Emotion Based Music Player

Team Members:

Bhargava Rathod

SaiPrathyusha Veguru

Sheshidhar Reddy Shaga

Likith Nuthi

Sai Meghana Eravelli

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.

Data specification

We have used the dataset which was downloaded from the Kaggle repository (https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset).

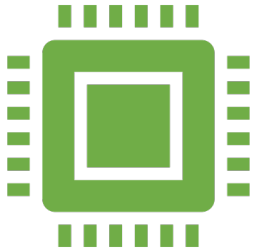
The dataset contains 7 classes which are “Angry, Fear, Sad, Disgust, Surprise, Happy, Neutral”. From the 7 classes, we have picked only 4 classes, Angry, Happy, Sad, and Neutral which are most relevant to our objective.

It has Train dataset of 21,077 images. And Validation dataset of 5140 images.

Image size of 48X48.

All images are in gray-scale.

We have trained the model using Tensor flow.



We have used Google Collab to compute the model.

We have used Jupyter Notebooks to capture the frames.

Training a Model

- 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 accuracy of 56% and test accuracy of 50% .
- 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.

Detecting emotion from webcam

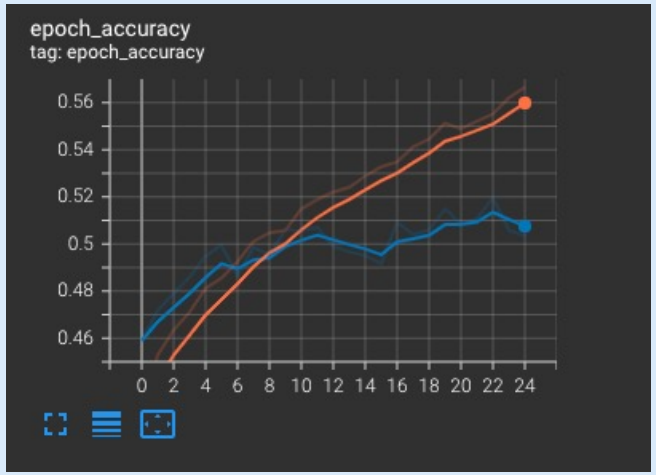
Here we will take frames from the OpenCV library and feed the frames to the trained model. Per each frame, the model will predict 1 emotion. In this code file, we wrote 3 functions.

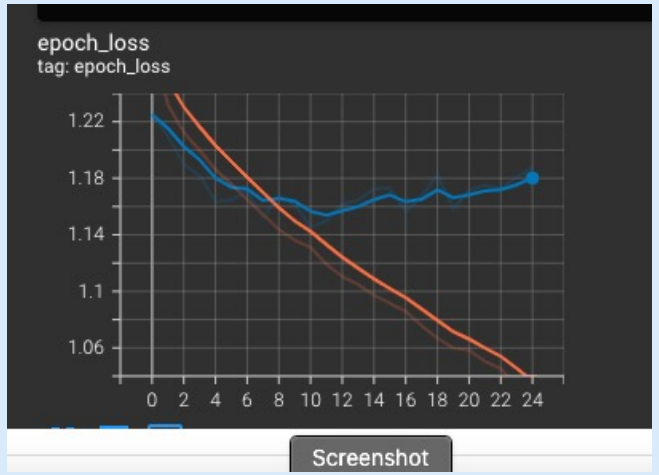
Emotion detection for an Image

Emotion detection for video through webcam.

Based on the detected image playing music.

Training Model 1





- Here we have used VGG16 pretrained model and with the combination of our Convolution, Maxpool and Dense layers
- We are not training the VGG16 weights and made "vgg.trainable = False".
- Since VGG16 model was trained on ImageNet dataset. The weights of VGG16 will be good enough to understand the Images.
- We have trained the model for 25 epochs.
- This model is clearly not performing well.
- Accuracy = 56%,
- Test Accuracy = 50%.

