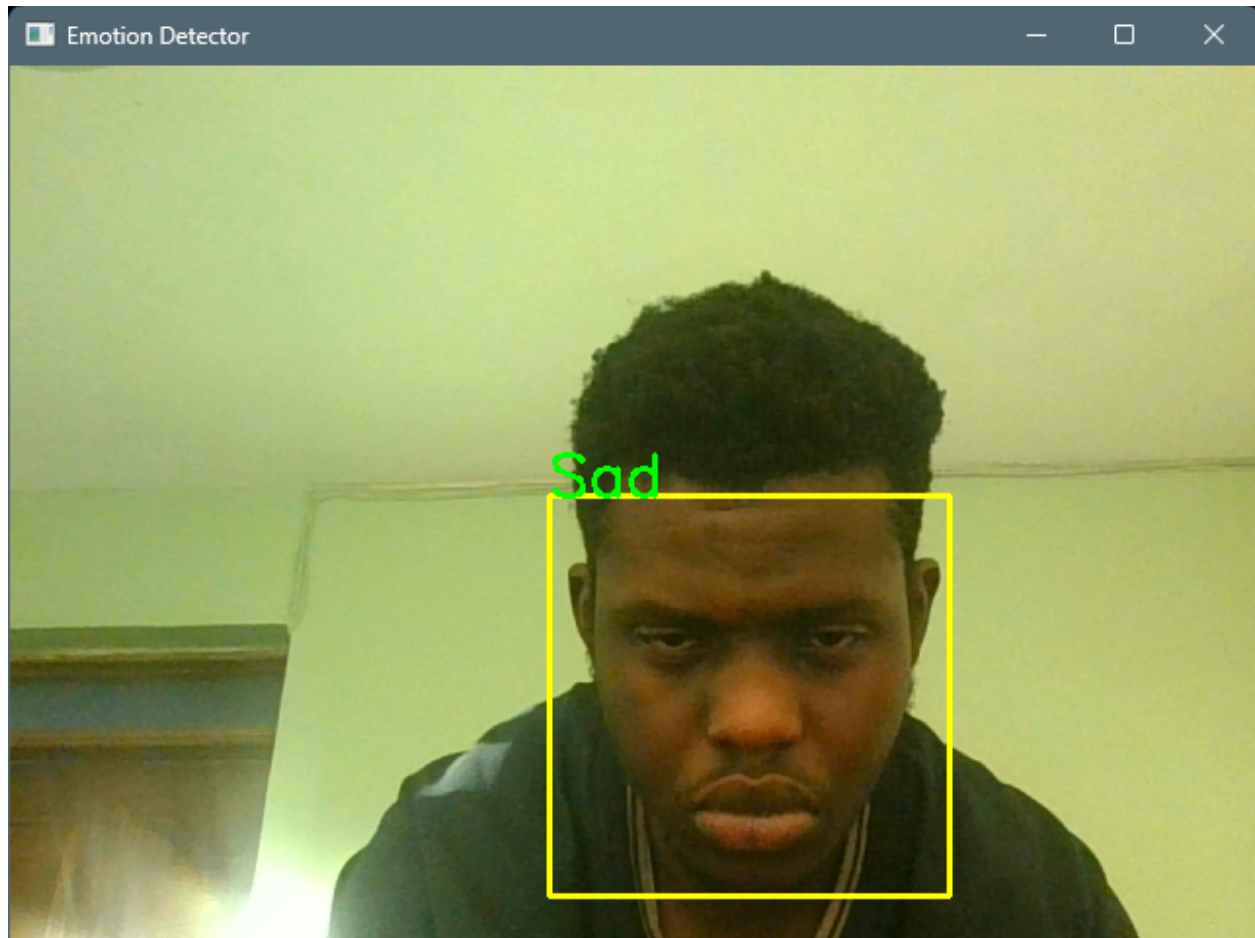
For a disgusted face it will predict:



For a surprised face it will predict:



For a sad face it will predict:



CHAPTER 5

PERFORMANCE EVALUATION

5.1 MODEL ACCURACY

The model performance metrics was based on both the training accuracy and the validation accuracy. The final validation accuracy was 64.32 while the training accuracy was 70.82

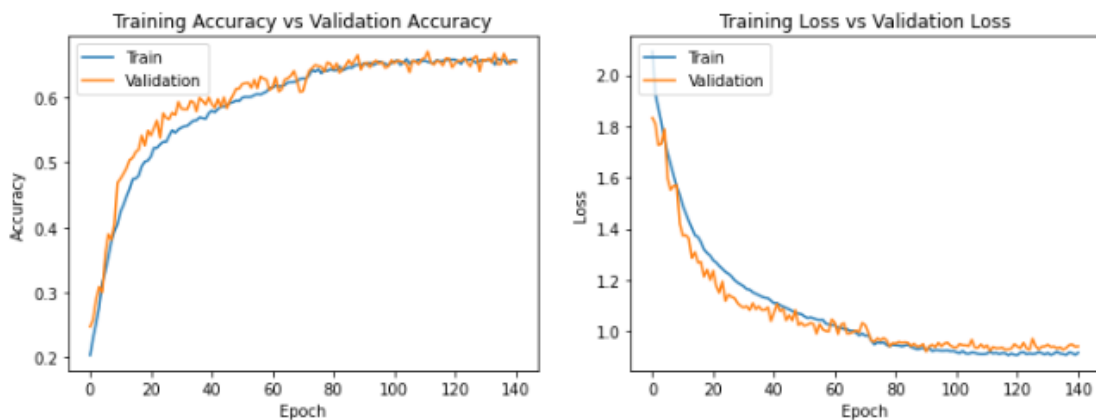

```
print("final train accuracy = {:.2f} , validation accuracy = {:.2f}".format(
359/359 [=====] - 41s 113ms/step - loss: 0.7762 - accuracy: 0.70
82
23/23 [=====] - 2s 105ms/step - loss: 0.9541 - accuracy: 0.6432
final train accuracy = 70.82 , validation accuracy = 64.32
```

```
[11]: fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
```

