STRESS DETECTOR AND

RECOMMENDATION

Introduction to Innovative Projects

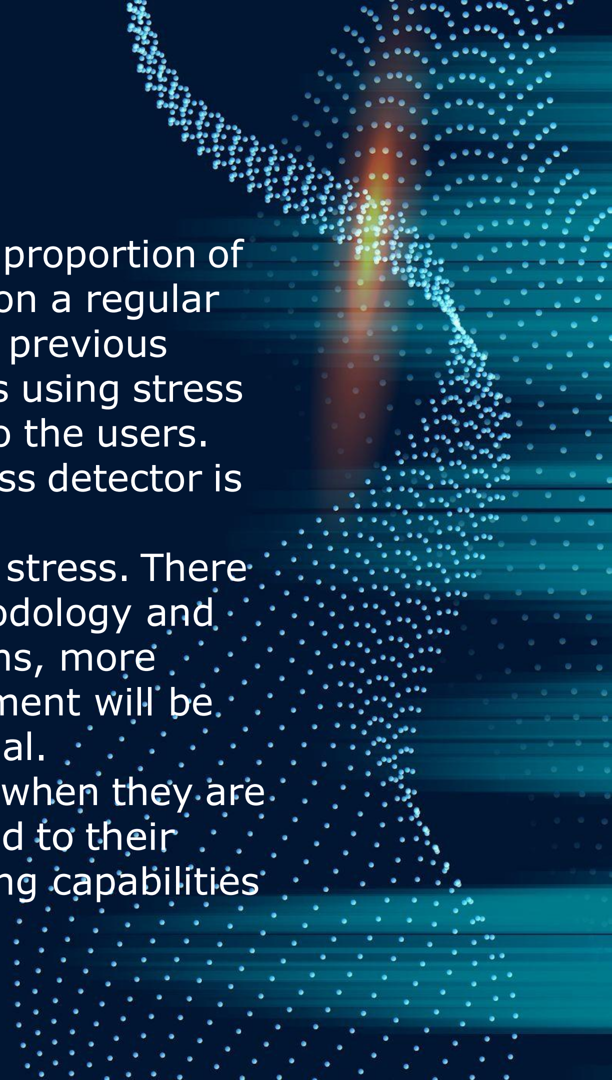
BY – VIRESH JAGADINNI (20BCI0292)

CONTENTS

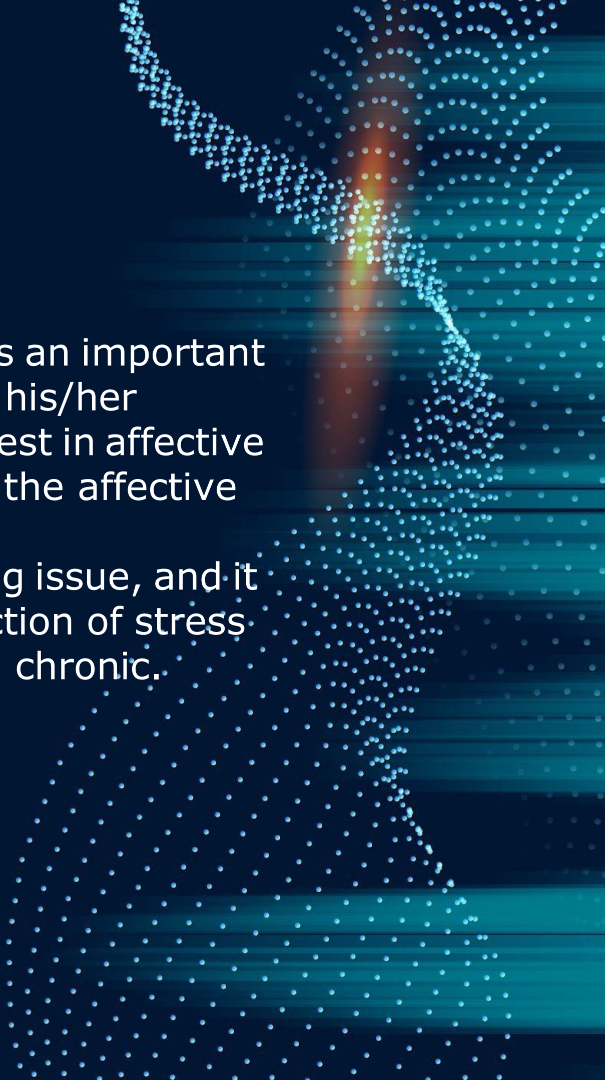
1. Introduction
2. Motivation
3. Innovation component in the project
4. Methodology
5. Future work

Introduction

- In today’s scenario due to increasing daily struggles, a large proportion of the population suffers from stress, anxiety and depression on a regular basis which affects them adversely. There have been many previous designs made in order to solve this issue of predicting stress using stress sensors which are costly and may not be easily accessible to the users. Therefore, a user friendly and easily accessible cheaper stress detector is used.
- Based on the results, a solution will be advised to reduce the stress. There are many prospects in this work. With more efficient methodology and using more factors like eating habits or eye blinking patterns, more accurate results can be analyzed, and more accurate treatment will be given to the patients. In the long run, it will be very beneficial.
- Stress can be defined as the reaction that people may have when they are subject to demands and pressures which do not correspond to their knowledge and abilities and that can challenge their handling capabilities



- In the last two decades, researchers have realized that there is an important relationship between the physical health of an individual and his/her emotional state/mental health. This has led to increasing interest in affective computing (AC) which makes use of technology to recognize the affective state of a person.
- Stress is a significant problem in modern society. It is a growing issue, and it has become an inescapable part of our daily lives. Early detection of stress will decrease the damage it causes and prevent it from being chronic.