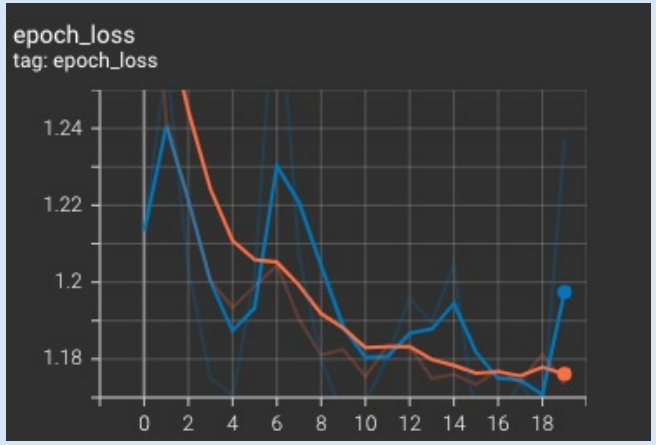
Model 1

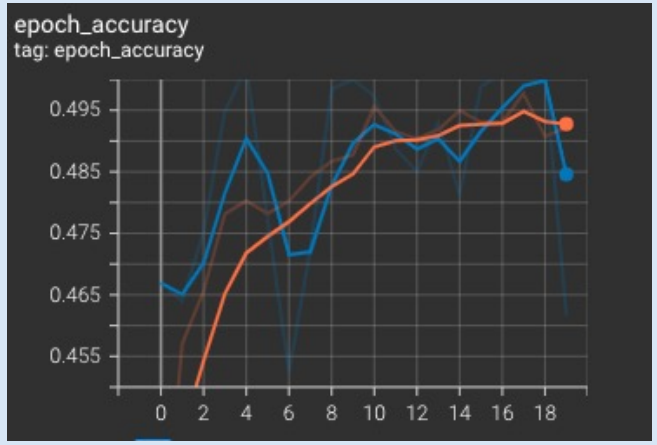
```
[ ] vgg = VGG16(include_top=False, weights='imagenet')
    vgg.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
```

```
[ ] input_layer = Input(shape=(targetsize,targetsize,3,))
    vgg_layer = vgg(input_layer)
    layer1 = Conv2D(filters=256,kernel_size=(3,3),strides=(1,1),padding="same",activation='relu',
                        kernel_initializer=tf.keras.initializers.he_normal(seed=20))(vgg_layer)
    layer2 = MaxPool2D(pool_size=(2, 2), strides=(1,1), padding="same")(layer1)
    flatten_layer = Flatten()(layer2)
    layer3 = Dense(128,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=24))(flatten_layer)
    layer4 = Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20))(layer3)
    output = Dense(4,activation='softmax',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=24))(layer4)
    model_1 = Model(inputs=input_layer,outputs=output)
    model_1.summary()
```

Training Model 2





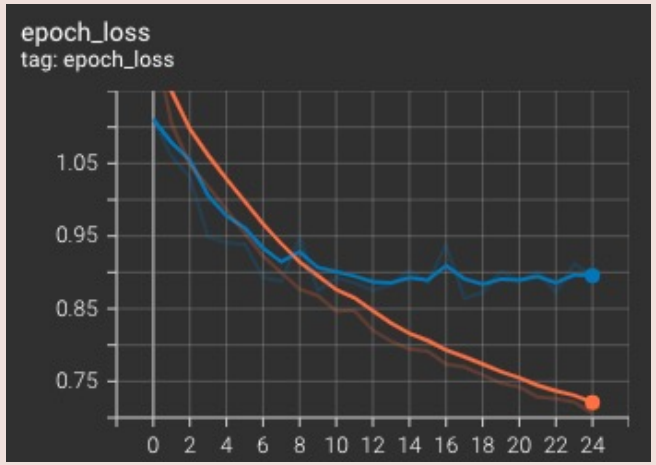
- As the previous model performance is less, here we are using the inceptionv3 model.
- Here image size of the dataset is 48X48. But the inceptionv3 model takes minimum of 96X96 images size. Hence, we have resized the image when we are loading the images using ImageDataGenerator.
- After training for 20 the model was unable to reach 50% accuracy. Which is not ideal.
- We got highest accuracy of 49.30% and Validation accuracy of 49.88%.

