PROJECT REPORT

EMOTION-BASED MUSIC PLAYER

Team Members

Bhargava Rathod (bhargavarathod@my.unt.edu)

SaiPrathyushaVeguru (saiprathyushaveguru@my.unt.edu)

Sheshidhar Reddy Shaga (sheshidharreddyshaga@my.unt.edu)

Sai Meghana Eravelli (saimeghanaeravelli@my.unt.edu)

Likith Chowdary Nuthi (likithchowdarynuthi@my.unt.edu)

Contents:

1. Abstract: The core objective of this project is to detect facial emotion and based on the detected emotion, we are randomly playing a song that is in the recommended playlist. In the first place, we have trained a deep learning image classification model which will detect the facial emotion, and then we are passing webcam frames to a trained model which will detect the facial emotion from the frames. We collect N frames from the webcam, and we are taking the majority vote from the N detected frames, and based on the most detected emotion, music will be played.

2. Data specification: We have used the dataset which was downloaded from the Kaggle repository (https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset). The dataset contains 7 classes which are "Angry, Fear, Sad, Disgust, Surprise, Happy, Neutral". From the 7 classes, we have picked only 4 classes, Angry, Happy, Sad, and Neutral which are most relevant to our objective.

3. Design and Milestones: Our whole project can be categorized into 2 modules

3.1. Training a Model: In this section, we have trained multiple models from which we have used the best model. Here we have used google collab since it is an image classification task it requires more processing and GPU power.

- We first tried with vgg16 without transfer learning, on top of it we have added our own layers. This model failed and gave us just around 50% accuracy.
- In model 2 we have trained using inceptionv3 without using the transfer learning approach. Here also the model behaved the same as the previous model. Accuracy was around 50%. Hence, we understood that the pre-trained weights of the model are not performing well. So, we decided to try using the transfer learning approach.
- In the 3rd model we have tried using the same model 1 architecture, but the only difference is here we have applied transfer learning on VGG16 + own layers. This model worked well and gave us train accuracy 70.07% and test accuracy 66.85%. If we train for more epochs the model is overfitting.
- Later we have tried multiple approaches like inceptionv3 transfer learning. But still, it's not giving better results than the previous model. We have also tried various approaches, but nothing worked better.

3.2. Detecting emotion from webcam: In this section, we have used Jupiter notebook. As we have already trained the model, it should not require more processing power, jupyter notebook works fine.

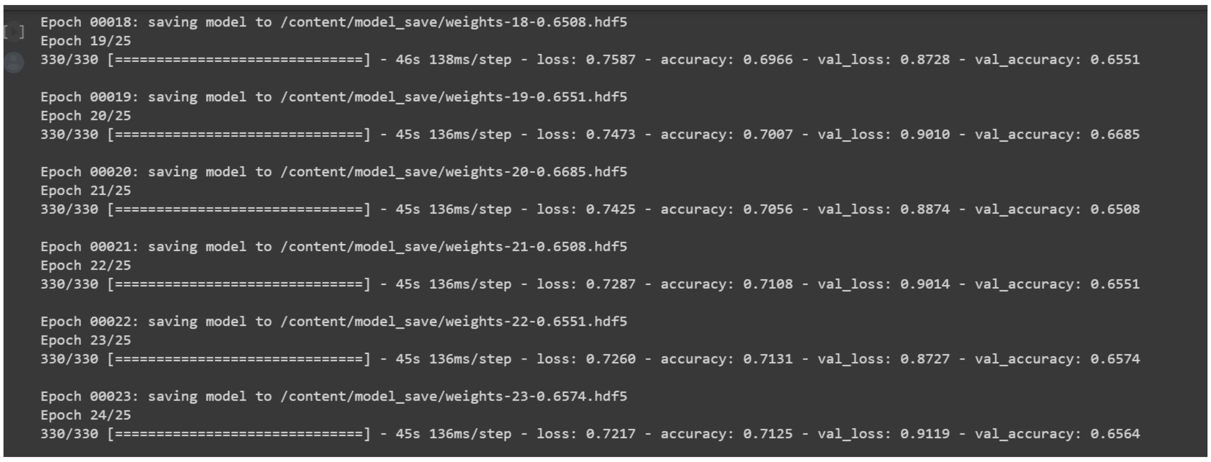
Here we will take frames from the OpenCV library and feed the frames to the trained model. Per each frame, the model will predict 1 emotion. In this code file, we wrote 3 functions.

- Emotion detection for an Image
- Emotion detection for video through webcam.
- Based on the detected image playing music.

4. Results and Analysis:

As already discussed in the design section we got 70% train accuracy and 66% test accuracy. Detecting an emotion is always a difficult task as the margin of difference in emotions is very less and with the limited resources, we couldn't train powerful. The good thing about the project is we are not only defending on 1 frame. We will pass approximately 30 or 40 frames and out of all detections we have chosen a majority voting approach where the most detected emotion will be the final detected emotion.