MOTIVATION

- In the present scenario, where everyone is suffering from day-to-day challenges like increasing competition, unemployment, inflation, etc. It leads to drastic increase in number of people suffering from stress, anxiety and depression.
- The stress affects people’s productivity which hinders both their personal growth and the nation’s development. It also adversely affects their health. According to studies, increase in stress also increases the probability of heart diseases in patients.
- This alarming situation of human health has stimulated the design of web application that detects the person’s stress level and provides solutions to reduce it with the help of artificial intelligence.

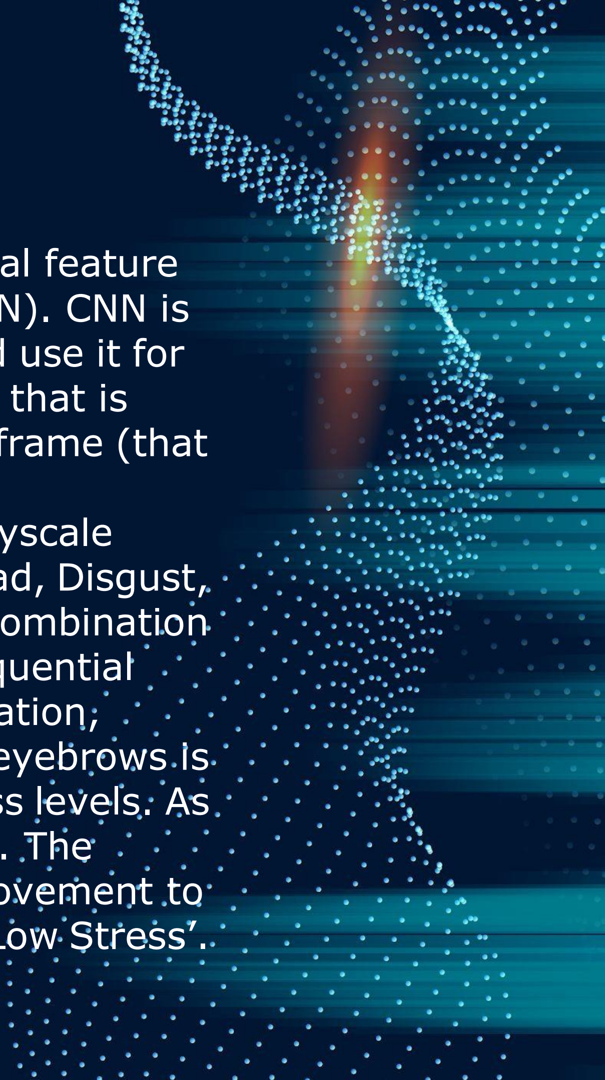
Innovation component in the project

The idea is to design a Stress detector API using OpenCV and Convolutional neural network.
In addition to the general method of detecting stress, a recommendation system has been proposed that will help the user to make an appropriate decision based on the emotion that has been detected by the API.

METHODOLOGY

Open CV is used to capture frames from video and extract facial feature like eyebrows of person using convolution neural network (CNN). CNN is use to reduce image to form that is easy to classify feature and use it for stress detection. Here spatial dependencies are pixel of image that is stored, and temporal dependencies is dynamic changes in the frame (that is image) with time captured in video.

For this model, we use dataset FER2013 containing 33,887 greyscale images which are classified into 7 emotions — Happy, Angry, Sad, Disgust, Surprise, Fear and Neutral using 5 convolutional layers with a combination of activation layers. These layers were implemented using Sequential model and contained 7 blocks each with their respective Activation, Normalisation and Flattening layers. The convex shape of the eyebrows is calculated using a normalisation formula to determine the stress levels. As the eyebrow movement changes the stress levels also change. The program calculates the cumulative value from the eyebrow movement to find total stress value and detect whether it is ‘High Stress’ or ‘Low Stress’.



- SECTION 1 - Euclidean distance and coordinates of the image is defined and emotions are classified.

- SECTION 2 - We normalize the image arrays in the range of 0-1, since neural networks are highly sensitive to non-normalized data.

- SECTION 3 - We extract the points of eye from the facial landmarks features and set to pre-process the image.

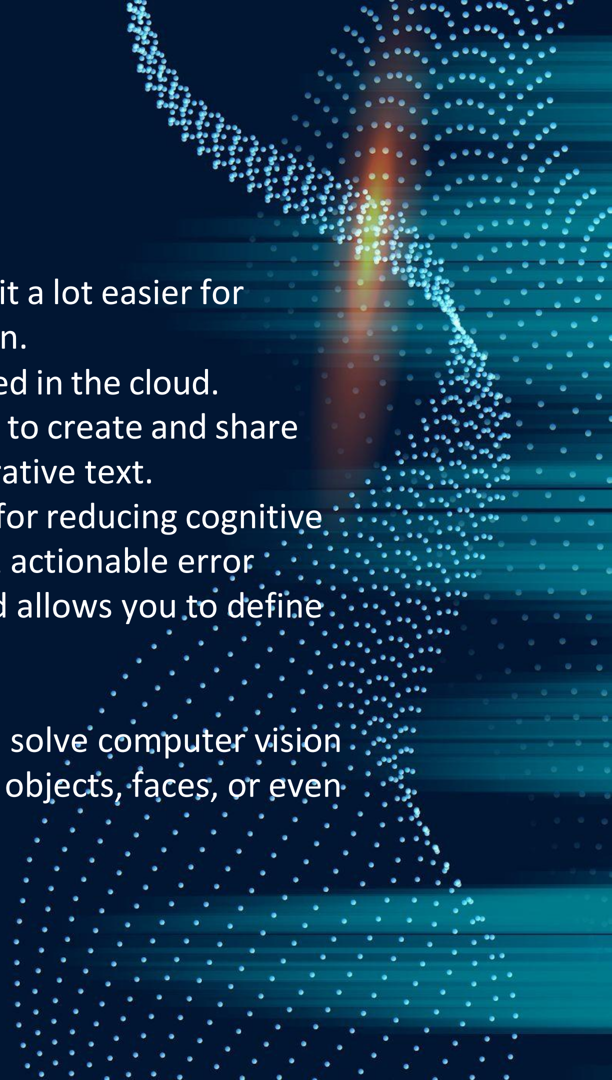
- SECTION 4 - The model then predicts the convex hull points of the eyebrow and analyze the shapes present in the images or video and get the contour of the left and right eye.