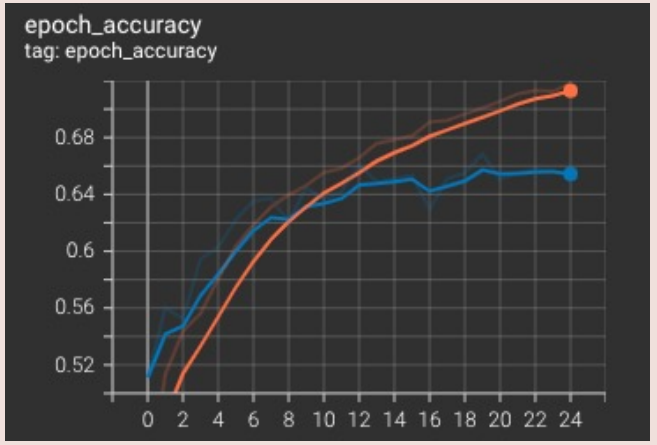
```
[ ]  from tensorflow.keras.applications.inception_v3 import InceptionV3

     inception = InceptionV3(input_shape=(96,96,3), weights='imagenet', include_top=False)
     # don't train existing weights
     for layer in inception.layers:
         layer.trainable = False

     flatten = Flatten(data_format='channels_last',name='Flatten')(inception.output)

     Out = Dense(units=len(classes),activation='softmax',kernel_initializer=tf.keras.initializers.glorot_normal(seed=3),name='Output')(flatten)

     model2 = Model(inputs=inception.input, outputs=Out)
     model2.summary()
```

```
[ ]
     batch_size  = 64


     datagen_train = ImageDataGenerator(rescale = 1./255,
                                        shear_range = 0.2,
                                        zoom_range = 0.2,
                                        horizontal_flip = True)

     datagen_val = ImageDataGenerator(rescale = 1./255)


     training_set = datagen_train.flow_from_directory(folder_path+"train",
                                                      target_size = (96,96),
                                                      batch_size=batch_size,
                                                      class_mode='categorical',
                                                      shuffle=True)


     test_set = datagen_val.flow_from_directory(folder_path+"test",
                                                target_size = (96,96),
                                                batch_size=batch_size,
                                                class_mode='categorical',
                                                shuffle=False)
```

```
Found 21077 images belonging to 4 classes.
Found 5140 images belonging to 4 classes.
```

Training Model 3





- We have used VGG16 again along with use of transfer learning.
- The transfer learning means, training some part of layers in the pretrained model(like VGG16).
- In the model we are performing the transfer learning on last 6 layers of VGG16 model.
- Unlike the inceptiov3 we are passing the images size as 48x48 using ImageDataGenerator.
- The training accuracy is around 70% and test accuracy is 66%. which may not be the best but it's decent enough.
- By using checkpoints, we have saved the model weights in each epoch and after training we have picked the best suitable weights.
- After loading the best weights, we have saved the model and downloaded into local system.
- With the downloaded file, we can load the model and use it for the emotion detection for real time data.

```
[ ] vgg = VGG16(include_top=False, weights='imagenet',input_shape=(48,48,3))
    for layer in vgg.layers[:-6]:
      layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
```

```
[ ] input_layer = Input(shape=(48,48,3,))
    vgg_layer = vgg(input_layer)
    layer1 = Conv2D(filters=1024,kernel_size=(7,7),strides=(1,1),padding="same",activation='relu',
                        kernel_initializer='he_normal')(vgg_layer)
    layer2 = Conv2D(filters=1024,kernel_size=(1,1),strides=(1,1),padding="same",activation='relu',
                        kernel_initializer='he_normal')(layer1)
    flatten_layer = Flatten()(layer2)
    layer3 = Dense(128,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=24))(flatten_layer)
    layer4 = Dense(64,activation='relu',kernel_initializer=tf.keras.initializers.he_normal(seed=20))(layer3)
    output = Dense(4,activation='softmax',kernel_initializer=tf.keras.initializers.GlorotNormal(seed=24))(layer4)
    model_3 = Model(inputs=input_layer,outputs=output)
    model_3.summary()
```

```
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
```

Results from Model 3

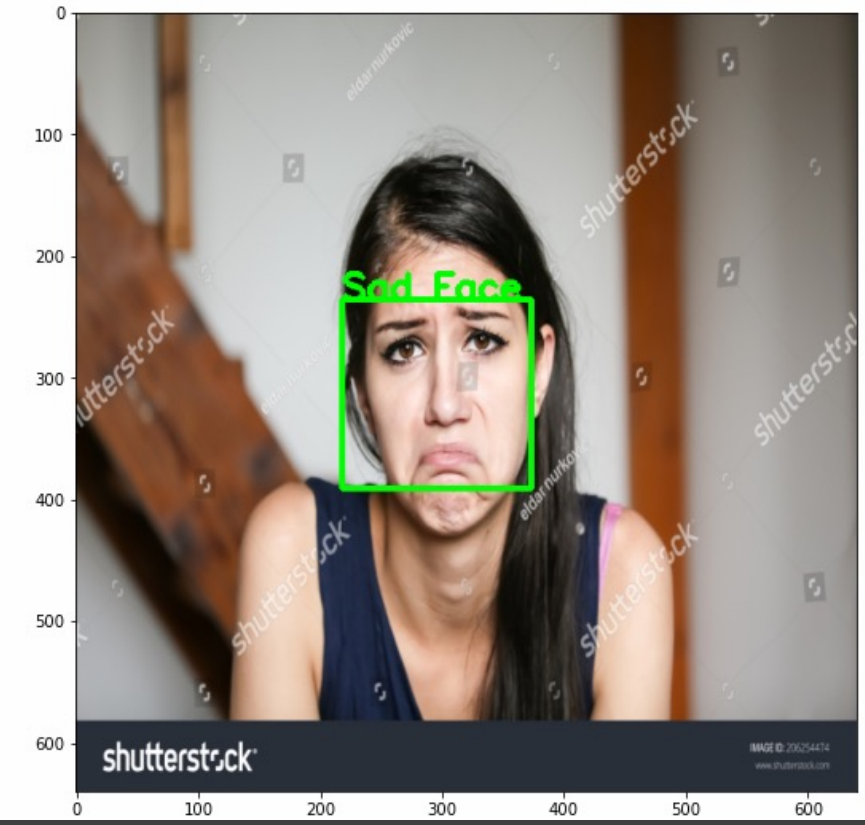
- In the whole training process, we have saved the weights for each epoch by using the concept of checkpoints.