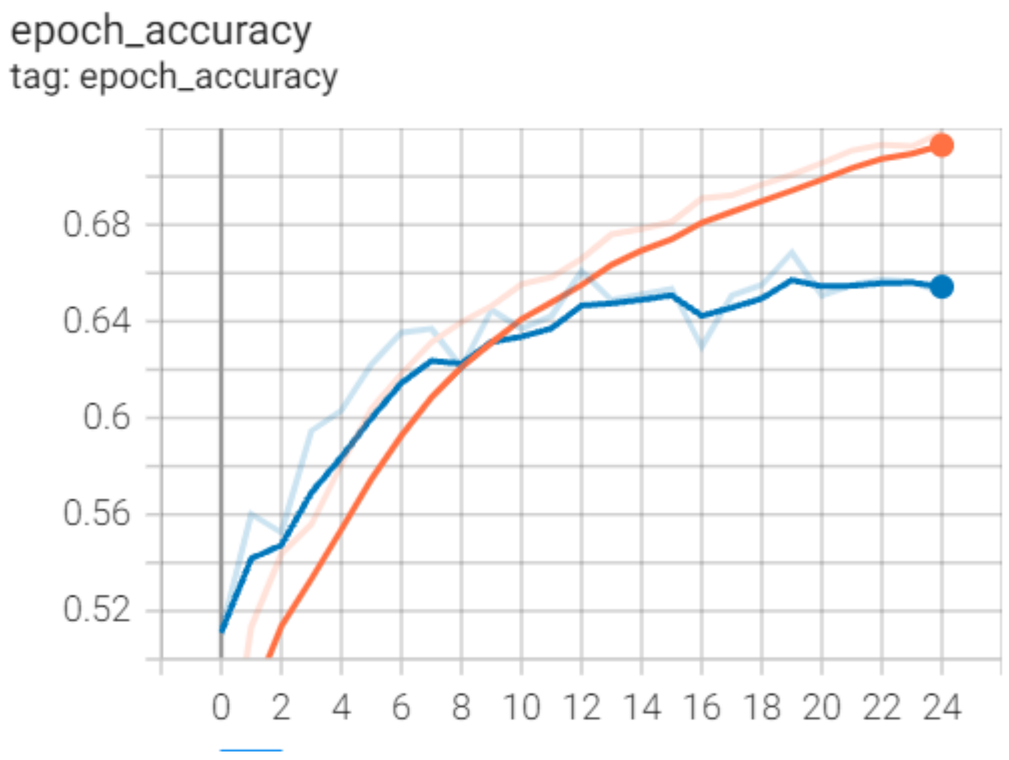
<u>Finding the results in the below screenshot, after training the model</u>.



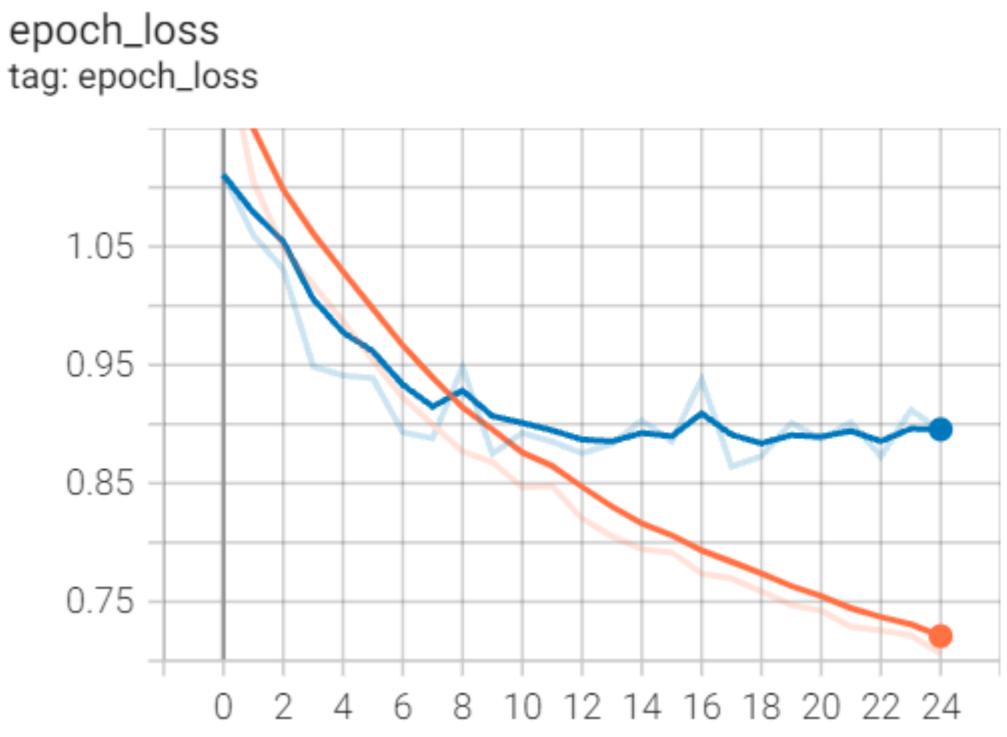
```
Epoch 00018: saving model to /content/model_save/weights-18-0.6508.hdf5
Epoch 19/25
330/330 [==============================] - 46s 138ms/step - loss: 0.7587 - accuracy: 0.6966 - val_loss: 0.8728 - val_accuracy: 0.6551

Epoch 00019: saving model to /content/model_save/weights-19-0.6551.hdf5
Epoch 20/25
330/330 [==============================] - 45s 136ms/step - loss: 0.7473 - accuracy: 0.7007 - val_loss: 0.9010 - val_accuracy: 0.6685

Epoch 00020: saving model to /content/model_save/weights-20-0.6685.hdf5
Epoch 21/25
330/330 [==============================] - 45s 136ms/step - loss: 0.7425 - accuracy: 0.7056 - val_loss: 0.8874 - val_accuracy: 0.6508

Epoch 00021: saving model to /content/model_save/weights-21-0.6508.hdf5
Epoch 22/25
330/330 [==============================] - 45s 136ms/step - loss: 0.7287 - accuracy: 0.7108 - val_loss: 0.9014 - val_accuracy: 0.6551

Epoch 00022: saving model to /content/model_save/weights-22-0.6551.hdf5
Epoch 23/25
330/330 [==============================] - 45s 136ms/step - loss: 0.7260 - accuracy: 0.7131 - val_loss: 0.8727 - val_accuracy: 0.6574

Epoch 00023: saving model to /content/model_save/weights-23-0.6574.hdf5
Epoch 24/25
330/330 [==============================] - 45s 136ms/step - loss: 0.7217 - accuracy: 0.7125 - val_loss: 0.9119 - val_accuracy: 0.6564
```

In the whole training process, we have saved the weights for each epoch by using the concept of checkpoints.

