EmoVision- Real-time Emotion Detection and Emoji Classification Using Deep Learning

Guide: **Prof . Preeti**
Professor
School of Engineering
Computer Science-AI&ML
Malla Reddy University ,India.

K.Siddartha
B-Tech
School of Engineering
Malla Reddy University
2111cs020523@mallareddyuniversity.ac.in

K,Sindhu Reddy
B-Tech
School of Engineering
Malla Reddy University
2111cs020524@mallareddyuniversity.ac.in

R.Shivani
B-Tech
School of Engineering
Malla Reddy University
2111cs020514@mallareddyuniversity.ac.in

K.Shubham
B-Tech
School of Engineering
Malla Reddy University
2111cs020519@mallareddyuniversity.ac.in

B.Siddardha
B-Tech
School of Engineering
Malla Reddy University
2111cs020520@mallareddyuniversity.ac.in

ABSTRACT

The " EmoVision- Real-time Emotion Detection and Emoji Classification " project introduces a sophisticated system for real-time emotion recognition and emoji classification, facilitated by a convolutional neural network (CNN) model. Leveraging advanced computer vision techniques and deep learning algorithms, the system seamlessly interprets facial expressions captured via webcam feeds. The process initiates with face detection algorithms identifying faces within the video frames, followed by preprocessing to standardize facial images. These standardized images are then fed into a pretrained CNN model, meticulously trained on extensive datasets of facial expressions. This CNN architecture, comprising convolutional, pooling, and fully connected layers, efficiently extracts relevant features to classify facial expressions into seven distinct emotions: anger, disgust, fear, happiness, neutrality, sadness, and surprise. Upon emotion prediction, the system dynamically displays corresponding emojis alongside the original captured images, offering intuitive visual representations of recognized emotions. Additionally, the system offers robust features such as report generation and statistical analysis, providing comprehensive insights into emotion distribution and frequency during sessions. By harnessing TensorFlow, OpenCV, and tkinter libraries, the project delivers a seamless fusion of deep learning, image processing, and user interface components, enabling interactive experiences across various domains, including user sentiment analysis, market research, and human-computer interaction.

Keywords: Deep Learning, Convolutional Neural Network, Facial Expressions, Image Processing, Emotion Recognition

1. INTRODUCTION.

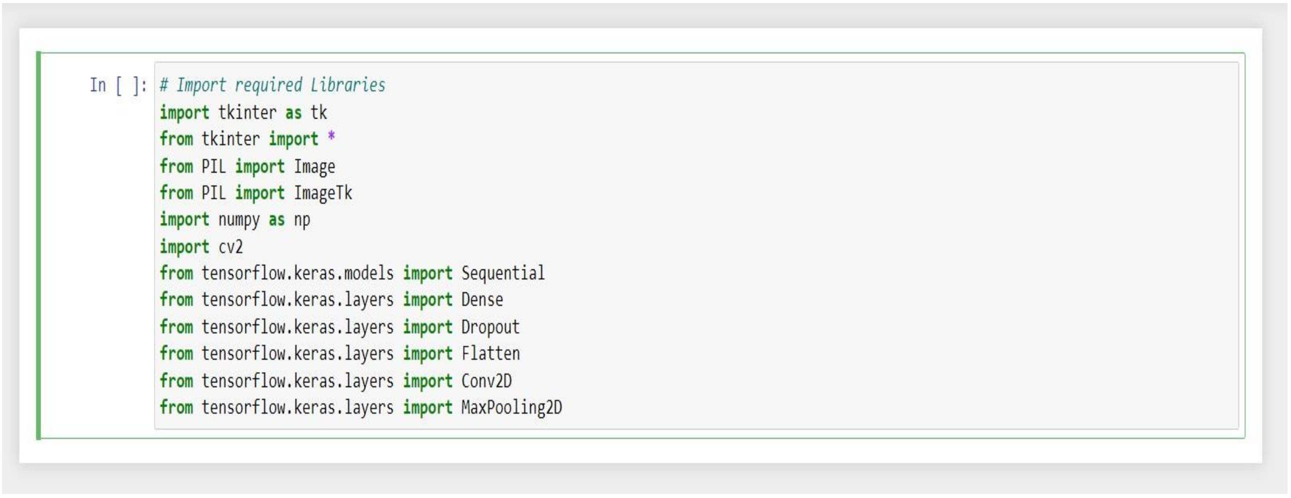
The "EmoVision- Real-time Emotion Detection and Emoji Classification" project is designed to analyse facial expressions in real-time and match them with appropriate emojis, using a combination of computer vision techniques and deep learning algorithms. Through a convolutional neural network (CNN) model trained on a dataset of labelled facial images depicting various emotions, the system can accurately identify emotions such as anger, happiness, or surprise from live video feeds captured by a webcam or video stream. This project offers an interactive user experience by overlaying emojis corresponding to detected emotions onto the user's face in real-time, facilitated by a graphical user interface (GUI). Additionally, users can capture snapshots of their expressions, and the system generates reports detailing emotion statistics for further analysis. Overall, "EmoVision- Real-time Emotion Detection and Emoji Classification" showcases the power of artificial intelligence in understanding and representing human emotions, providing a practical and engaging application for emotion recognition.

