PROJECT UPDATE

FACIAL EMOTION DETECTION

Group 12

Gayatri Annapurna Vadlani

Mohammed Johny Shaik

GitLink: https://github.com/Gayatri345/CSCE5222.git

Problem statement:

We are planning to implement our project to detect facial emotions from video which uses Open CV haarcascades algorithm on video for face detection, facial key features and input it to the Deep Learning model to detect the emotion from the video. We are planning to work on emotions like angry, disgust, happy, sad, surprise and neutral.

TIMELINES:

Module Working on	**Study**	**Coding**	**Result**	**Due date**
Dataset	Done	Done	Done	Oct-10
Extracting facial features	Done	Done	Done	Oct-15
Developing a Neural Network Model	Done	In Progress	-	Oct-30
Experimenting with already available DeepFace for emotion recognition and calculating percentage of emotion.	Done	Done	Done	Oct-20
Video input to the model and calculate percentage of emotions. Removal of noise from the input and inputting video for facial recognition	In Progress	In Progress	In Progress	Nov 1-Nov 7
Training and Plotting the accuracies, predictions and confusion matrix.	Done	In Progress	In Progress	Nov5-Nov10

Error check and final code evaluation	In Progress	In Progress	In Progress	By Nov-15

Results/Progress till date:

1. **DataSet:**

   We worked on the dataset, we studied different classes of data available and converted the required data in to test.csv file and train.csv files which we are planning to use as input for our developed model for testing and training.

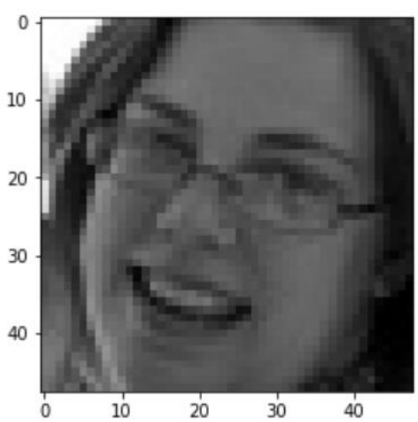
   We also plot a histogram to understand the distribution of different classes in the dataset FER-2013.

Sample training image imported from our dataset FER-2013:

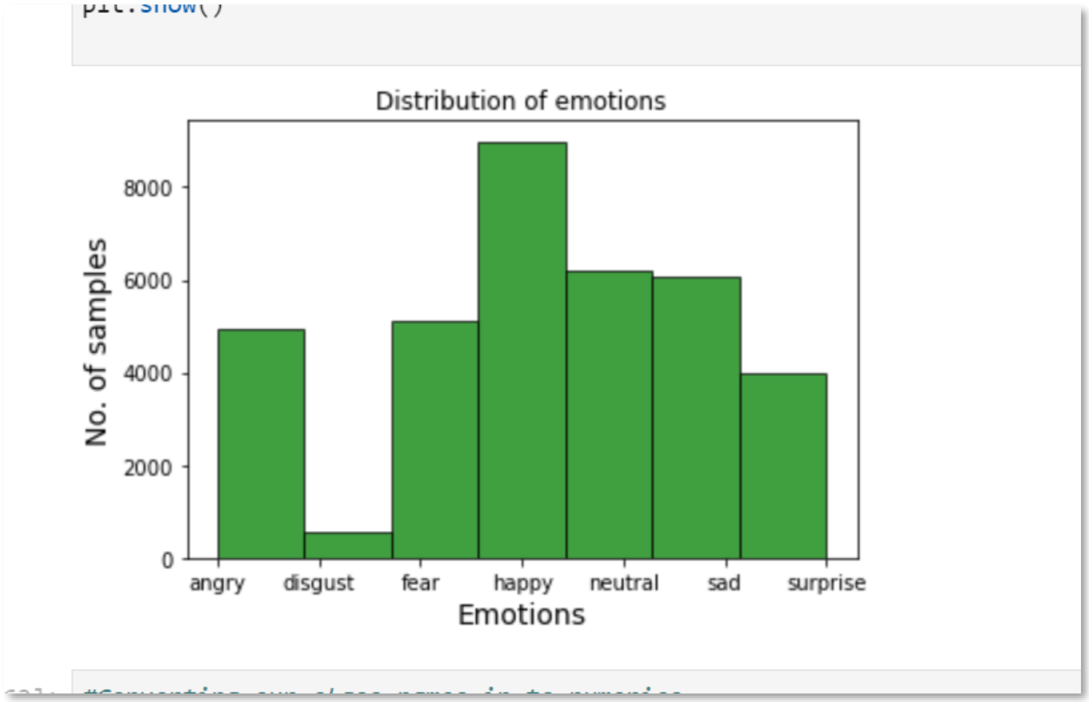
```
[156]: #Reading an image from train subfolder and a image from happy dir
       image = cv2.imread("dataset/train/happy/Training_10019449.jpg")

       #Plotting the image read
       plt.imshow(image)

[156]: <matplotlib.image.AxesImage at 0x1d5199a4940>
```