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]:

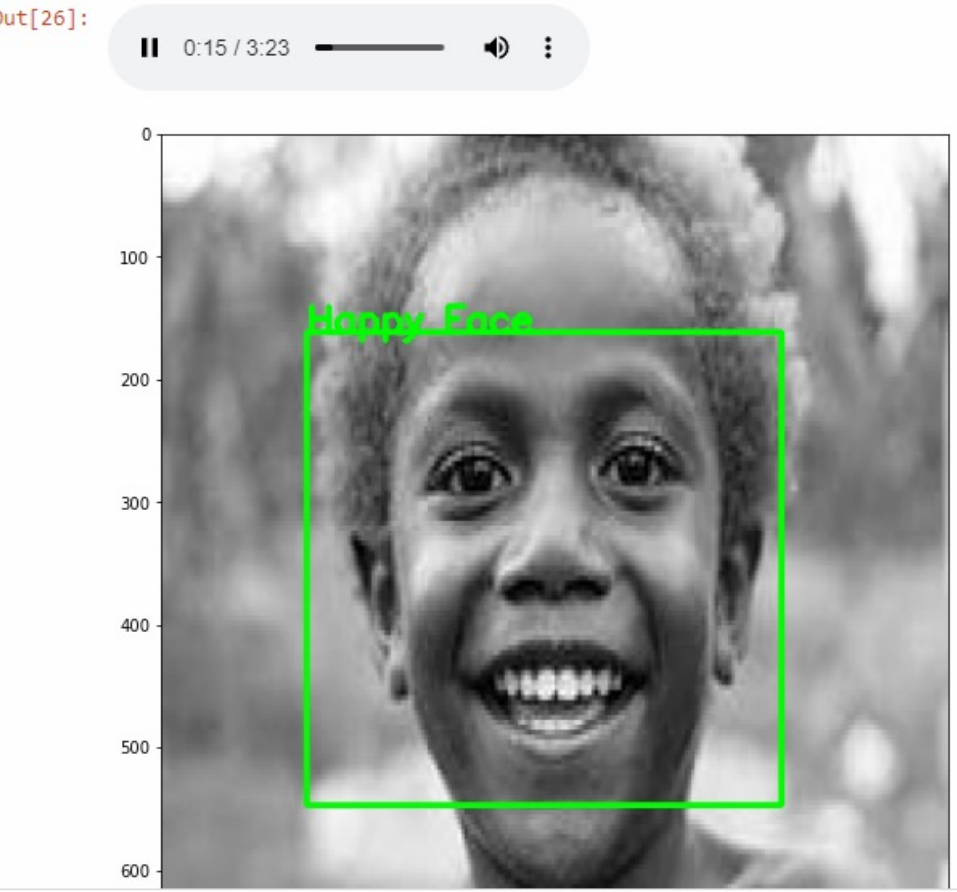


Happy:

```
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:

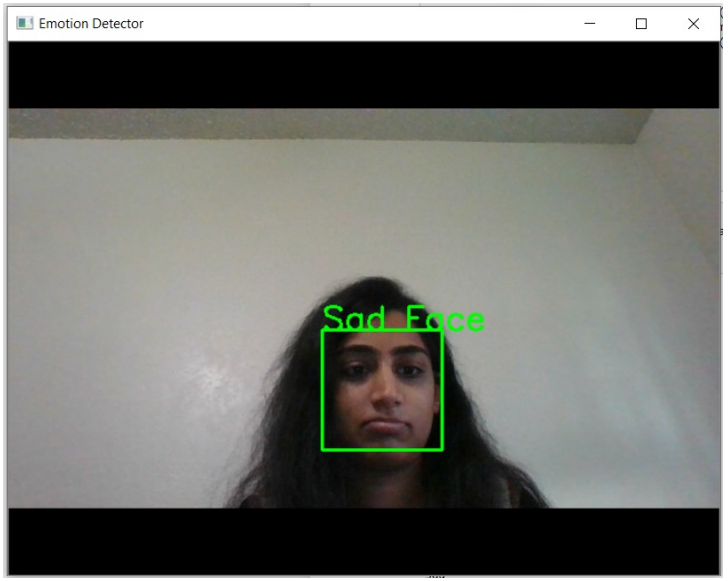
```
In [27]: path=r"C:\Users\Manasa\Downloads\angry.jpg"
         label = emotion_image(model,path,classes)
         print("detected emotion is {}".format(label))
         target_file = get_song(label)
         print(target_file)
         Audio(data=target_file,autoplay=True)

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

Out[27]: 0:06 / 3:27
```

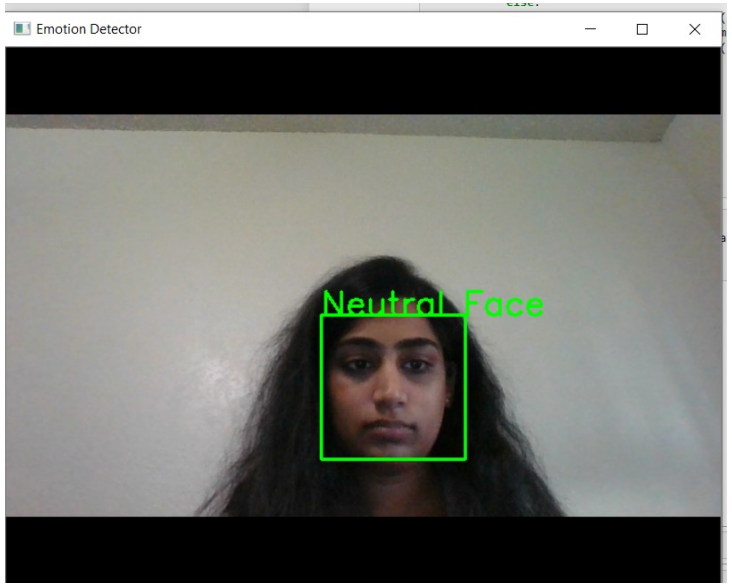
Neutral:

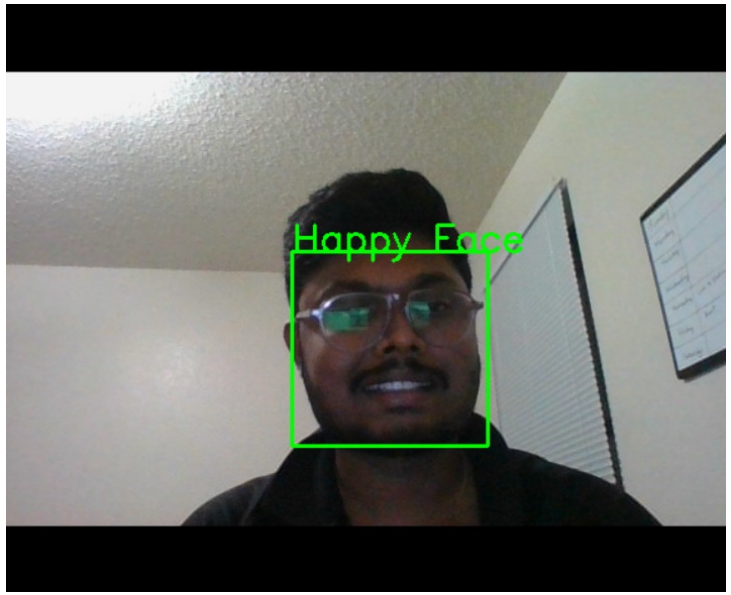
In [28]:
```
path=r"C:\Users\Manasa\Downloads\neutral.jpg"
label = emotion_image(model,path,classes)
print("detected emotion is {}".format(label))
target_file = get_song(label)
print(target_file)
Audio(data=target_file,autoplay=True)
```

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

Out[28]:







Emotion Detection from WEB-CAM

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

Out[34]:

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.

Links

Repository / Achieve:

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

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

Model demo:

https://youtu.be/InkZci6RzVA

Thank You