**faceAlter: Real Time Emojis with Facial Expressions**

**A PROJECT REPORT**

*Submitted by*

**Hitesh Agarwal 19BAI10030**

**Dev Singh 19BAI10093**

**Akshat Bharadwaj 19BAI10188**

**P. Yash Reddy 19BAI10190**

*In partial fulfillment for the award of the degree*

*Of*

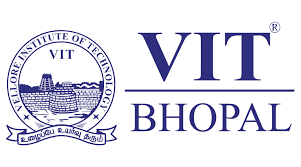
**BACHELOR OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

*Specialization in*

***ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING***

****

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**VIT BHOPAL UNIVERSITY**

**KOTHRIKALAN, SEHORE**

**MADHYA PRADESH- 466114**

MAY 2021

**VIT BHOPAL UNIVERSITY KOTHRIKALAN,SEHORE**

**MADHYA PRADESH- 466114**

**BONAFIDE CERTIFICATE**

Certified that this project report titled “**faceAlter- Real Time Emojis with Facial Expressions**” is the bonafide work of **“HITESH AGARWAL (19BAI10030), DEV SINGH (19BAI10093), AKSHAT BHARADWAJ (19BAI10188), P. YASH REDDY (19BAI10190)”** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project/research work on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

**PROGRAM CHAIR PROJECT GUIDE**

DR. S SOUNTHARRAJAN, DR. PON HARSHAVARDHANAN,

Senior Assistant Professor, Associate Professor,

School of AI and ML Division, School of AI and ML Division,

VIT BHOPAL UNIVERSITY. VIT BHOPAL UNIVERSITY.

The Project Exhibition II examination is held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ACKNOWLEDGEMENT**

First and foremost, we would like to thank the Almighty for His presence and immense blessing throughout the project work.

We wish to express our heartfelt gratitude to Dr. S SOUNTHARRAJAN, Head of the Department, School of Computer Science for much of his valuable support and encouragement in carrying out this work.

We would like to thank our internal guide Dr. PON HARSHAVARDHANAN for continually guiding and actively participating in our project, giving valuable suggestions to complete the Project Work.

We would like to thank all the technical and teaching staff of the School of Computer Engineering, who extended directly or indirectly all the support.

Last, but not the least, we are deeply indebted to our Parents who have been the greatest support while we worked day and night on the project to make it a success.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Serial no** | **Title** | **Page no.** |
| 1 | Abstract | 05 |
| 2 | List of Abbreviation and List of Figures | 06 |
| 3 | Introduction | 07 |
| 4 | Literature Survey | 08 |
| 5 | Project Procedure | 09-10 |
| 6 | Work Done | 11-18 |
| 7 | Observation and Result | 19 - 20 |
| 8 | Conclusion and Future Work | 21 |
| 9 | References | 22 |
|  | | |

**ABSTRACT**

Emoji are ideograms used in messages and webpages to interact in a way that resembles the way we feel. Emojis are of different kinds, facial expressions, common objects, places and animals. It also led to increasing data science research dedicated to emoji-driven storytelling.

This project is based on making emojis using machine learning from an image. With advancements in computer vision and deep learning, it is now possible to detect human emotions from images. In this project, we will classify human facial expressions to filter and map corresponding emojis.

In order to do so, we will be utilizing the convolution neural network feature to recognize facial emotions, and for that purpose the dataset to be used is “FER2013”. The model will be trained using this dataset to map the emotion to the corresponding emoji. The input will be fed to the model in the form of a “box” which will be obtained using haarcascade xml by OpenCV which will generate the bounding box of the faces in the webcam.

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **ABBREVIATION** | **FULL FORM** |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| CNN | Convolutional Neural Network |
| FER2013 | Facial Expression Recognition 2013 |
| IDE | Integrated Development Environment |
| GUI | Graphical User Interface |
| ADAM | Adaptive Movement Estimation |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **TITLE** | **PAGE NO** |
| 1 | Output Images | 19 - 20 |

**INTRODUCTION**

Artificial Intelligence is an interdisciplinary branch of computer science, however, a good part of it is commercialized now as a technology. It involves computers “learning” from a massive amount of data so that they can do tasks that normally require human intervention. Computer scientists and mathematicians have conducted decades of research and development on AI.