Class Distribution using Histogram Plot:



Exporting the data required to csv files:

```
[174]: df_train.to_csv('dataset/train.csv')

[175]: df_test.to_csv('dataset/test.csv')

[176]: df_encoded.to_csv('dataset/datasetFER_2013.csv')
```

2. **Facial feature extraction:**

   We used haarcascade algorithm to extract facial frontal-feature. We verified the same by using image of happy face as shown in the below picture.

   Reading an image:

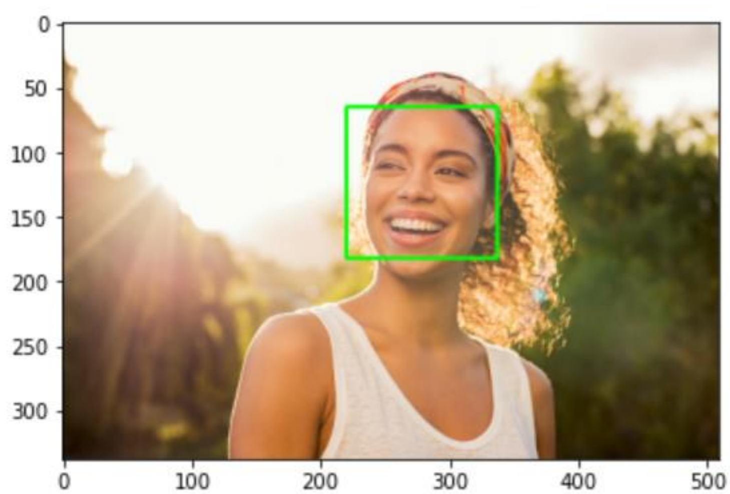
```
[6]: plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

```
[6]: <matplotlib.image.AxesImage at 0x188c3078c70>
```

```
[7]: plt.imshow(cv2.cvtColor(image, cv2.COLOR_RGB2GRAY),cmap='gray'
```

Detecting Face in the picture using haarcascade, and drawing a bounding box around the face.

```
]: <matplotlib.image.AxesImage at 0x188d03865b0>
```

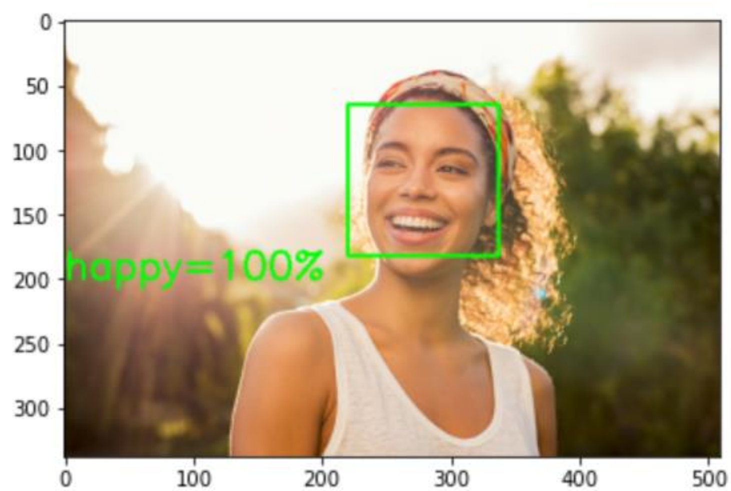


Getting amount of emotions:

We want to generate the percentage of emotion on the picture. For this we applied some python coding logic and tried to display the percentage of dominant emotion.

Here we are testing by using DeepFace in python library. Later this model will be replaced by our custom developed model deep neural network.