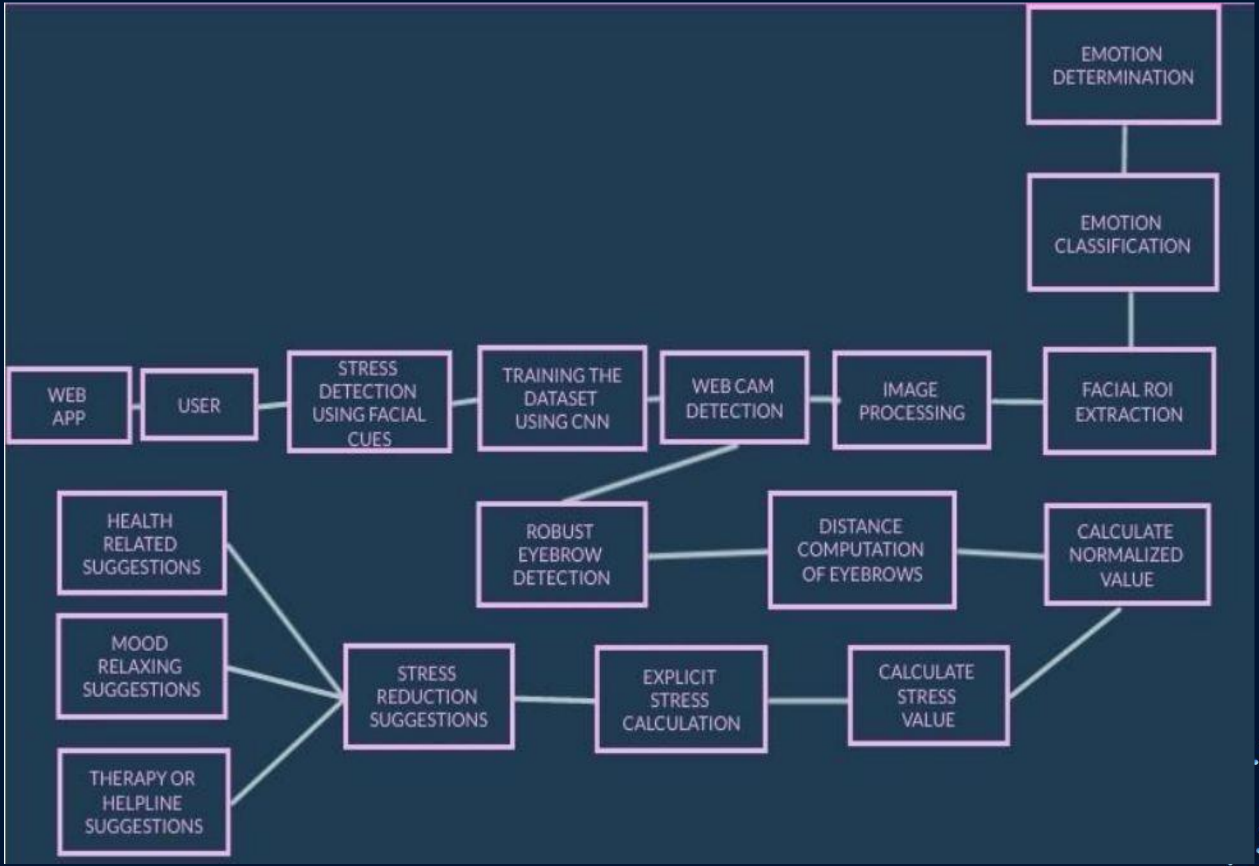
- SECTION 5 - The next segment is to train the data model using the different pooling layers of CNN using keras by importing sequential model.

- SECTION 6 - We simply calculate the distances using the distances and an Object Detection Algorithm to identify faces in an image or a real time video. The video captured is pre-processed in order to display stress value on the screen

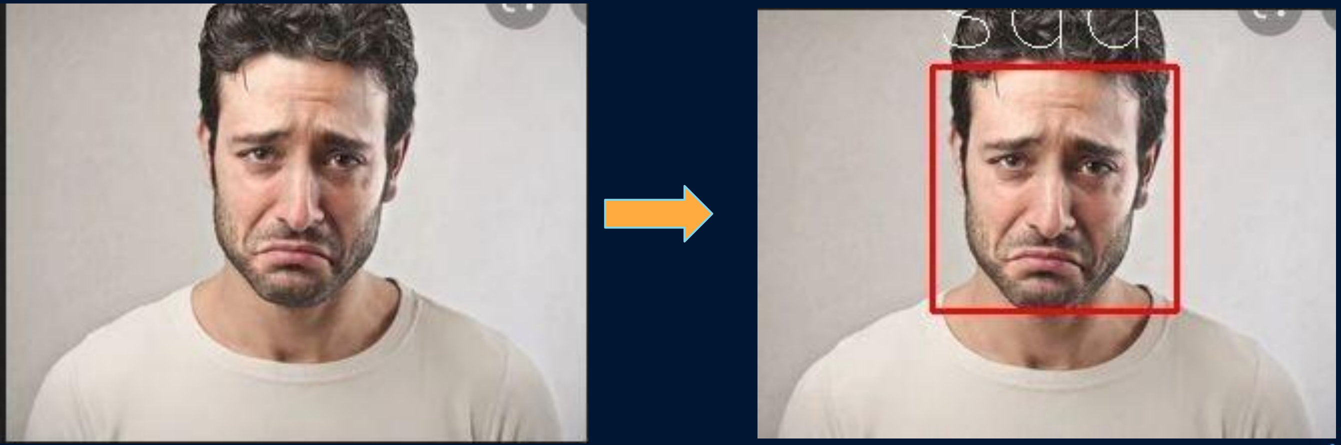
Tgchnicnl Stncfi usgd:

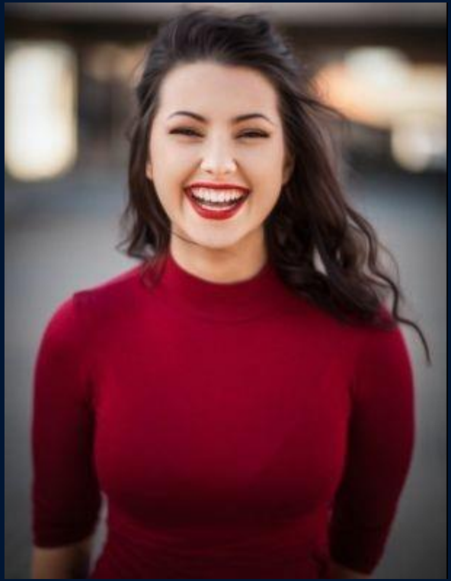
- **GITHUB** : offers a cloud-based Git repository hosting service making it a lot easier for individuals and teams to use Git for version control and collaboration.
- **CODESPACES** : codespace is a development environment that's hosted in the cloud.
- **JUPYTER** Notebook: an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- **Keras**: an API designed for human beings that follows best practices for reducing cognitive load and minimizes the number of user actions and provides clear & actionable error messages. It wraps the efficient numerical computation libraries and allows you to define and train neural network models in just a few lines of code.
- **Scipy**: Computes the Euclidean distance between two 1-D arrays.
- **OpenCV:** OpenCV-Python is a library of Python bindings designed to solve computer vision problems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

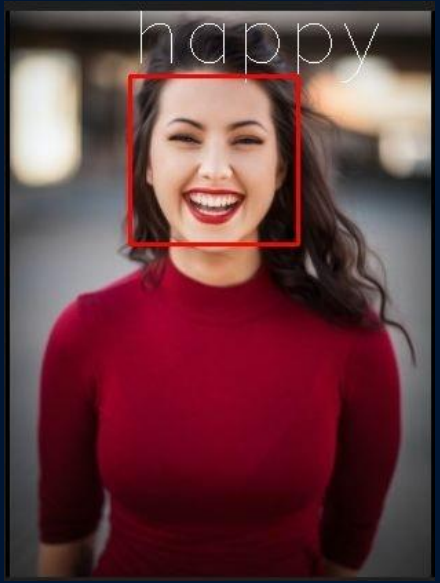



EMOTION DETERMINATION
EMOTION CLASSIFICATION
WEB APP
USER
STRESS DETECTION USING FACIAL CUES
TRAINING THE DATASET USING CNN
WEB CAM DETECTION
IMAGE PROCESSING
FACIAL ROI EXTRACTION
HEALTH RELATED SUGGESTIONS
ROBUST EYEBROW DETECTION
DISTANCE COMPUTATION OF EYEBROWS
CALCULATE NORMALIZED VALUE
MOOD RELAXING SUGGESTIONS
STRESS REDUCTION SUGGESTIONS
EXPLICIT STRESS CALCULATION
CALCULATE STRESS VALUE
THERAPY OR HELPLINE SUGGESTIONS

Output






happy

FUTURE WORKS

- In the future, we plan to develop the code even more to increase accuracy and speed. We also plan to develop our keystroke code and using flask structure develop an application for our model.

THANK YOU!!

Process

In [2]:

```
from scipy.spatial import distance as dist
from imutils import face_utils
import numpy as np
import  math
import imutils
import  time
import  dlib
import cv2
from cv2 import VideoWriter_fourcc, VideoWriter
import matplotlib.pyplot as plt
from tensorflow.keras.utils import img_to_array
# from keras.preprocessing.image import img_to_array
from keras.models import load_model
```