If you are wondering why AI matters, then you only have to look at the wide range of use cases it has. A few examples of AI use cases are as follows:

Static image recognition, classification, and tagging

Algorithmic trading (basically through Trading Bots)

Predictive maintenance of equipment in heavy industries

Processing patient data in an efficient and scalable manner

Artificial Intelligence “trains” computer systems using algorithms so that an AI-powered system can progressively improve its performance. One such application of this outstanding branch is the Facial Recognition that has been used in various fields such as security purposes and fun applications these are efficient and this is the task that we are designing in our current project.

**LITERATURE SURVEY**

The project Face Alter, is based on making emojis using machine learning from an image.

To do so, we will be building a model which will be utilizing the convolution neural network feature to recognize facial emotions, and for that purpose the dataset to be used is “FER2013”. The detection of emotions or facial expressions requires analysis and assessment of decisions in predicting a person's emotions or group of people in communicating. This research proposes the design of a system that can predict and recognize the classification of facial emotions based on feature extraction using the Convolution Neural Network (CNN) algorithm in real-time.

The model will be trained using this dataset to map the emotion to the corresponding emoji. The input will be fed to the model in the form of a “box” which will be obtained using haarcascade xml by OpenCV which will generate the bounding box of the faces in the webcam.

There on, facial features aka emotions would be extracted then fed to model to classify them under given 6 facial expressions and then display them in real time.

At last, we will have a real time emojifier of yourself.

**PROJECT PROCEDURE**

As mentioned earlier, we will be building a convolution neural network architecture using OpenCV and train our model using FER2013 dataset.

The workflow of faceAlter is a 3-stage process:

Stage 1 - Facial detection and tracing

Stage 2 - Extraction of features

Stage 3 - Classification

The model uses haarcascade xml to detect the face and traces the bounding box in the webcam capturing a frame which is then preprocessed. After preprocessing and extraction of the required features, the feature points are then fed to the model for classification.

Step 1: To choose a suitable programming language

Programming languages like Python are the most popular and suited languages to develop a Machine Learning model as they are extremely flexible and versatile and if incase, we want to further develop our model in the future, it is going to make our work easier.

Step 2: To choose a platform/IDE

Pycharm or Anaconda are the most preferred platforms for developing machine learning models.

Step3: Understanding the basic concepts required

faceAlter deals with various niches of computer science making it a sophisticated machine learning model. The pre requisites for the development are fundamentals of computer vision, basic understanding of deep learning concepts, knowledge about convolutional neural networks and OpenCV.

Step 4: Understanding objective and capabilities of faceAlter

This is also one of the most crucial aspects of our model as the user should be clear about what can it do and what it is made to do.

Step 5: Developing the model

Goal in this should be to keep the code simple and concise so that it is easier to optimize and traverse. While building the model make sure that proper documentation is done in order to make it easier for the end user to understand faceAlter. Proper code ethics and good decision-making flow should be implemented.

Step 6: Training - Testing - Optimizing

Keeping logs is a fantastic way to keep our model updated. Any kind of errors can be identified easily by the timestamps in the log file. Using 80:20 ratio of dataset we train our model and then we test it thoroughly. We keep tabs of all the processes to look for ways to optimize our model and get better and more accurate results.

NumPy: NumPy is a Python library that provides a simple yet powerful data structure: n-dimensional array. In that, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays.

OpenCV/cv2: OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Keras: Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

Keras.layers

Keras.optimizers

Keras.preprocessing.image

Keras.emotion\_models

PIL: Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.

OS: The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

**WORK DONE**

#Importing necessary modules/libraries for training

#Importing necessary modules/libraries for training

import numpy as np

import cv2

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D

from keras.optimizers import Adam

from keras.layers import MaxPooling2D

from keras.preprocessing.image import ImageDataGenerator

#Setting up global variables

train\_dir = 'C:/Users/aksha/Desktop/Project/FER2013/train'

val\_dir = 'C:/Users/aksha/Desktop/Project/FER2013/test'

train\_datagen = ImageDataGenerator(rescale=1./255)

val\_datagen = ImageDataGenerator(rescale=1./255)

train\_generator = train\_datagen.flow\_from\_directory(

train\_dir,

target\_size=(48,48),

batch\_size=64,

color\_mode="grayscale",

class\_mode='categorical')

validation\_generator = val\_datagen.flow\_from\_directory(

val\_dir,

target\_size=(48,48),

batch\_size=64,

color\_mode="grayscale",

class\_mode='categorical')