Epoch VS Accuracy Plot



Epoch VS Loss Plot

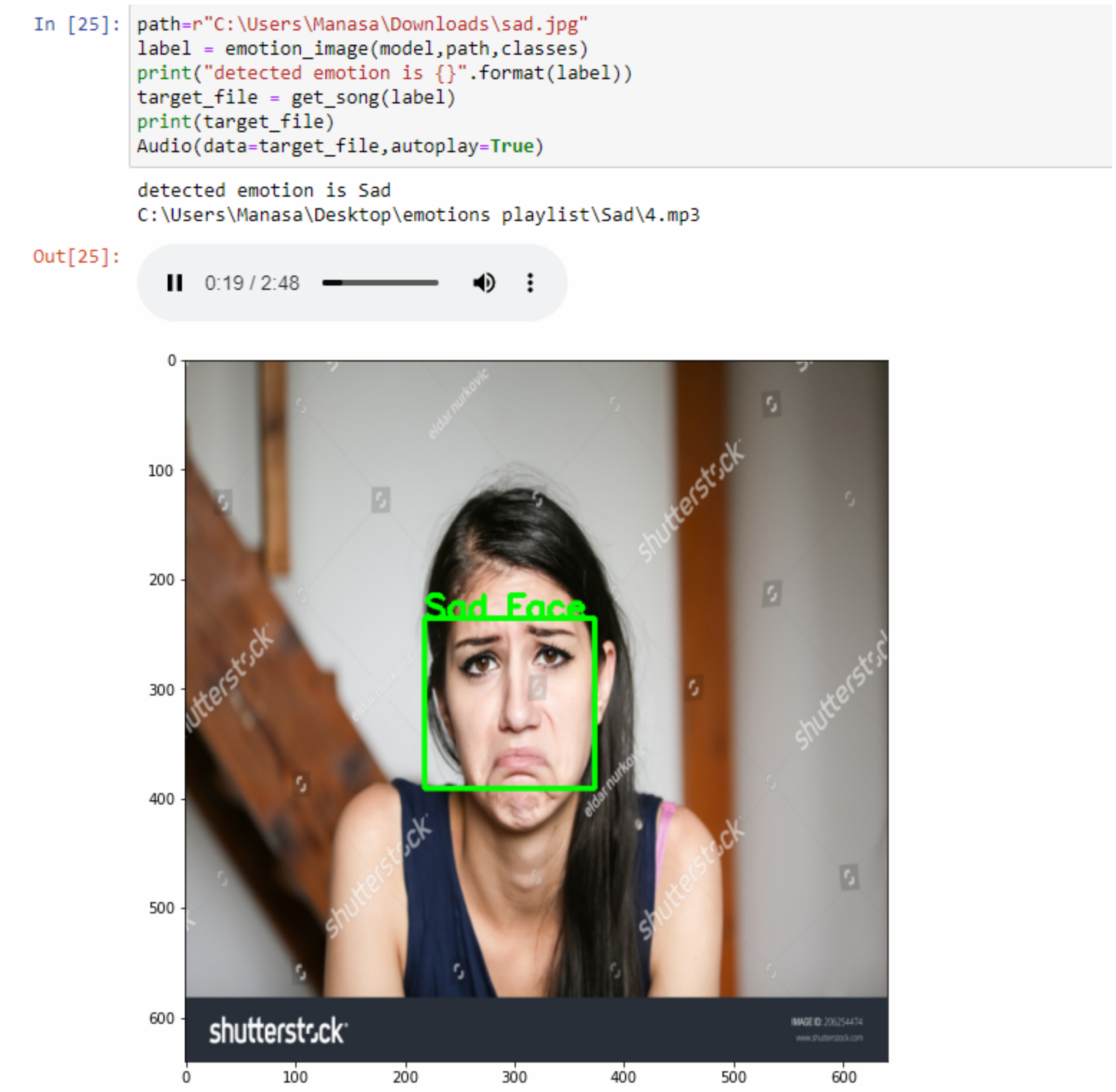


From the above plots we can observe that as we train for more epochs, the model is overfitting to the training dataset. Hence, we picked reasonable weights.

We have chosen Jupyter Notebook because it comes with a good UI to plot the images and OpenCV also provides a separate window to display the video. Hence, we have not deployed it into any web application.