5.1.1 MODEL PERFORMANCE GRAPHS

The image below is a graphical representation on the left of the training accuracy against validation accuracy while the other to the right graphical representation is that of training loss against validation loss

```
plt.show()
```



+ Code

+ Markdown

```
[ ]:
```

onsole

APPENDIX:

SAMPLE CODE SCREENSHOTS

The code for the interface that would deploy the model:

```
File Edit Selection View Go Run Terminal Help
EMOTION.py - Visual Studio Code

Get Started EMOTION.py 4 x facial emotion.ipynb facial-emotion.ipynb notebookfde360b676.ipynb

C:\Users\USER> Documents > FACIAL_EMOTION > FACIAL > EMOTION.py > ...

17 while True:
18     _, frame = cap.read()
19     labels = []
20     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
21     faces = face_classifier.detectMultiScale(gray)
22
23     for (x,y,w,h) in faces:
24         cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,255), 2)
25         roi_gray = gray[y:y+h, x:x+w]
26         roi_gray = cv2.resize(roi_gray, (48,48), interpolation=cv2.INTER_AREA)
27
28
29
30         if np.sum([roi_gray])!=0:
31             roi = roi_gray.astype('float')/255.0
32             roi = img_to_array(roi)
33             roi = np.expand_dims(roi, axis=0)
34
35             prediction = classifier.predict(roi)[0]
36             label=emotion_labels[prediction.argmax()]
37             label_position = (x,y)
38             cv2.putText(frame, label, label_position, cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
39         else:
40             cv2.putText(frame, 'No Faces', (30,80), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
41     cv2.imshow('Emotion Detector', frame)
42     if cv2.waitKey(1) & 0xFF == ord('q'):
43         break
44
45 cap.release()
46 cv2.destroyAllWindows()
47
48
```

```
File Edit Selection View Go Run Terminal Help
EMOTION.py - Visual Studio Code

Get Started EMOTION.py 4 x facial emotion.ipynb facial-emotion.ipynb notebookfde360b676.ipynb

C:\Users\USER> Documents > FACIAL_EMOTION > FACIAL > EMOTION.py > ...

1 from keras.models import load_model
2 from time import sleep
3 from keras.utils import img_to_array
4 from keras.preprocessing import image
5 import cv2
6 import numpy as np
7
8 face_classifier = cv2.CascadeClassifier('./haarcascade_frontalface_default.xml')
9 classifier = load_model('./model1.h5')
10
11 emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Composed', 'Sad', 'Surprised']
12
13 cap = cv2.VideoCapture(0)
14
15
16
17 while True:
18     _, frame = cap.read()
19     labels = []
20     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
21     faces = face_classifier.detectMultiScale(gray)
22
23     for (x,y,w,h) in faces:
24         cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,255), 2)
25         roi_gray = gray[y:y+h, x:x+w]
26         roi_gray = cv2.resize(roi_gray, (48,48), interpolation=cv2.INTER_AREA)
27
28
29
30         if np.sum([roi_gray])!=0:
31             roi = roi_gray.astype('float')/255.0
32             roi = img_to_array(roi)
```

The code below was used to train the model:

kaggle.com/chirchiremanuel/facial-emotion/edit

FACIAL EMOTION

Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code

Draft Session off (run a cell to start)

```
Trainable params: 4,474,759
Non-trainable params: 3,968
```

```
[7]:
checkpoint = ModelCheckpoint("./model.h5", monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                                min_delta=0,
                                patience=50,
                                verbose=1,
                                restore_best_weights=True
                                )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                                           factor=0.2,
                                           patience=10,
                                           verbose=1,
                                           min_delta=0.0001)

callbacks_list = [early_stopping, checkpoint, reduce_learningrate]

epochs = 300
```

Console

kaggle.com/chirchiremanuel/facial-emotion/edit

FACIAL EMOTION

Draft saved

File Edit View Run Add-ons Help

Share Save Version 0

Run All Code

Draft Session off (run a cell to start)

```
[2]:
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
import keras
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout, BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from tensorflow.keras.applications import VGG16, InceptionResNetV2
from keras import regularizers
from tensorflow.keras.optimizers import Adam, RMSprop, SGD, Adamax
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

/kaggle/input/fer2013/test/surprise/PublicTest_78686873.jpg
/kaggle/input/fer2013/test/surprise/PrivateTest_58522921.jpg
/kaggle/input/fer2013/test/surprise/PrivateTest_83796714.jpg
/kaggle/input/fer2013/test/surprise/PublicTest_84428313.jpg
/kaggle/input/fer2013/test/surprise/PrivateTest_87978901.jpg
/kaggle/input/fer2013/test/surprise/PrivateTest_90978621.jpg
/kaggle/input/fer2013/test/surprise/PrivateTest_27580582.jpg
```

Console