In [4]:

```
def eye_brow_distance(leye,reye):
    global points
    distq = dist.euclidean(leye,reye)
    #calculation of distance between left and right eye.
    points.append(int(distq))
    return distq

def emotion_finder(faces,frame):
    global emotion_classifier
    EMOTIONS = ["angry" ,"disgust","fear", "happy", "sad", "surprise","neutral"]
    x,y,w,h = face_utils.rect_to_bb(faces)
    frame = frame[y:y+h,x:x+w]
    roi = cv2.resize(frame,(64,64))
    roi = roi.astype("float") / 255.0
    roi = img_to_array(roi)
    roi = np.expand_dims(roi,axis=0)
    preds = emotion_classifier.predict(roi)[0]
    emotion_probability = np.max(preds)
    label = EMOTIONS[preds.argmax()]
    if label in ['fear','sad', 'neutral']:
        label = 'stressed'
    else:
        label = 'not stressed'
    return label

def normalize_values(points,disp):
    normalized_value = abs(disp - np.min(points))/abs(np.max(points) - np.min(points))
    stress_value = np.exp(-(normalized_value))
    return stress_value



detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
emotion_classifier = load_model("_mini_XCEPTION.102-0.66.hdf5", compile=False)
print(emotion_classifier, flush = True)
cap = cv2.VideoCapture('ved.mp4')
'''cap =cv2.VideoCapture(0)

fps=30 # Frames per second
size=(int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))

videoWriter=cv2.VideoWriter('MyVedio.avi',cv2.VideoWriter_fourcc('I','4','2','0'),fps,siz

success, frame =cap.read()
```

```
    #read gives two outputs

numFramesRemaining = 10*fps-1

while success and numFramesRemaining >0:
    videoWriter.write(frame)
    success,frame= cap.read()
    numFramesRemaining -=1'''

points = []
stress_list = []
stressval_list = []
stressgraph = []
size=0
while(True):
    _,frame = cap.read()
    if(not _): break
    frame = cv2.flip(frame,1)
    frame = imutils.resize(frame, width=500,height=500)


    (lBegin, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eyebrow"]
    (rBegin, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eyebrow"]

    #preprocessing the image
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

    detections = detector(gray,0)
    for detection in detections:
        emotion = emotion_finder(detection,gray)
        cv2.putText(frame, emotion, (10,10),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1
        shape = predictor(frame,detection)
        shape = face_utils.shape_to_np(shape)

        leyebrow = shape[lBegin:lEnd]
        reyebrow = shape[rBegin:rEnd]

        reyebrowhull = cv2.convexHull(reyebrow)
        leyebrowhull = cv2.convexHull(leyebrow)

        cv2.drawContours(frame, [reyebrowhull], -1, (0, 0, 255), 1)
        cv2.drawContours(frame, [leyebrowhull], -1, (0, 0, 255), 1)

        distq = eye_brow_distance(leyebrow[-1],reyebrow[0])
        stress_value = normalize_values(points,distq)
        print(stress_value)
        #if stress_value!=1.0: stress_list.append(stress_list)
        if math.isnan(stress_value):
            continue
        #cv2.putText(frame,"stress level:{}".format(str(int(stress_value*100))),(20,40),c
        #stress_list.append(frame)
        cv2.putText(frame,"stress level:{}".format(str(int(stress_value*100))),(20,40),cv
        stress_list.append(frame)
    height, width, layers = frame.shape
    size = (width,height)
    stressval_list.append(stress_value)
out = cv2.VideoWriter('resvid.avi',cv2.VideoWriter_fourcc(*'DIVX'), 10, size)
cap.release()
print("END REACHED")
for i in range(len(stress_list)):
    out.write(stress_list[i])
```

```
<keras.engine.functional.Functional object at 0x00000298FFEB1B20>
1/1 [==============================] - 0s 473ms/step
nan
```

```
C:\Users\Naina\AppData\Local\Temp/ipykernel_4572/3626964051.py:27: RuntimeWarning: invalid
value encountered in double_scalars
  normalized_value = abs(disp - np.min(points))/abs(np.max(points) - np.min(points))
1/1 [==============================] - 0s 51ms/step
0.9963304773755267
1/1 [==============================] - 0s 32ms/step
1.0
1/1 [==============================] - 0s 34ms/step
0.7165313105737893
1/1 [==============================] - 0s 50ms/step
0.7165313105737893
1/1 [==============================] - 0s 36ms/step
0.7165313105737893
1/1 [==============================] - 0s 42ms/step
0.36787944117144233
1/1 [==============================] - 0s 34ms/step
0.513417119032592
1/1 [==============================] - 0s 41ms/step
0.7165313105737893
1/1 [==============================] - 0s 39ms/step
1.0
1/1 [==============================] - 0s 42ms/step
0.5480051968723176
1/1 [==============================] - 0s 52ms/step
0.5488116360940265
1/1 [==============================] - 0s 59ms/step
0.6693203702378075
1/1 [==============================] - 0s 47ms/step
0.5488116360940265
1/1 [==============================] - 0s 74ms/step
0.6703200460356393
1/1 [==============================] - 0s 35ms/step
1.0
1/1 [==============================] - 0s 51ms/step
0.8174912623151172
1/1 [==============================] - 0s 47ms/step
0.9986988433691236
1/1 [==============================] - 0s 63ms/step
0.8453972514000658
1/1 [==============================] - 0s 41ms/step
0.7156272220906609
1/1 [==============================] - 0s 53ms/step
1.0
1/1 [==============================] - 0s 52ms/step
0.7165313105737893
1/1 [==============================] - 0s 44ms/step
0.9989925088965526
1/1 [==============================] - 0s 57ms/step
0.8816218988384807
1/1 [==============================] - 0s 35ms/step
0.9992819205181686
1/1 [==============================] - 0s 64ms/step
0.8464817248906141
1/1 [==============================] - 0s 56ms/step
0.6065306597126334
1/1 [==============================] - 0s 34ms/step
0.6592406302004438
1/1 [==============================] - 0s 66ms/step
0.9992819205181686
1/1 [==============================] - 0s 35ms/step
0.8464817248906141
1/1 [==============================] - 0s 49ms/step
0.9992819205181686
1/1 [==============================] - 0s 34ms/step
1.0
1/1 [==============================] - 0s 33ms/step
```

```
0.9975794926125358
1/1 [==============================] - 0s 31ms/step
0.9993941730606076
1/1 [==============================] - 0s 30ms/step
0.9994215132903
1/1 [==============================] - 0s 29ms/step
0.9994841867917443
1/1 [==============================] - 0s 40ms/step
0.9482495195900245
1/1 [==============================] - 0s 54ms/step
0.9468116258023203
1/1 [==============================] - 0s 42ms/step
0.9482495195900245
1/1 [==============================] - 0s 55ms/step
0.9487294800164372
1/1 [==============================] - 0s 44ms/step