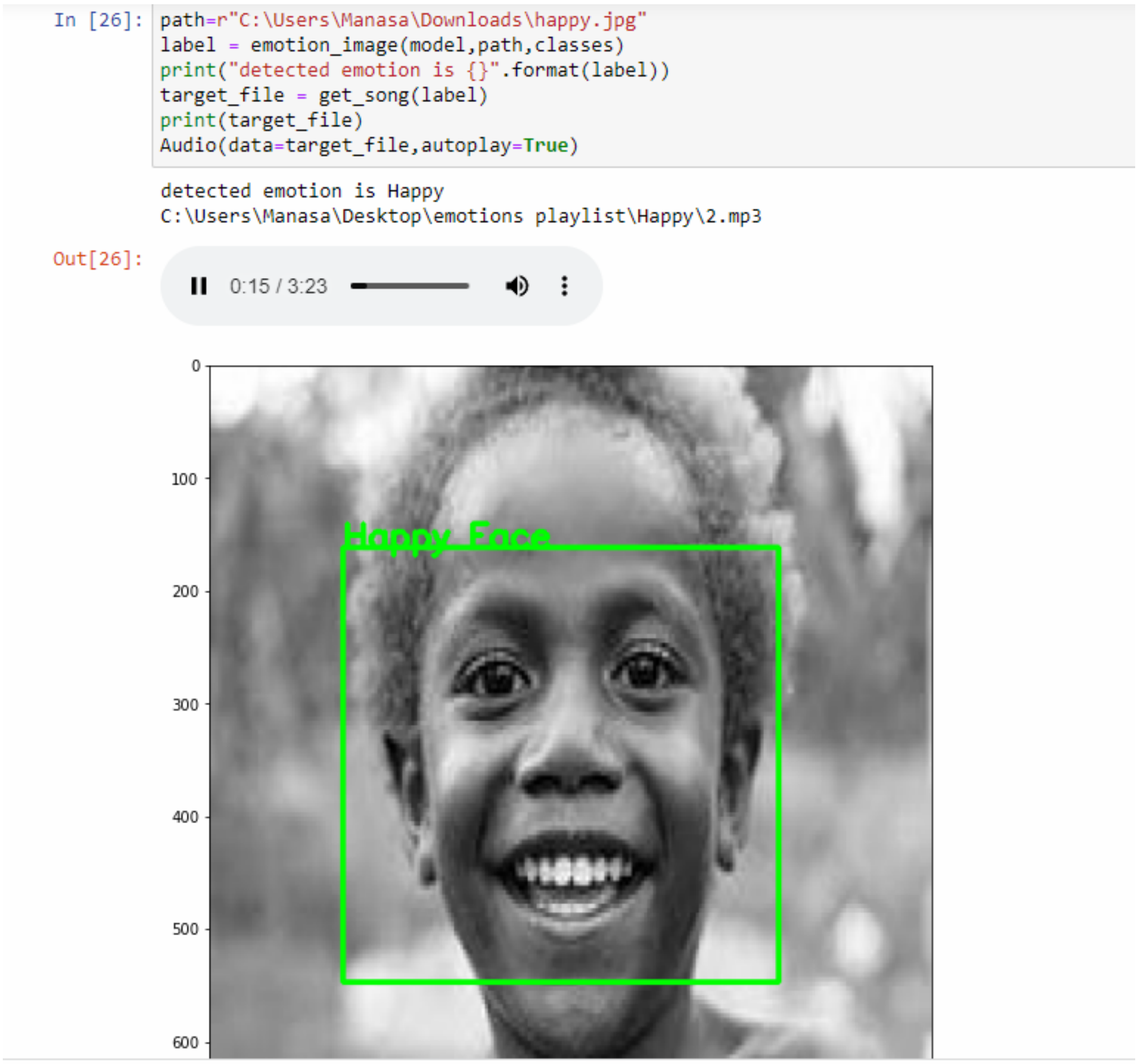
Emotions detected and playing songs bypassing input image:

SAD:



```
In [25]: path=r"C:\Users\Manasa\Downloads\sad.jpg"
         label = emotion_image(model,path,classes)
         print("detected emotion is {}".format(label))
         target_file = get_song(label)
         print(target_file)
         Audio(data=target_file,autoplay=True)

         detected emotion is Sad
         C:\Users\Manasa\Desktop\emotions playlist\Sad\4.mp3

Out[25]:
```

Happy:

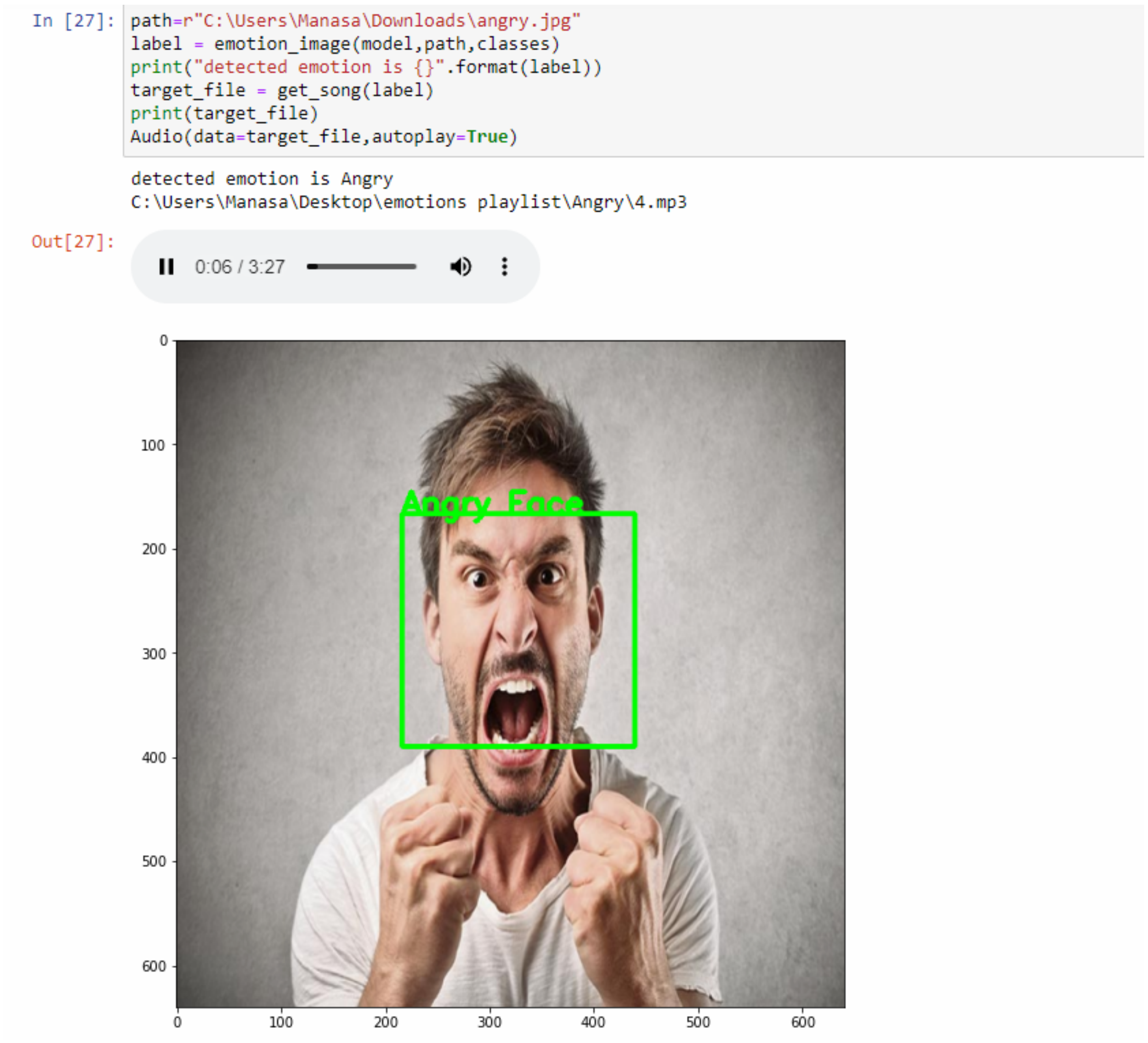


```
In [26]: path=r"C:\Users\Manasa\Downloads\happy.jpg"
         label = emotion_image(model,path,classes)
         print("detected emotion is {}".format(label))
         target_file = get_song(label)
         print(target_file)
         Audio(data=target_file,autoplay=True)
```

```
detected emotion is Happy
C:\Users\Manasa\Desktop\emotions playlist\Happy\2.mp3
```