2. METHODOLOGY

A. Proposed System.

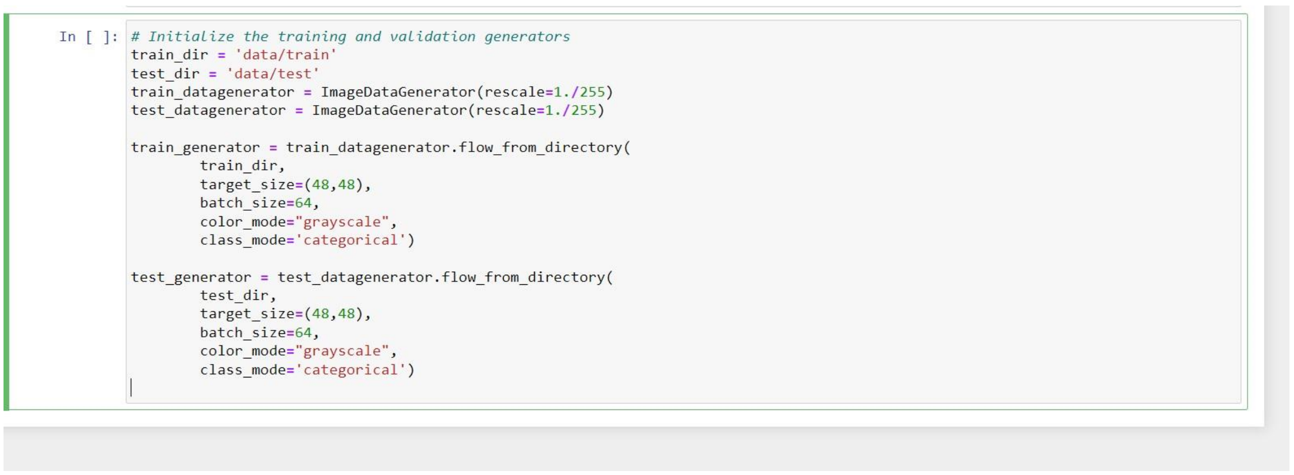
1.Data Collection: Gather a dataset of facial expression images labeled with corresponding emojis. You can use existing datasets like FER2013 create your own dataset by collecting images of different facial expressions.

-imports:



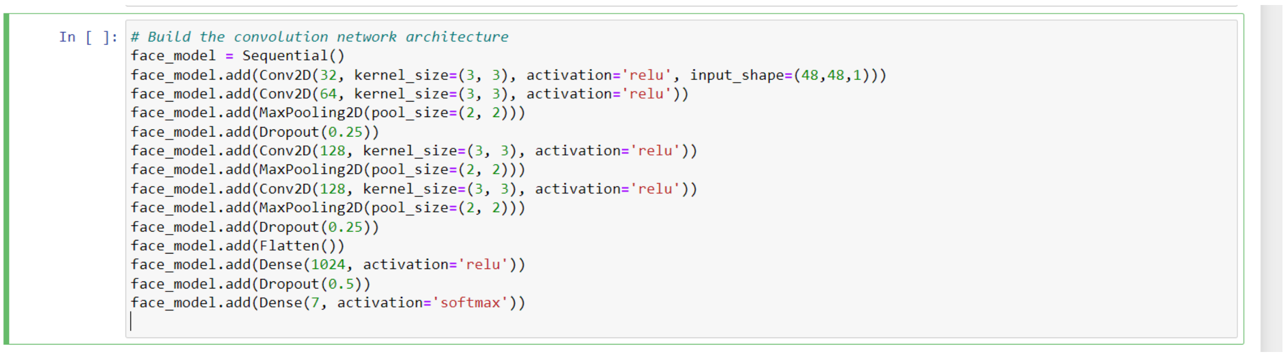
```
In [ ]: # Import required Libraries
        import tkinter as tk
        from tkinter import *
        from PIL import Image
        from PIL import ImageTk
        import numpy as np
        import cv2
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
        from tensorflow.keras.layers import Dropout
        from tensorflow.keras.layers import Flatten
        from tensorflow.keras.layers import Conv2D
        from tensorflow.keras.layers import MaxPooling2D
```