0.9468116258023203
1/1 [==============================] - 0s 46ms/step
0.853523652728261
1/1 [==============================] - 0s 50ms/step
0.8539396656235352
1/1 [==============================] - 0s 49ms/step
0.9468116258023203
1/1 [==============================] - 0s 44ms/step
0.8983023733820897
1/1 [==============================] - 0s 44ms/step
0.9000876262522592
1/1 [==============================] - 0s 42ms/step
0.8996408615292885
1/1 [==============================] - 0s 42ms/step
0.9000876262522592
1/1 [==============================] - 0s 45ms/step
0.8539396656235352
1/1 [==============================] - 0s 53ms/step
0.9487294800164372
1/1 [==============================] - 0s 44ms/step
0.853523652728261
1/1 [==============================] - 0s 53ms/step
0.7686205265937358
1/1 [==============================] - 0s 70ms/step
0.7682594463669772
1/1 [==============================] - 0s 47ms/step
0.8539396656235352
1/1 [==============================] - 0s 37ms/step
0.8097702243888371
1/1 [==============================] - 0s 56ms/step
0.6915120242186787
1/1 [==============================] - 0s 32ms/step
0.6915120242186787
1/1 [==============================] - 0s 47ms/step
0.6918258252705172
1/1 [==============================] - 0s 37ms/step
0.7292129525252351
1/1 [==============================] - 0s 38ms/step
0.6563555554708402
1/1 [==============================] - 0s 43ms/step
0.5907775139012317
1/1 [==============================] - 0s 45ms/step
0.6560628872775571
1/1 [==============================] - 0s 46ms/step
0.6227038648477501
1/1 [==============================] - 0s 51ms/step
0.560488043568919
1/1 [==============================] - 0s 43ms/step
0.6560628872775571
1/1 [==============================] - 0s 27ms/step
```

```
0.5907775139012317
1/1 [==============================] - 0s 72ms/step
0.6227038648477501
1/1 [==============================] - 0s 34ms/step
0.5905227207908098
1/1 [==============================] - 0s 65ms/step
0.5044883526787212
1/1 [==============================] - 0s 51ms/step
0.560488043568919
1/1 [==============================] - 0s 52ms/step
0.5044883526787212
1/1 [==============================] - 0s 61ms/step
0.5317515301305707
1/1 [==============================] - 0s 32ms/step
0.5315294720643506
1/1 [==============================] - 0s 57ms/step
0.5042809703296425
1/1 [==============================] - 0s 50ms/step
0.4786229725112321
1/1 [==============================] - 0s 66ms/step
0.5315294720643506
1/1 [==============================] - 0s 64ms/step
0.560488043568919
1/1 [==============================] - 0s 48ms/step
0.47842924870869347
1/1 [==============================] - 0s 66ms/step
0.4539027161944048
1/1 [==============================] - 0s 50ms/step
0.560488043568919
1/1 [==============================] - 0s 59ms/step
0.45408372383450274
1/1 [==============================] - 0s 64ms/step
0.4786229725112321
1/1 [==============================] - 0s 26ms/step
0.5602502113620464
1/1 [==============================] - 0s 34ms/step
0.5897592055889845
1/1 [==============================] - 0s 43ms/step
0.5315294720643506
1/1 [==============================] - 0s 52ms/step
0.560488043568919
1/1 [==============================] - 0s 36ms/step
0.5602502113620464
1/1 [==============================] - 0s 47ms/step
0.6563555554708402
1/1 [==============================] - 0s 36ms/step
0.5905227207908098
1/1 [==============================] - 0s 33ms/step
0.5044883526787212
1/1 [==============================] - 0s 40ms/step
0.4539027161944048
1/1 [==============================] - 0s 51ms/step
0.5317515301305707
1/1 [==============================] - 0s 52ms/step
0.4786229725112321
1/1 [==============================] - 0s 36ms/step
0.43063345021612087
1/1 [==============================] - 0s 47ms/step
0.4786229725112321
1/1 [==============================] - 0s 42ms/step
0.5317515301305707
1/1 [==============================] - 0s 37ms/step
0.45408372383450274
1/1 [==============================] - 0s 46ms/step
0.4539027161944048
1/1 [==============================] - 0s 44ms/step
```

```
0.45408372383450274
1/1 [==============================] - 0s 44ms/step
0.43063345021612087
1/1 [==============================] - 0s 51ms/step
0.5042809703296425
1/1 [==============================] - 0s 57ms/step
0.4539027161944048
1/1 [==============================] - 0s 31ms/step
0.45408372383450274
1/1 [==============================] - 0s 48ms/step
0.5044883526787212
1/1 [==============================] - 0s 42ms/step
0.43080261519743523
1/1 [==============================] - 0s 44ms/step
0.43063345021612087
1/1 [==============================] - 0s 41ms/step
0.408715141105984
1/1 [==============================] - 0s 41ms/step
0.5042809703296425
1/1 [==============================] - 0s 47ms/step
0.45408372383450274
1/1 [==============================] - 0s 46ms/step
0.5315294720643506
1/1 [==============================] - 0s 58ms/step
0.4539027161944048
1/1 [==============================] - 0s 38ms/step
0.45408372383450274
1/1 [==============================] - 0s 52ms/step
0.45408372383450274
1/1 [==============================] - 0s 52ms/step
0.43080261519743523
1/1 [==============================] - 0s 41ms/step
0.36787944117144233
1/1 [==============================] - 0s 70ms/step
0.43080261519743523
1/1 [==============================] - 0s 48ms/step
0.45408372383450274
1/1 [==============================] - 0s 44ms/step
0.43080261519743523
1/1 [==============================] - 0s 42ms/step
0.4539027161944048
1/1 [==============================] - 0s 48ms/step
0.408715141105984
1/1 [==============================] - 0s 52ms/step
0.3876122521072523
1/1 [==============================] - 0s 39ms/step
0.4539027161944048
1/1 [==============================] - 0s 60ms/step
0.36787944117144233
1/1 [==============================] - 0s 68ms/step
0.43080261519743523
1/1 [==============================] - 0s 63ms/step
0.45408372383450274
1/1 [==============================] - 0s 48ms/step
0.43063345021612087
1/1 [==============================] - 0s 48ms/step
0.43080261519743523
1/1 [==============================] - 0s 66ms/step
0.4085570087611661
1/1 [==============================] - 0s 39ms/step
0.43080261519743523
1/1 [==============================] - 0s 45ms/step
0.43080261519743523
1/1 [==============================] - 0s 56ms/step
0.408715141105984
1/1 [==============================] - 0s 50ms/step
```

```
0.3876122521072523
1/1 [==============================] - 0s 53ms/step
0.5044883526787212
END REACHED
```