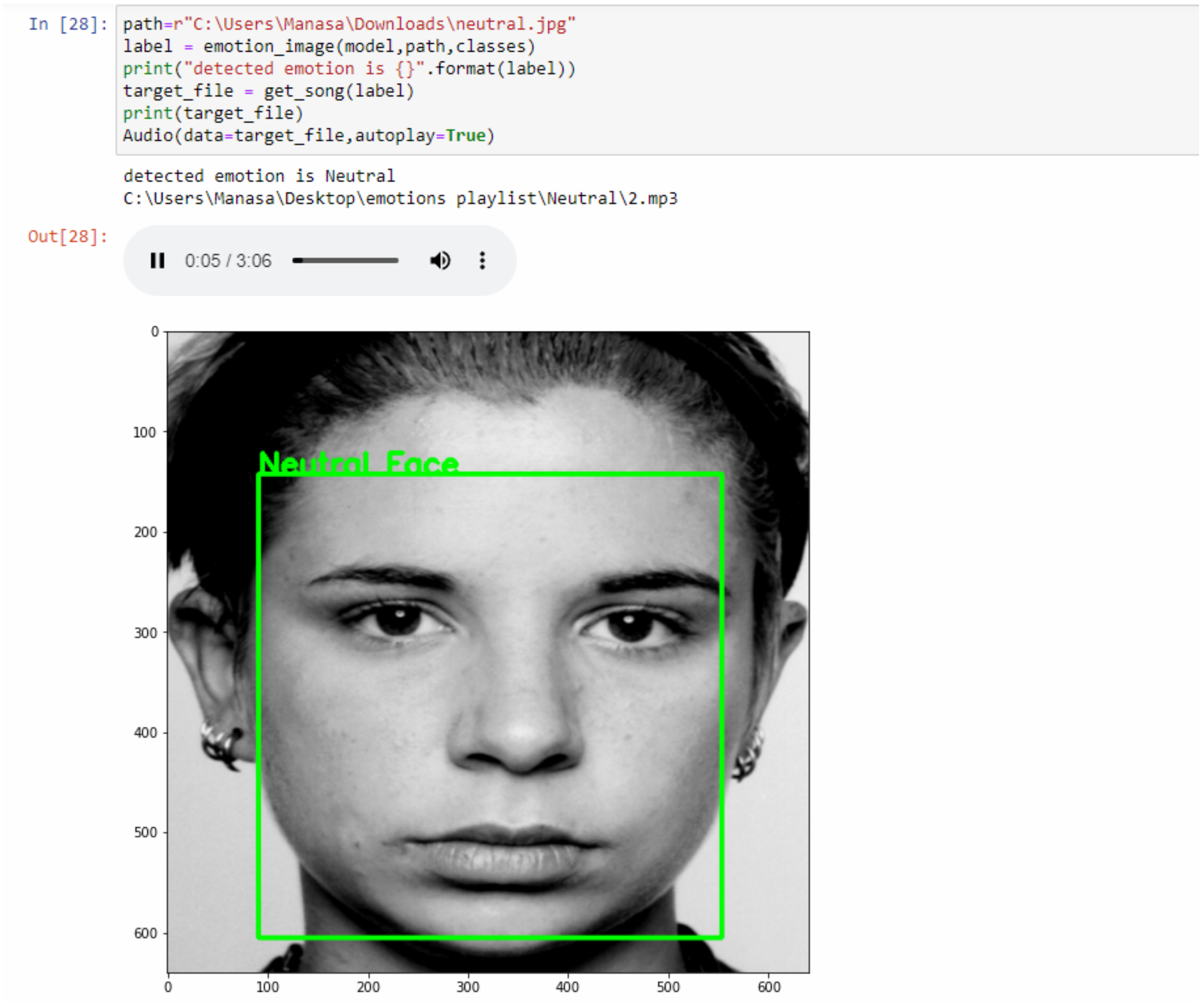
```
Out[26]:
```

Angry:



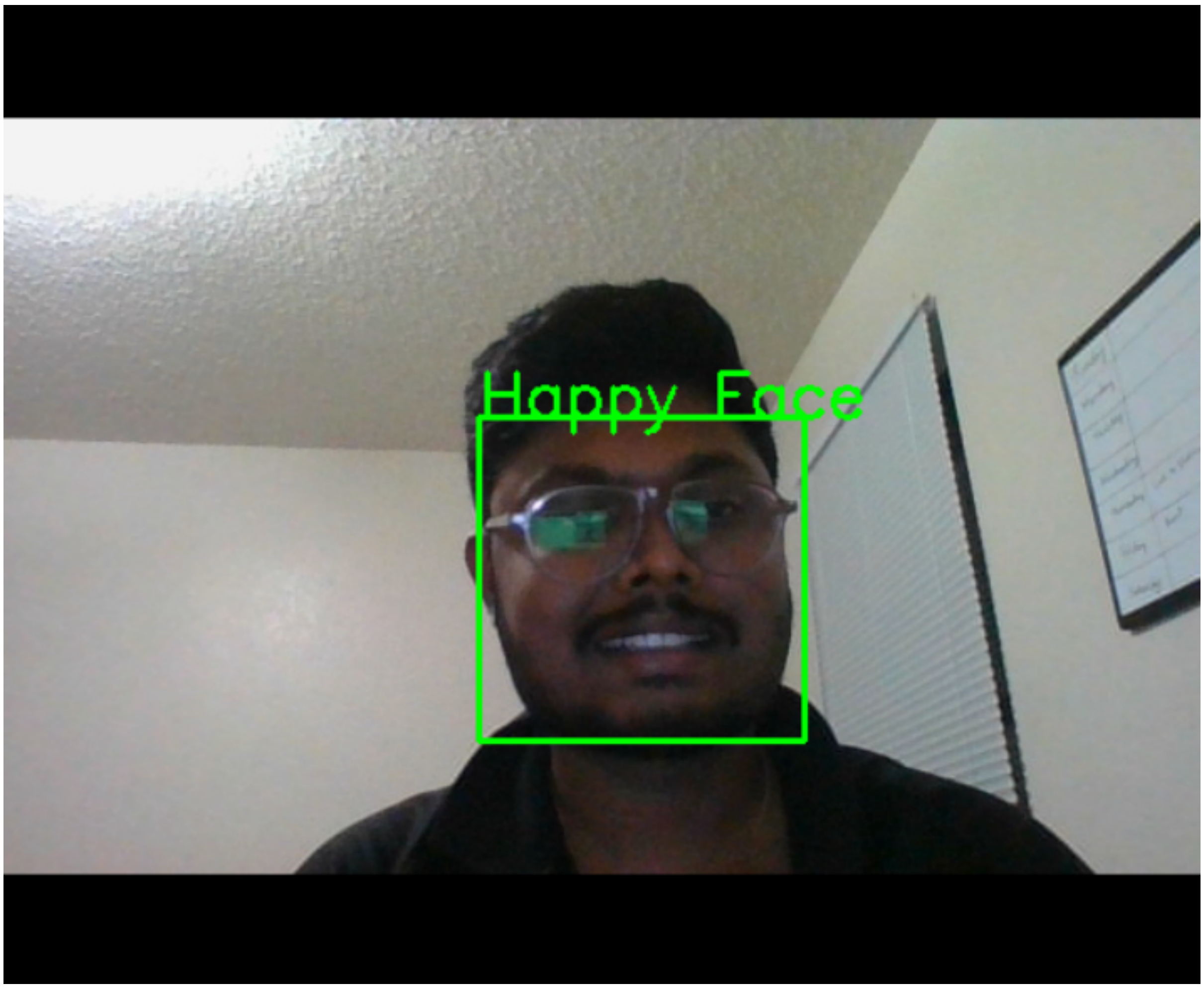
```
In [27]: path=r"C:\Users\Manasa\Downloads\angry.jpg"
         label = emotion_image(model,path,classes)
         print("detected emotion is {}".format(label))
         target_file = get_song(label)
         print(target_file)
         Audio(data=target_file,autoplay=True)

         detected emotion is Angry
         C:\Users\Manasa\Desktop\emotions playlist\Angry\4.mp3

Out[27]:
```