2. Initialize the training and validation generators:



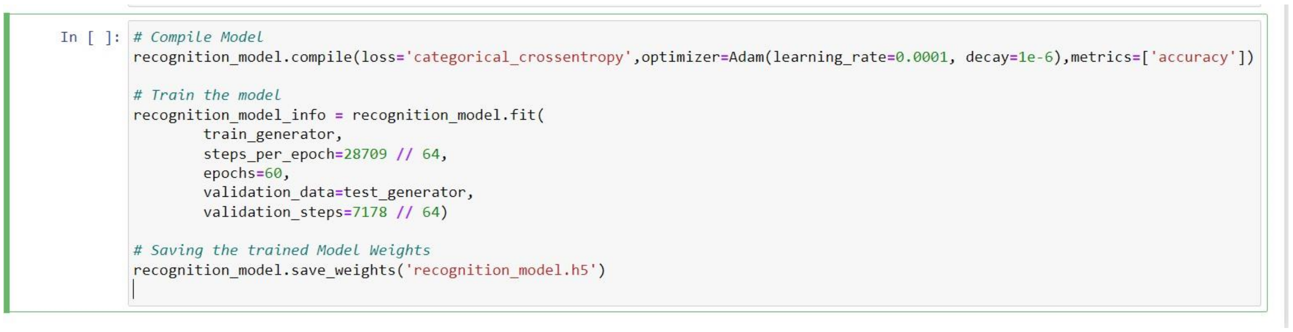
```
In [ ]: # Initialize the training and validation generators
        train_dir = 'data/train'
        test_dir = 'data/test'
        train_datagenerator = ImageDataGenerator(rescale=1./255)
        test_datagenerator = ImageDataGenerator(rescale=1./255)

        train_generator = train_datagenerator.flow_from_directory(
                train_dir,
                target_size=(48,48),
                batch_size=64,
                color_mode="grayscale",
                class_mode='categorical')

        test_generator = test_datagenerator.flow_from_directory(
                test_dir,
                target_size=(48,48),
                batch_size=64,
                color_mode="grayscale",
                class_mode='categorical')
```