```
<matplotlib.image.AxesImage at 0x1890636ebe0>
```



FUTURE DEVELOPMENTS:

1. Developing model for facial emotion recognition.
2. Inputting video and using haarcascade to detect faces in the video.
3. Trying input a noise video and study the accuracy before and after removing noise in the video.
4. Compare the model with DeepFace model.
5. Generating confusion matrix, and predictions with valid and test set.

Project Update 2:

1. **Real Time Video Demo:**
   Designed the coding part and checked with DeepFace model.

REALTIME VIDEO

```
import cv2
from deepface import DeepFace

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(1)
#check if web cam is opened correctly
if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOError("Cannot open the webcam")
while True:
    ret,frame =cap.read();
    result =DeepFace.analyze(frame, actions = ['emotion'],enforce_detection=False)
    #cv2.putText(image,predictions['dominant_emotion'] +'='+ str(round(Emotions_perc['happy']))+'%',(0,200),font,1,(

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #print(face_cascade.empty())
    faces = face_cascade.detectMultiScale(gray,1.1,4)

    #Draw a rectangle
    for(x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w, y+h),(0,255,0),2)
    font = cv2.FONT_HERSHEY_SIMPLEX
    #putText() to insert text on the video

        #print(i)
```

2. Model

Sample Images generated from the prepared csv file.

Model:

CONVOLUTION LAYER OUTPUT: * #Ref:https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

input image = (48,48,1)=(h,w,d); filter/kernel_size = (3,3,1)=(fh,fw,fd); output = (h-fh+1),(w-fw+1),1; =>((48-3+1),(48-3+1),1); => (46,46,1); *Because number of filters are 32 output should be equal to (46,46,32)

MaxPool2D Output

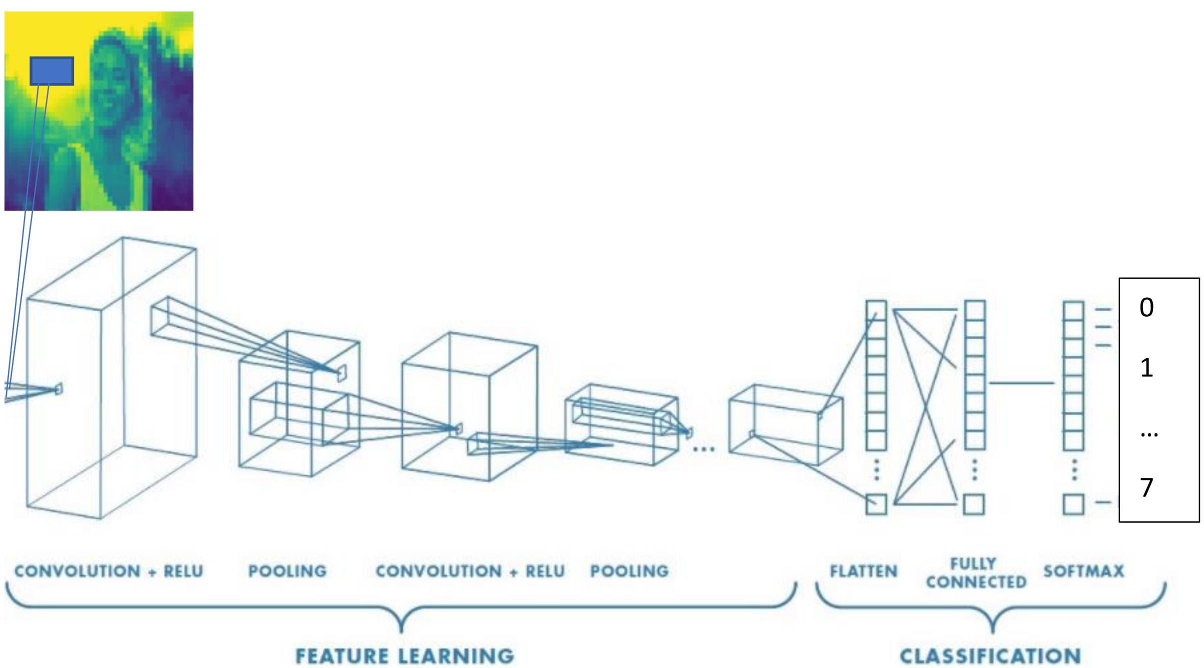
#Ref:https://dingyan89.medium.com/calculating-parameters-of-convolutional-and-fully-connected-layers-with-keras-186590df36c6

input = (46,46,1) filter= (2,2,1) if strides=(1,1) output=(45,45,32) if stride == (2,2)/None; then output = [(46-2)/2+1,(46-2)/2+1,32] Output = (23,23,32)

Batch Normalization Layer output

#Ref: https://keras.io/api/layers/normalization_layers/batch_normalization/ We are Normalizing each channel.

During training (i.e. when using fit() or when calling the layer/model with the argument training=True), the layer normalizes its output using the mean and standard deviation of the current batch of inputs. That is to say, for each channel being normalized, the layer returns gamma * (batch - mean(batch)) / sqrt(var(batch) + epsilon) + beta, where:



BaseModel2