#Preparing model for training

emotion\_model = Sequential()

emotion\_model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48,48,1)))

emotion\_model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Flatten())

emotion\_model.add(Dense(1024, activation='relu'))

emotion\_model.add(Dropout(0.5))

emotion\_model.add(Dense(7, activation='softmax'))

# emotion\_model.load\_weights('emotion\_model.h5')

#Training the model

cv2.ocl.setUseOpenCL(False)

emotion\_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}

emotion\_model.compile(loss='categorical\_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-6),metrics=['accuracy'])

emotion\_model\_info = emotion\_model.fit\_generator(

train\_generator,

steps\_per\_epoch=28709 // 64,

epochs=1,

validation\_data=validation\_generator,

validation\_steps=7178 // 64)

emotion\_model.save\_weights('emotion\_model.h5')

#Setting up the webcam and creating bounding box

cap = cv2.VideoCapture(0)

while True:

# Find haar cascade to draw bounding box around face

ret, frame = cap.read()

if not ret:

break

bounding\_box =

cv2.CascadeClassifier('C:/Users/aksha/Desktop/haarcascade\_frontalface\_

default.xml')

gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2gray)

num\_faces = bounding\_box.detectMultiScale(gray\_frame,scaleFactor=1.3,

minNeighbors=5)

for (x, y, w, h) in num\_faces:

cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

roi\_gray\_frame = gray\_frame[y:y + h, x:x + w]

cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray\_frame, (48,

48)), -1), 0)

#Setting up the webcam and creating bounding box

emotion\_prediction = emotion\_model.predict(cropped\_img)

maxindex = int(np.argmax(emotion\_prediction))

cv2.putText(frame, emotion\_dict[maxindex], (x+20, y-60),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE\_AA)

cv2.imshow('Video', cv2.resize(frame,(1200,860),interpolation = cv2.INTER\_CUBIC))

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

#Importing necessary modules/libraries for GUI

import tkinter as tk

from tkinter import \*

import cv2

from PIL import Image, ImageTk

import os

import numpy as np

#Setting up global variables

emotion\_dict = {0: " Angry ", 1: "Disgusted", 2: " Fearful ", 3: " Happy ", 4: " Neutral ", 5: " Sad ", 6: "Surprised"}

emoji\_dist={0:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/angry.png",1:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/disgusted.png",2:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/fearful.png",3:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/happy.png",4:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/neutral.png",5:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/sad.png",6:"C:/Users/aksha/Desktop/Project/emoji-creator-project-code/emojis/emojis/surpriced.png"}

global last\_frame1

last\_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)

global cap1

show\_text=[3]

#Defining functions

def show\_vid2():

frame2=cv2.imread(emoji\_dist[show\_text[0]])

pic2=cv2.cvtColor(frame2,cv2.COLOR\_BGR2RGB)

img2=Image.fromarray(frame2)

imgtk2=ImageTk.PhotoImage(image=img2)

lmain2.imgtk2=imgtk2

lmain3.configure(text=emotion\_dict[show\_text[0]],font=('arial',45,'bold'))

lmain2.configure(image=imgtk2)

print("b4")

#show\_vid()

print("after")

lmain2.after(dlp, show\_vid2)

def show\_vid():

cap1 = cv2.VideoCapture(0, cv2.CAP\_DSHOW)

if not cap1.isOpened():

print("cant open the camera1")

print("Cam is opened")

flag1, frame1 = cap1.read()

frame1 = cv2.resize(frame1,(600,500))

bounding\_box =

cv2.CascadeClassifier('C:/Users/aksha/Desktop/haarcascade\_frontalface\_

default.xml')

gray\_frame = cv2.cvtColor(frame1, cv2.COLOR\_BGR2GRAY)

num\_faces = bounding\_box.detectMultiScale(gray\_frame,scaleFactor=1.3,

minNeighbors=5)

for (x, y, w, h) in num\_faces:

cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)

roi\_gray\_frame = gray\_frame[y:y + h, x:x + w]

cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray\_frame, (48,

48)), -1), 0)

prediction = emotion\_model.predict(cropped\_img)

maxindex = int(np.argmax(prediction))

cv2.putText(frame1, emotion\_dict[maxindex], (x+20, y-60),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE\_AA)

show\_text[0]=maxindex

if flag1 is None:

print ("Major error!")