Training the model ;)

In [3]:

```
from __future___ import print_function
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation,Flatten,BatchNormalization
from keras.layers import Conv2D,MaxPooling2D
import os

num_classes = 7 # number of labels
img_rows,img_cols = 48,48
batch_size = 32 #number of traning example utlized in 1 iteration

train_data_dir = 'train'
validation_data_dir = 'test'

train_datagen = ImageDataGenerator(
                                        rescale=1./255,
                                        rotation_range=30,
                                        shear_range=0.3,
                                        zoom_range=0.3,
                                        width_shift_range=0.4,
                                        height_shift_range=0.4,
                                        horizontal_flip=True,
                                        fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
                                        train_data_dir,
                                        color_mode='grayscale',
                                        target_size=(img_rows,img_cols),
                                        batch_size=batch_size,
                                        class_mode='categorical',
                                        shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                                                        validation_data_dir,
                                                        color_mode='grayscale',
                                                        target_size=(img_rows,img_cols),
                                                        batch_size=batch_size,
                                                        class_mode='categorical',
                                                        shuffle=True)


model = Sequential()

# LAYER 1

model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(img_
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(32,(3,3),padding='same',kernel_initializer='he_normal',input_shape=(img_
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))
```

```python
# LAYER 2

model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(64,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))

# LAYER 3

model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Conv2D(128,(3,3),padding='same',kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2))


# LAYER 4

model.add(Flatten())
model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# LAYER 5

model.add(Dense(64,kernel_initializer='he_normal'))
model.add(Activation('elu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# LAYER 6

model.add(Dense(num_classes,kernel_initializer='he_normal'))
model.add(Activation('softmax'))

print(model.summary())
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320       
                                                                 
 activation (Activation)     (None, 48, 48, 32)        0         
                                                                 
 batch_normalization (BatchN  (None, 48, 48, 32)       128       
 ormalization)                                                   
                                                                 
 conv2d_1 (Conv2D)           (None, 48, 48, 32)        9248      
                                                                 
 activation_1 (Activation)   (None, 48, 48, 32)        0         
                                                                 
 batch_normalization_1 (Batc  (None, 48, 48, 32)       128       
 hNormalization)                                                 
```

```
max_pooling2d (MaxPooling2D   (None, 24, 24, 32)       0
)

dropout (Dropout)            (None, 24, 24, 32)       0

conv2d_2 (Conv2D)            (None, 24, 24, 64)       18496

activation_2 (Activation)    (None, 24, 24, 64)       0

batch_normalization_2 (Batc   (None, 24, 24, 64)      256
hNormalization)

conv2d_3 (Conv2D)            (None, 24, 24, 64)       36928

activation_3 (Activation)    (None, 24, 24, 64)       0

batch_normalization_3 (Batc   (None, 24, 24, 64)      256
hNormalization)

max_pooling2d_1 (MaxPooling   (None, 12, 12, 64)      0
2D)

dropout_1 (Dropout)          (None, 12, 12, 64)       0

conv2d_4 (Conv2D)            (None, 12, 12, 128)      73856

activation_4 (Activation)    (None, 12, 12, 128)      0

batch_normalization_4 (Batc   (None, 12, 12, 128)     512
hNormalization)

conv2d_5 (Conv2D)            (None, 12, 12, 128)      147584

activation_5 (Activation)    (None, 12, 12, 128)      0

batch_normalization_5 (Batc   (None, 12, 12, 128)     512
hNormalization)

max_pooling2d_2 (MaxPooling   (None, 6, 6, 128)       0
2D)

dropout_2 (Dropout)          (None, 6, 6, 128)        0

flatten (Flatten)            (None, 4608)             0

dense (Dense)                (None, 64)               294976

activation_6 (Activation)    (None, 64)               0

batch_normalization_6 (Batc   (None, 64)              256
hNormalization)

dropout_3 (Dropout)          (None, 64)               0

dense_1 (Dense)              (None, 64)               4160

activation_7 (Activation)    (None, 64)               0

batch_normalization_7  (Batc  (None, 64)              256
hNormalization)

dropout_4 (Dropout)          (None, 64)               0

dense_2 (Dense)              (None, 7)                455
```

```
 activation_8 (Activation)    (None, 7)                 0

=================================================================
Total params: 588,327
Trainable params: 587,175
Non-trainable params: 1,152
_________________________________________________________________
None
```

In [5]:

```
from tensorflow.keras.optimizers import RMSprop,SGD,Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint('Users/Naina/Stress/Emotion_little_vgg.h5',
                             monitor='val_loss',
                             mode='min',
                             save_best_only=True,
                             verbose=1)

earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              verbose=1,
                              min_delta=0.0001)

callbacks = [earlystop,checkpoint,reduce_lr]

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])

nb_train_samples = 24176
nb_validation_samples = 3006
epochs=25

history=model.fit_generator(
                train_generator,
                steps_per_epoch=nb_train_samples//batch_size,
                epochs=epochs,
                callbacks=callbacks,
                validation_data=validation_generator,
                validation_steps=nb_validation_samples//batch_size)

model.save('trained_model.hdf5')
```

```
C:\Users\Naina\AppData\Local\Temp/ipykernel_8988/3405561283.py:33: UserWarning: `Model.fit
_generator` is deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  history=model.fit_generator(
Epoch 1/25
755/755 [==============================] - ETA: 0s - loss: 1.7966 - accuracy: 0.2494
Epoch 00001: val_loss improved from inf to 1.76592, saving model to Users/Naina/Stress\Emo
tion_little_vgg.h5
755/755 [==============================] - 236s 311ms/step - loss: 1.7966 - accuracy: 0.24
94 - val_loss: 1.7659 - val_accuracy: 0.2769 - lr: 0.0010
Epoch 2/25
755/755 [==============================] - ETA: 0s - loss: 1.7724 - accuracy: 0.2649
Epoch 00002: val_loss improved from 1.76592 to 1.75244, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
```