3. Build the CNN Architecture.



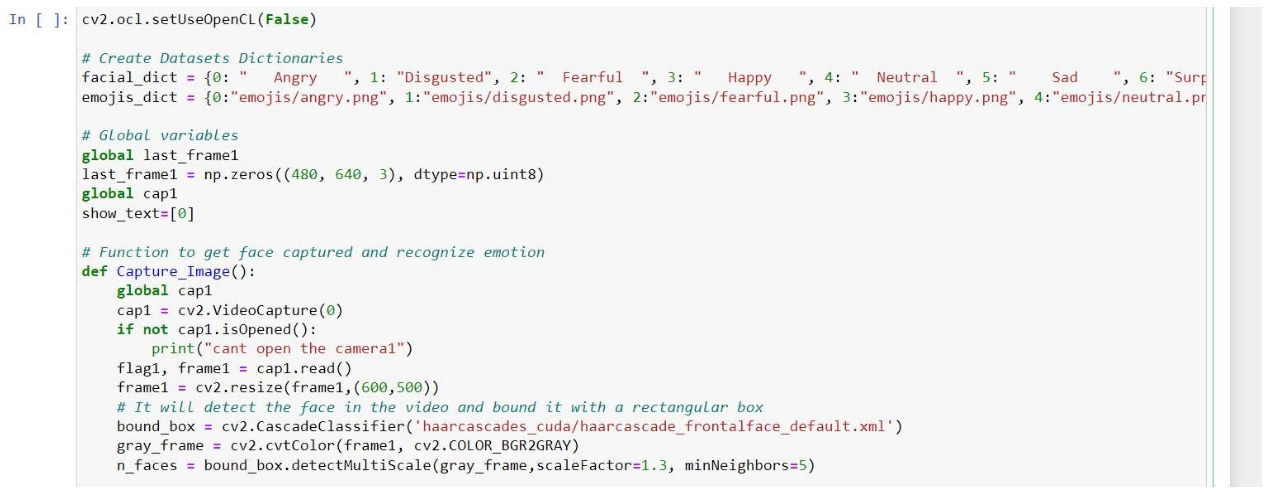
```
In [ ]: # Build the convolution network architecture
        face_model = Sequential()
        face_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
        face_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
        face_model.add(MaxPooling2D(pool_size=(2, 2)))
        face_model.add(Dropout(0.25))
        face_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
        face_model.add(MaxPooling2D(pool_size=(2, 2)))
        face_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
        face_model.add(MaxPooling2D(pool_size=(2, 2)))
        face_model.add(Dropout(0.25))
        face_model.add(Flatten())
        face_model.add(Dense(1024, activation='relu'))
        face_model.add(Dropout(0.5))
        face_model.add(Dense(7, activation='softmax'))
```

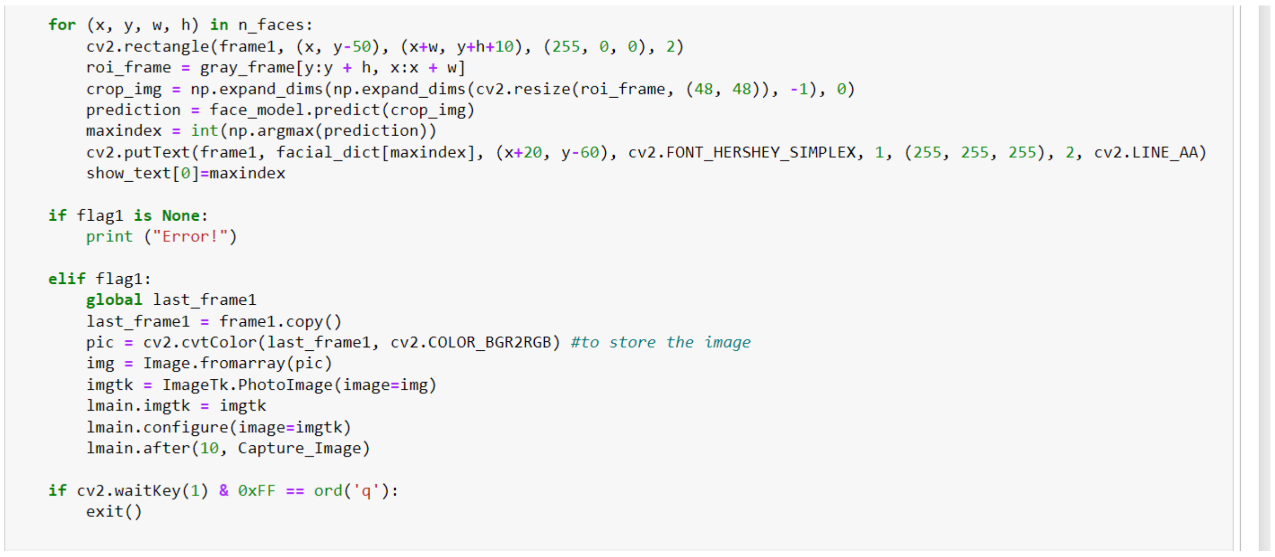
4. Compile and train the model and Save the model weights.



```
In [ ]: # Compile Model
        recognition_model.compile(loss='categorical_crossentropy',optimizer=Adam(learning_rate=0.0001, decay=1e-6),metrics=['accuracy'])

        # Train the model
        recognition_model_info = recognition_model.fit(
                train_generator,
                steps_per_epoch=28709 // 64,
                epochs=60,
                validation_data=test_generator,
                validation_steps=7178 // 64)

        # Saving the trained Model Weights
        recognition_model.save_weights('recognition_model.h5')
```