elif flag1:

global last\_frame1

last\_frame1 = frame1.copy()

pic = cv2.cvtColor(last\_frame1, cv2.COLOR\_BGR2RGB)

img = Image.fromarray(pic)

imgtk = ImageTk.PhotoImage(image=img)

lmain.imgtk = imgtk

lmain.configure(image=imgtk)

lmain.after(dlp, show\_vid)

#cv2.imshow('Emotion',frame1)

if cv2.waitKey(10) == ord('b'):

exit()

#cap1.release()

print("lol")

#show\_vid2()

print("no")

#cv2.destroyAllWindows()

#Main function and GUI

if \_\_name\_\_ == '\_\_main\_\_':

root=tk.Tk()

img = ImageTk.PhotoImage(Image.open("C:/Users/aksha/Desktop/pure-black-background-f82588d3.jpg"))

heading = Label(root,image=img,bg='black')

heading.pack()

heading2=Label(root,text="Photo to Emoji",pady=0, font=('arial',45,'bold'),bg='black',fg='#CDCDCD')

heading2.pack()

lmain = tk.Label(master=root,padx=50,bd=10)

lmain2 = tk.Label(master=root,bd=10)

lmain3=tk.Label(master=root,bd=10,fg="#CDCDCD",bg='black')

lmain.pack(side=LEFT)

lmain.place(x=50,y=250)

lmain3.pack()

lmain3.place(x=960,y=250)

lmain2.pack(side=RIGHT)

lmain2.place(x=900,y=350)

root.title("Photo To Emoji")

root.geometry("1400x900+100+10")

root['bg']='black'

exitbutton = Button(root,text='Quit',fg="red",command=root.destroy

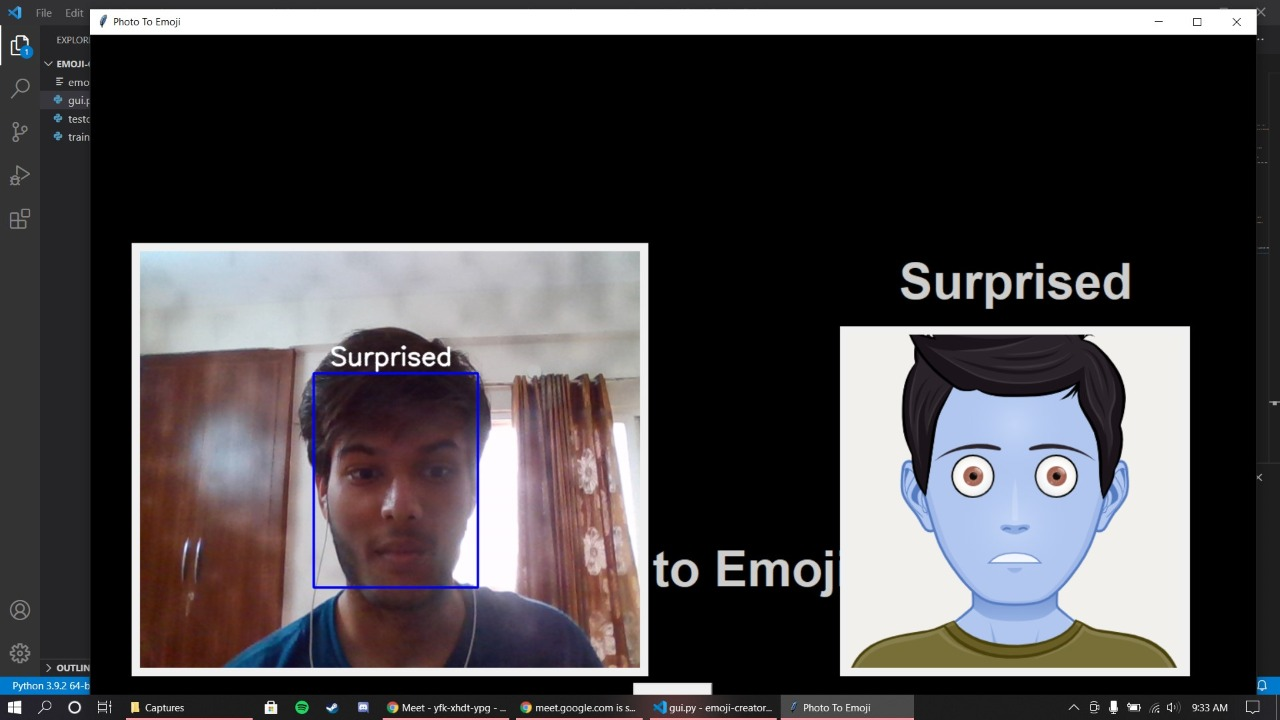
font=('arial',25,'bold')).pack(side = BOTTOM)