```
BaseModel2 = tf.keras.models.Sequential([tf.keras.layers.Conv2D(32,(3, 3), activation='relu', input_shape=(48,48,1)),
                                         tf.keras.layers.MaxPool2D((2,2),strides=None),
                                         tf.keras.layers.BatchNormalization(),

                                         tf.keras.layers.Conv2D(64,(3, 3), activation='relu', input_shape=(48,48,1)),
                                         tf.keras.layers.MaxPool2D((2,2),strides=None),
                                         tf.keras.layers.BatchNormalization(),

                                         tf.keras.layers.Conv2D(128,(3, 3), activation='relu', input_shape=(48,48,1)),
                                         tf.keras.layers.MaxPool2D((2,2),strides=None),
                                         tf.keras.layers.BatchNormalization(),

                                         tf.keras.layers.Conv2D(256,(3, 3), activation='relu', input_shape=(48,48,1)),
                                         tf.keras.layers.MaxPool2D((2,2),strides=None),
                                         tf.keras.layers.BatchNormalization(),


                                         tf.keras.layers.Flatten(),

                                         tf.keras.layers.Dense(128,activation= 'relu'),
                                         tf.keras.layers.Dense(7,activation='softmax')])
```

```
new_model.summary()
```

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 46, 46, 32)        320
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
batch_normalization_25 (Batc (None, 23, 23, 32)        128
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_26 (Batc (None, 10, 10, 64)        256
_________________________________________________________________
conv2d_31 (Conv2D)           (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 4, 4, 128)         0
_________________________________________________________________
batch_normalization_27 (Batc (None, 4, 4, 128)         512
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 2, 2, 256)         295168
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 1, 1, 256)         0
_________________________________________________________________
batch_normalization_28 (Batc (None, 1, 1, 256)         1024
_________________________________________________________________
flatten_3 (Flatten)          (None, 256)               0
```

BaseModel2 train accuracy vs validation accuracy:

```
808/808 [==============================] - 103s 127ms/step - loss: 0.0830 - accuracy: 0.9780 - val_loss: 2.2530 - va
l_accuracy: 0.5152

Epoch 00018: saving model to checkpoint\BaseModel2.h5
Epoch 19/20
808/808 [==============================] - 103s 127ms/step - loss: 0.0776 - accuracy: 0.9796 - val_loss: 2.3557 - va
l_accuracy: 0.5136

Epoch 00019: saving model to checkpoint\BaseModel2.h5
Epoch 20/20
808/808 [==============================] - 104s 129ms/step - loss: 0.0714 - accuracy: 0.9815 - val_loss: 2.4631 - va
l_accuracy: 0.5023
```

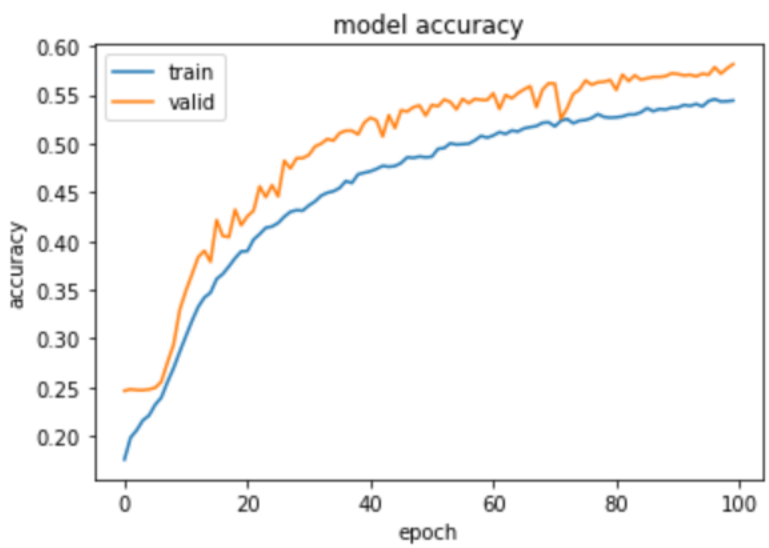
Model3 with Dropout layers to avoid overfitting:

```
Model3.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 46, 46, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 23, 23, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 23, 23, 32)        128
_________________________________________________________________
dropout (Dropout)            (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 4, 4, 128)         512
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 2, 2, 256)         295168
```