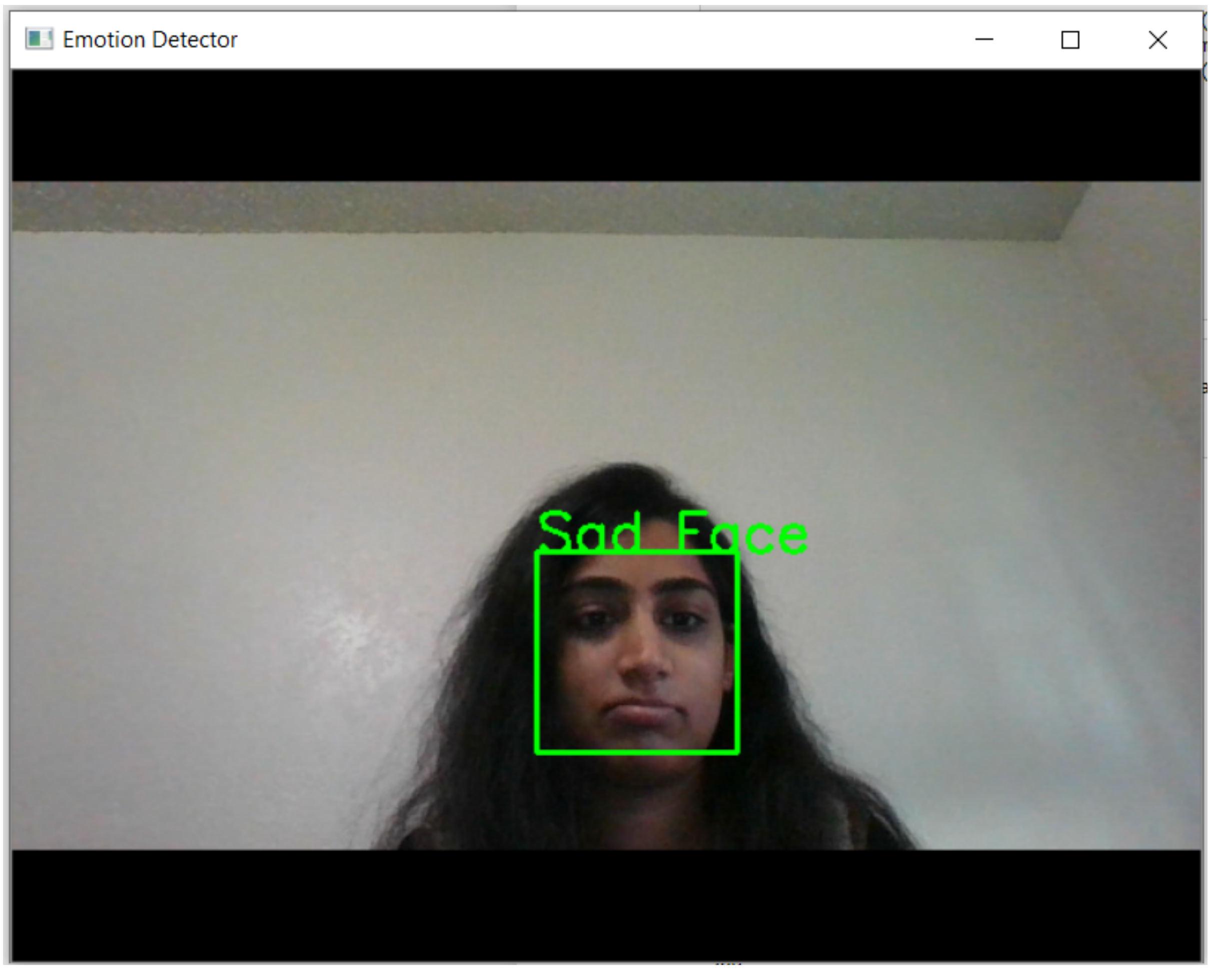
Neutral:

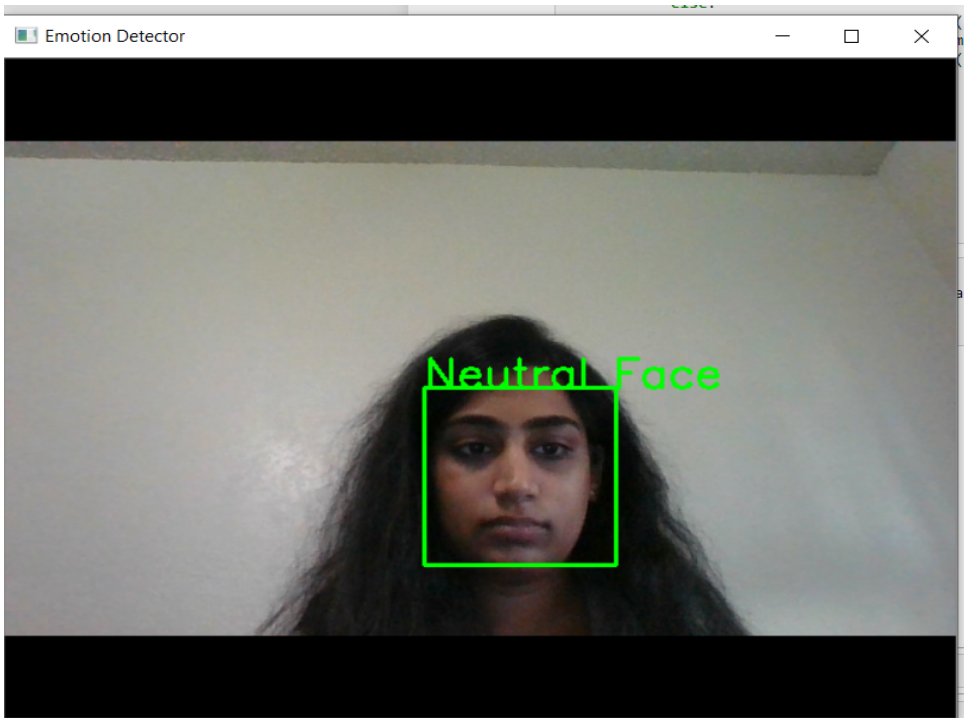


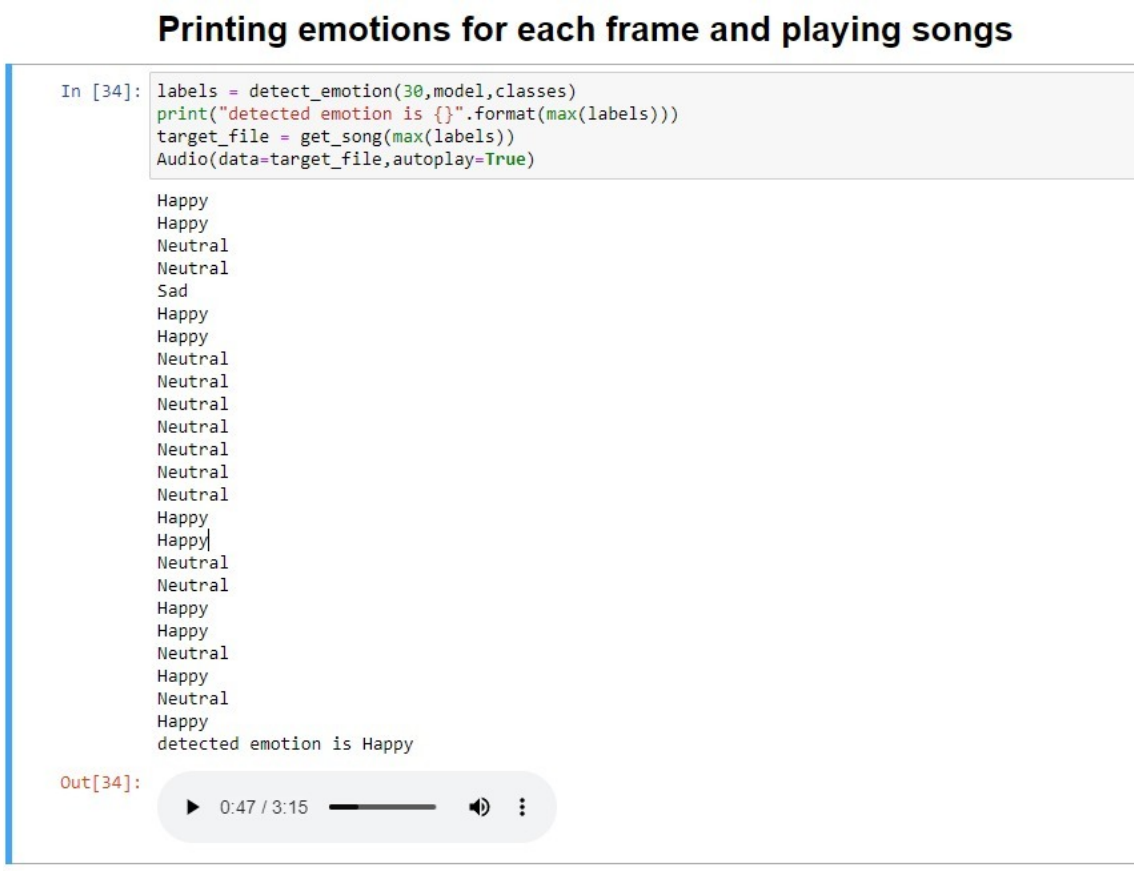
```
In [28]: path=r"C:\Users\Manasa\Downloads\neutral.jpg"
         label = emotion_image(model,path,classes)
         print("detected emotion is {}".format(label))
         target_file = get_song(label)
         print(target_file)
         Audio(data=target_file,autoplay=True)

         detected emotion is Neutral
         C:\Users\Manasa\Desktop\emotions playlist\Neutral\2.mp3
Out[28]:
```

Emotion Detection from WEB-CAM




Emotion Detector
Sad Face


Emotion Detector
Neutral Face



Printing emotions for each frame and playing songs

```
In [34]: labels = detect_emotion(30,model,classes)
         print("detected emotion is {}".format(max(labels)))
         target_file = get_song(max(labels))
         Audio(data=target_file,autoplay=True)

         Happy
         Happy
         Neutral
         Neutral
         Sad
         Happy
         Happy
         Neutral
         Neutral
         Neutral
         Neutral
         Neutral
         Neutral
         Neutral
         Happy
         Happy
         Neutral
         Neutral
         Happy
         Happy
         Neutral
         Happy
         Neutral
         Happy
         detected emotion is Happy

Out[34]: 0:47 / 3:15
```

5. Repository / Achieve:

https://github.com/bhargavarathod/emotion_based_musicplayer.git

6. CODE:

Emotion_model_training_code:
https://colab.research.google.com/drive/1lPm758yq79HnKT0lSSrhckacUpxYsHwE?usp=sharing

Emotion_detection_code:

https://github.com/bhargavarathod/emotion_based_musicplayer/blob/dbe8fa66368f18deff0c998dc2280997cc3b4a93/emotion_detection.ipynb