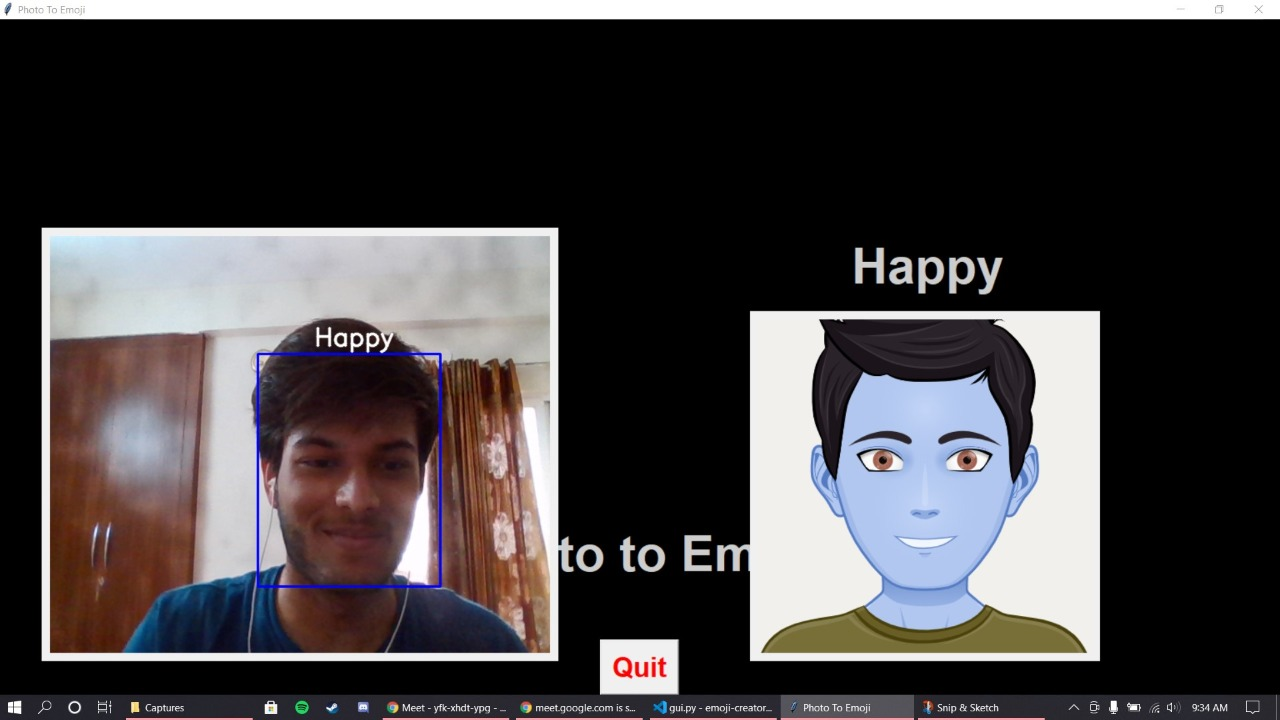
show\_vid()