Train Accuracy vs Validation Accuracy

5]:
```python
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```

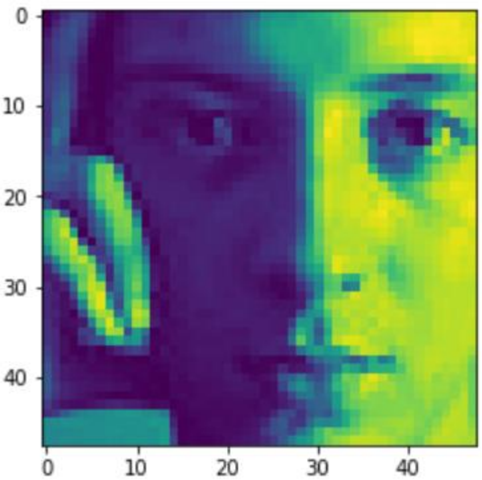


Accuracy achieved: 54%

Validation accuracy: 58%

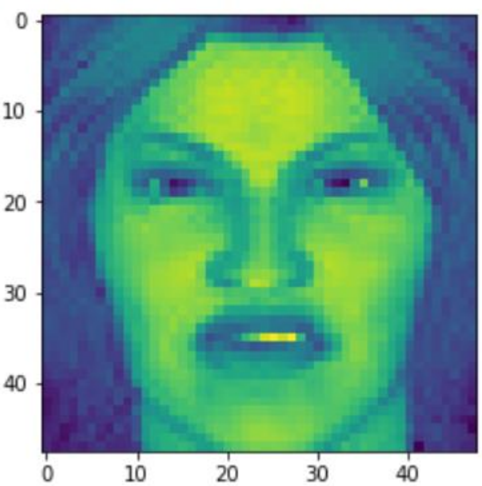
Predictions with Model2:

```
Original_label== neutral
Predicted_label== happy
Emotion_percentage 68.63 %
```

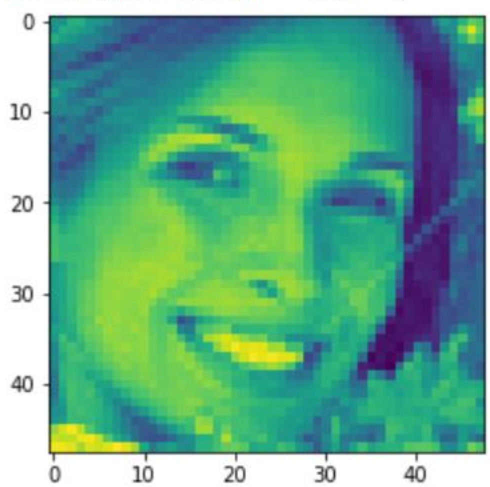


Predictions with Model3:

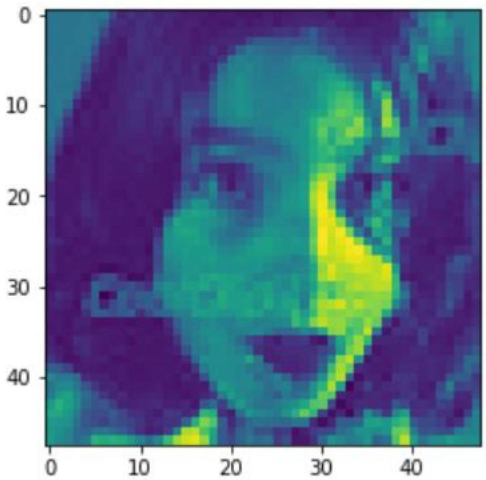
```
Original_label== disgust
Predicted_label == happy
Emotion_percentage = 100.0 %
```



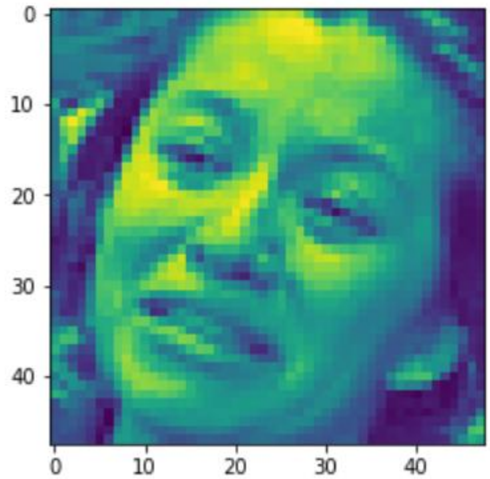
```
Original_label== happy
Predicted_label == happy
Emotion_percentage = 100.0 %
```



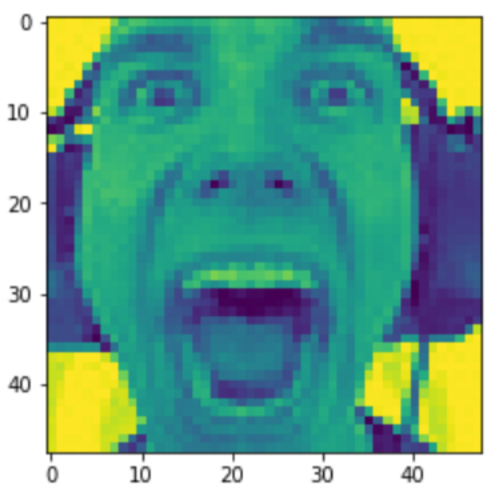
Original_label== surprise
Predicted_label == fear
Emotion_percentage = 100.0 %



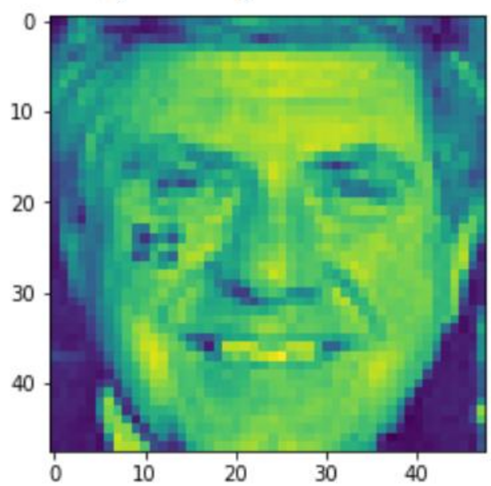
Original_label== sad
Predicted_label == fear
Emotion_percentage = 100.0 %



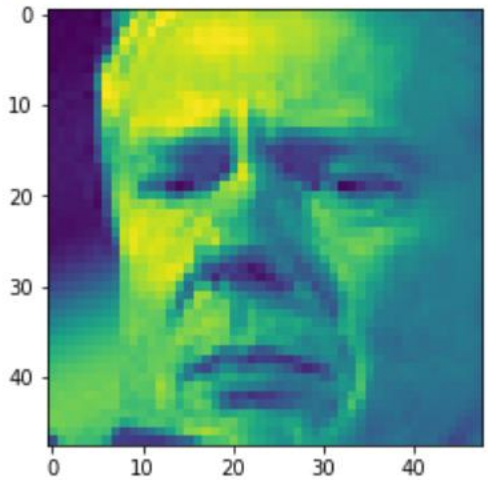
Original_label== angry
Predicted_label == angry
Emotion_percentage = 100.0 %



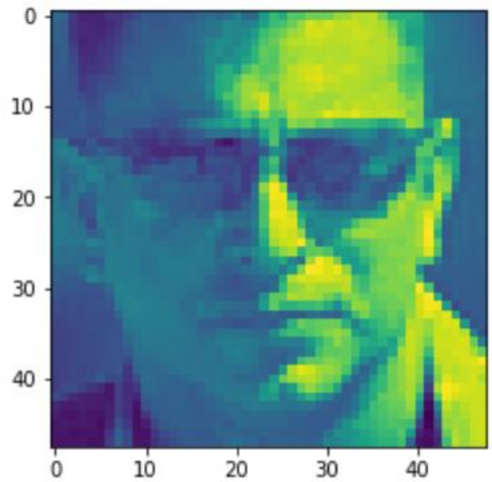
Original_label== happy
Predicted_label == happy
Emotion_percentage = 100.0 %



Original_label== fear
Predicted_label == fear
Emotion_percentage = 100.0 %



Original_label== neutral
Predicted_label == angry
Emotion_percentage = 97.93 %



Future Work:

1. **To improve accuracy**
2. **To merge both the modules**
3. **Check the predictions on video with designed model.**