6.Predicting Emotions

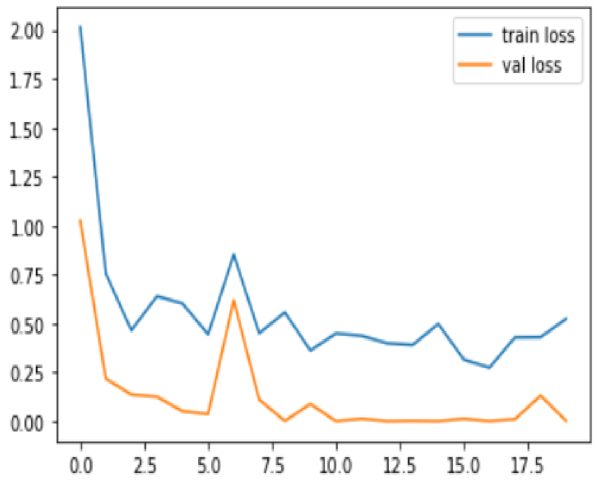

```
In [ ]: cv2.ocl.setUseOpenCL(False)

        # Create Datasets Dictionaries
        facial_dict = {0: "   Angry    ", 1: "Disgusted", 2: "  Fearful   ", 3: "    Happy    ", 4: "  Neutral  ", 5: "    Sad    ", 6: "Surp
        emojis_dict = {0:"emojis/angry.png", 1:"emojis/disgusted.png", 2:"emojis/fearful.png", 3:"emojis/happy.png", 4:"emojis/neutral.pr

        # Global variables
        global last_frame1
        last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
        global cap1
        show_text=[0]

        # Function to get face captured and recognize emotion
        def Capture_Image():
            global cap1
            cap1 = cv2.VideoCapture(0)
            if not cap1.isOpened():
                print("cant open the camera1")
            flag1, frame1 = cap1.read()
            frame1 = cv2.resize(frame1,(600,500))
            # It will detect the face in the video and bound it with a rectangular box
            bound_box = cv2.CascadeClassifier('haarcascades_cuda/haarcascade_frontalface_default.xml')
            gray_frame = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
            n_faces = bound_box.detectMultiScale(gray_frame,scaleFactor=1.3, minNeighbors=5)
```


```
    for (x, y, w, h) in n_faces:
        cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
        roi_frame = gray_frame[y:y + h, x:x + w]
        crop_img = np.expand_dims(np.expand_dims(cv2.resize(roi_frame, (48, 48)), -1), 0)
        prediction = face_model.predict(crop_img)