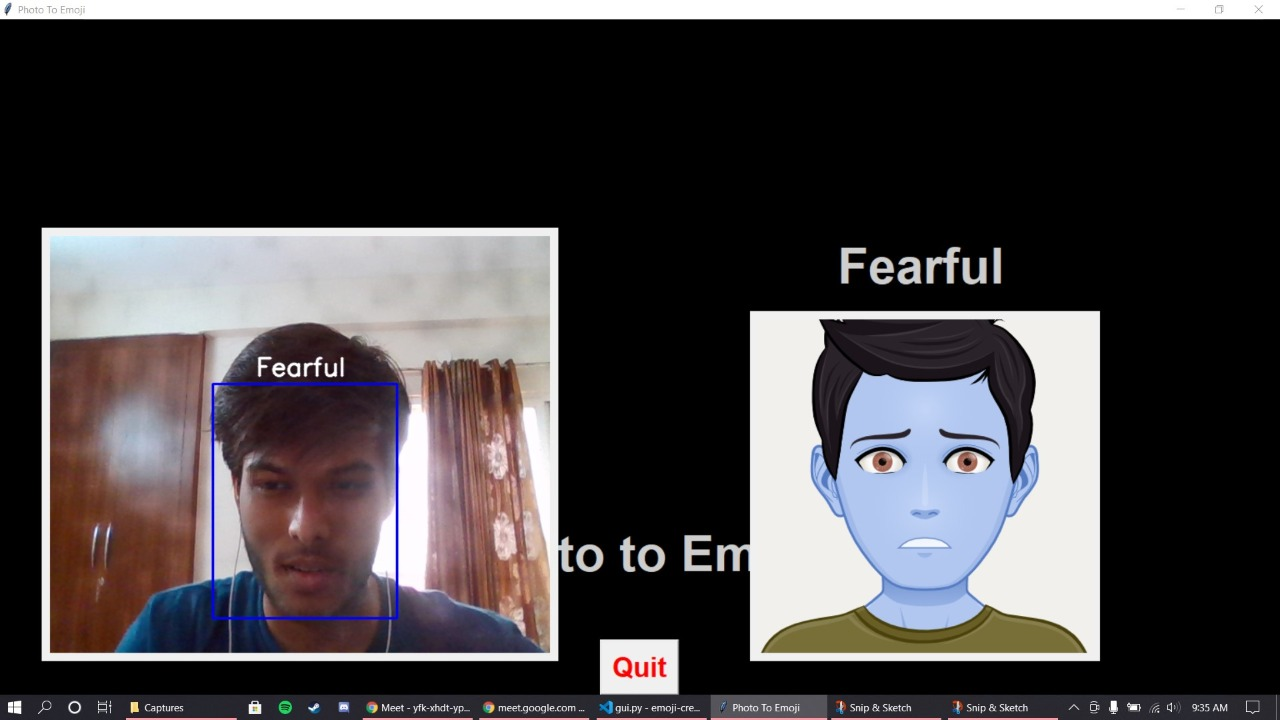
show\_vid2()

root.mainloop()

**RESULT and OBSERVATION**







As the application is executed the TensorFlow libraries are initiated and the haarcascade frontal face classifier starts running with a bounding box on the user’s face with real time prediction of their current emotion, which is done by our CNN model from trained on the FER2013 dataset. An emoji is displayed alongside the user’s face with corresponding emotion.

The prediction and presentation are done simultaneously and in real time due to which this particular environment demands for high processing speed, which unfortunately, could not be provided due to system limitations, thus a set delay was introduced.

From the above outputs we can make out that the actual image and the respective emoji predicted by the model have same emotion representation. Thus, after training the dataset the accuracy of the model is 84.56% and loss is reduced to 0.3 thus, we can expect that this model is capable enough to give somewhat accurate predictions.

The Tkinter module does its job at the trigger for opening and closing the camera source to capture frames and display them depending on the set delay.

The only drawback is, due to its hardware requirements the CPU usage tends to become high as the application opens and closes multiple times because of triggered mechanism of the application. To overcome this situation, we had to delay the execution time from 10 millisecond to 1 second. Thus, for future recommendations we can work on building the fast and efficient system.

**CONCLUSION and FUTURE RECOMMENDATION**

The use of convolutional neural networks and using the ADAM optimizer gives us an optimal result for our project to classify the prediction done by the trained model (emotion\_model.h5).

From this we can conclude that the project predicts the emotion of the user using the haarcascade frontal face classifier and the emotion is then displayed on the bounding box which produces the image of the respective emoji.

The project is quite simple to use and would be of great use in the field of machine learning in the upcoming years.

Although the model is not yet trained for day-to-day usage but our future aim is to increase the accuracy and consistency of the application.

**REFERENCES**