        maxindex = int(np.argmax(prediction))
        cv2.putText(frame1, facial_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 ("Error!")

    elif flag1:
        global last_frame1
        last_frame1 = frame1.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB) #to store the image
        img = Image.fromarray(pic)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
        lmain.after(10, Capture_Image)

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

B. Developing GUI and mapping with Emojis:

Create a folder named emojis and save the emojis corresponding to each of the seven emotions in the dataset. The trained model is tested on a set of images. Random images are introduced to the network and the output label is compared to the original known label of the image. Parameters used for evaluation are F1 score, precision and recall. Precision is the proportion of predicted positives that are truly positives.

-Testing:



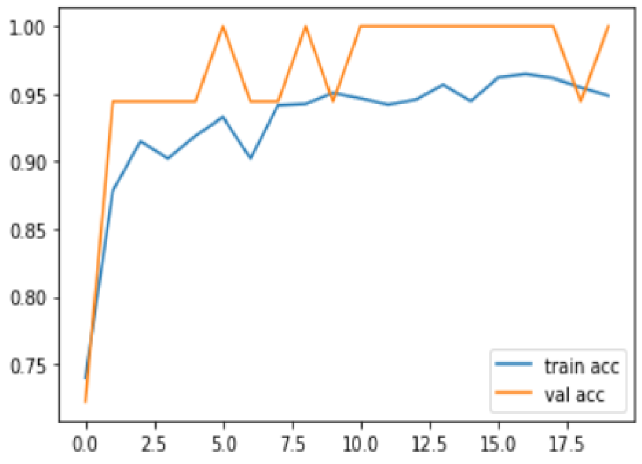
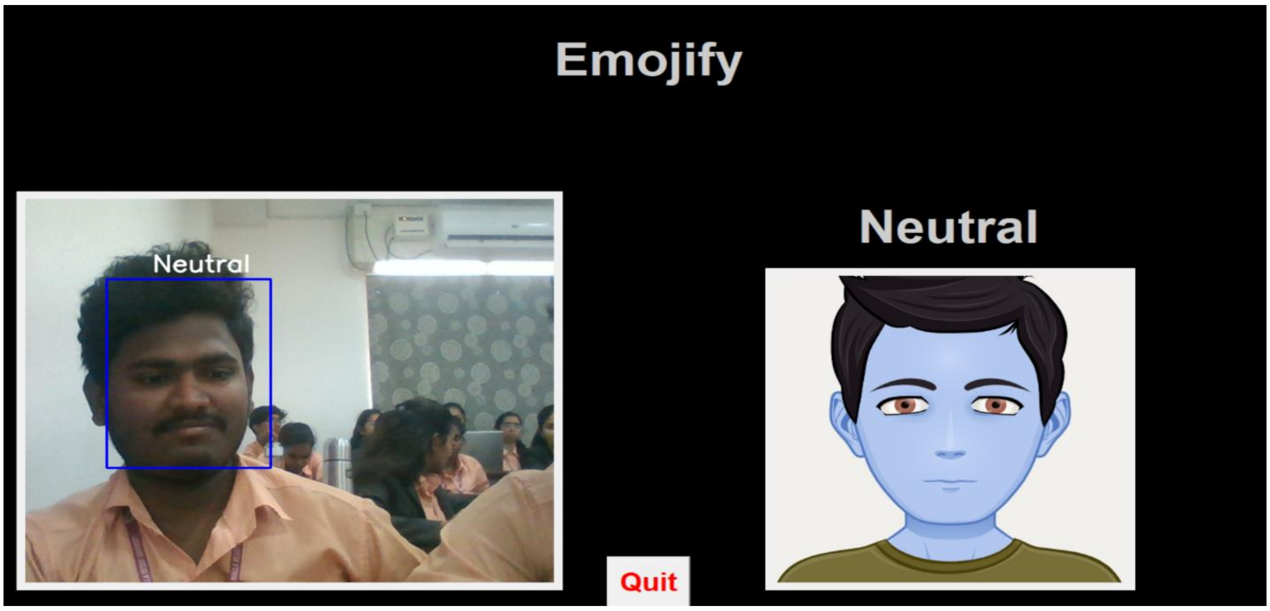
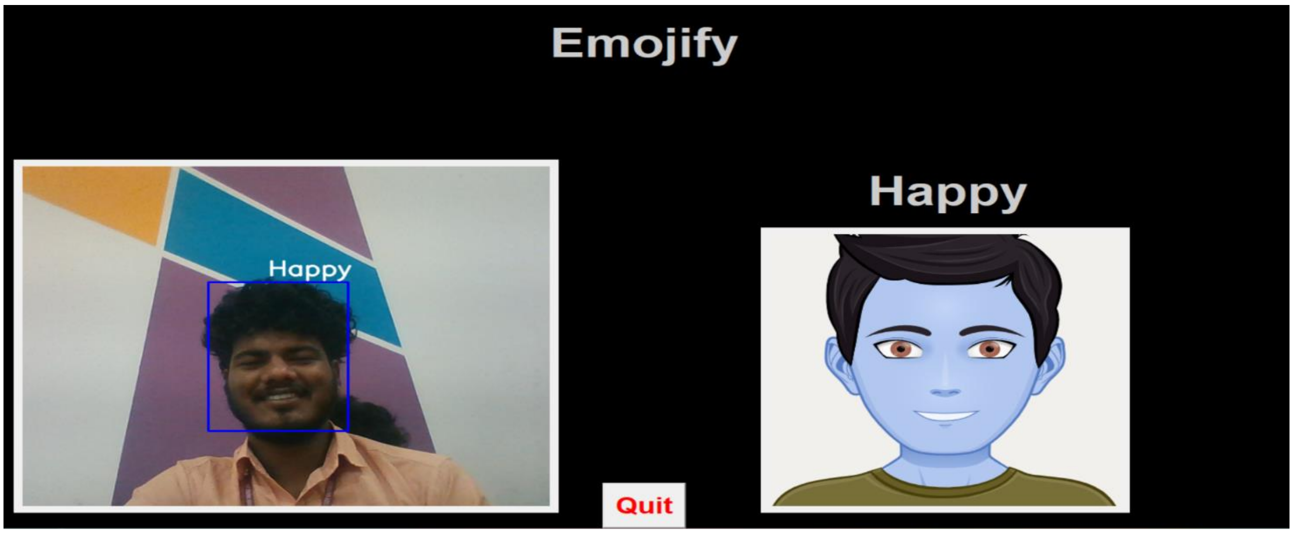


Figure 6: Loss Plot Figure 7: Accuracy Plot

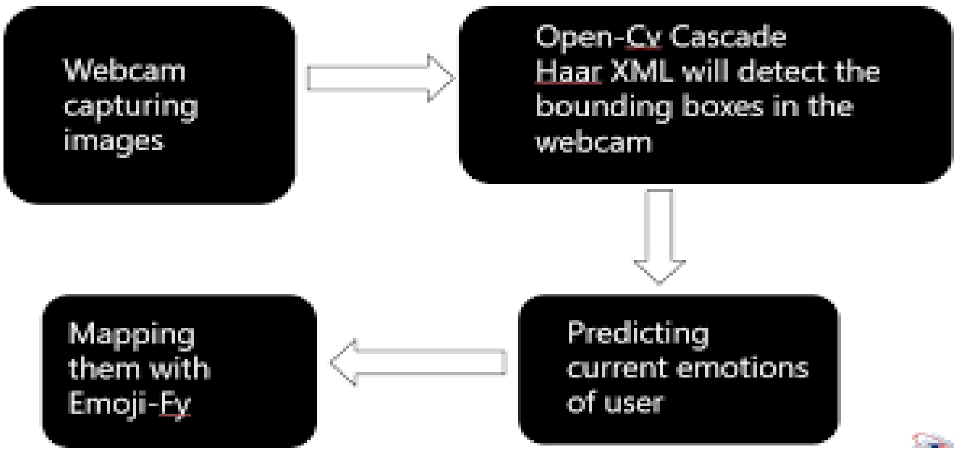
-Final Output:

Here are some images of how will this project look. This dataset consists of facial emotions of following categories: 0:angry 1:disgust 2:feat 3:happy 4:sad 5:surprise 6:natural.





3. BLOCK DIAGRAM


Webcam capturing images
Open-Cv Cascade Haar XML will detect the bounding boxes in the webcam
Predicting current emotions of user
Mapping them with Emoji-Fy

(Reference-google images)

4.TOOLS USED

• We have used diverse records technology associated libraries like keras, TensorFlow, OpenCV, NumPy , Tkinter , pandas ,PIL etc. For the logic mapping need intermediate knowledge of CNN and RNN.

• For Code Running and Development use environments like VS Code , Jupyter Notebook or colab . most Important thing is you need to install all pip files which you imported in project code

5.CONCLUSION

As Today's generation people is loving the trend of communicating with non-verbal cues like emoticons so we thought why not bring out our own emojis.

With advancements in computer vision and deep learning, we will now able to detect human emotions from images. In this deep learning project, we will classify human facial expressions to filter and map corresponding emojis or avatars.

The result we are expected is the use of EmoVision in chatting world. We want people to communicate with their own customisable emoticon. The project will recognize one's current emotion and convert that emotion's emoji so that the customer gets emoji of their face and use it in chatting.

6.REFERENCES

1. R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018.
2. https://ijcrt.org/papers/IJCRT2101596.pdf
3. https://www.irjmets.com/uploadedfiles/paper//issue_5_may_2023/40298/final/fin_irjmets1685176934.pdf
4. https://www.irjet.net/archives/V8/i11/IRJET-V8I11164.pdf
5. https://data-flair.training/blogs/create-emoji-with-deep-learning/
6. I. Zafeiriou, S.; Zhang, C.; Zhang, Z. A survey on face detection in the wild: Past, present and future. Comput. Vis. Image Underst. 2015, 138, 1–24.