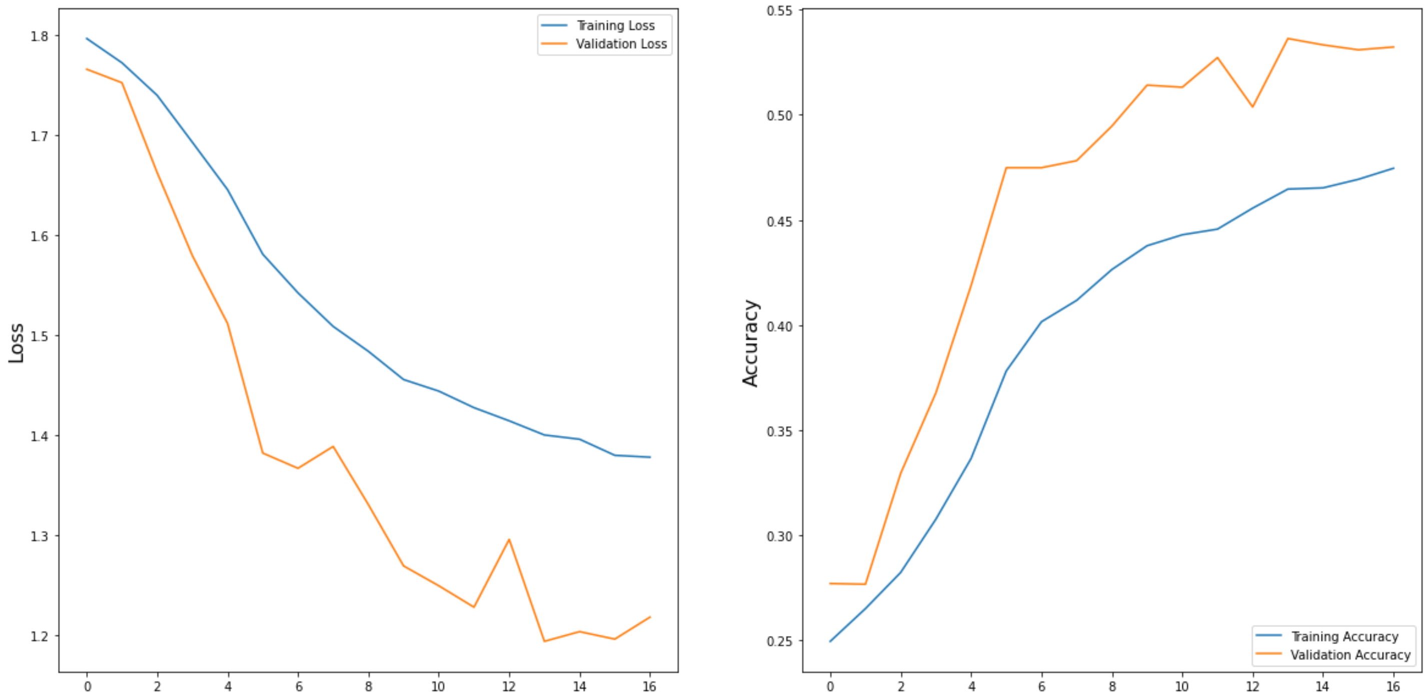
```
755/755 [==============================] - 232s 308ms/step - loss: 1.7724 - accuracy: 0.26
49 - val_loss: 1.7524 - val_accuracy: 0.2765 - lr: 0.0010
Epoch 3/25
755/755 [==============================] - ETA: 0s - loss: 1.7399 - accuracy: 0.2821
Epoch 00003: val_loss improved from 1.75244 to 1.66264, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 233s 308ms/step - loss: 1.7399 - accuracy: 0.28
21 - val_loss: 1.6626 - val_accuracy: 0.3293 - lr: 0.0010
Epoch 4/25
755/755 [==============================] - ETA: 0s - loss: 1.6930 - accuracy: 0.3075
Epoch 00004: val_loss improved from 1.66264 to 1.57970, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 233s 309ms/step - loss: 1.6930 - accuracy: 0.30
75 - val_loss: 1.5797 - val_accuracy: 0.3676 - lr: 0.0010
Epoch 5/25
755/755 [==============================] - ETA: 0s - loss: 1.6456 - accuracy: 0.3364
Epoch 00005: val_loss improved from 1.57970 to 1.51193, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 236s 312ms/step - loss: 1.6456 - accuracy: 0.33
64 - val_loss: 1.5119 - val_accuracy: 0.4187 - lr: 0.0010
Epoch 6/25
755/755 [==============================] - ETA: 0s - loss: 1.5811 - accuracy: 0.3781
Epoch 00006: val_loss improved from 1.51193 to 1.38241, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 235s 312ms/step - loss: 1.5811 - accuracy: 0.37
81 - val_loss: 1.3824 - val_accuracy: 0.4748 - lr: 0.0010
Epoch 7/25
755/755 [==============================] - ETA: 0s - loss: 1.5425 - accuracy: 0.4015
Epoch 00007: val_loss improved from 1.38241 to 1.36719, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 236s 312ms/step - loss: 1.5425 - accuracy: 0.40
15 - val_loss: 1.3672 - val_accuracy: 0.4748 - lr: 0.0010
Epoch 8/25
755/755 [==============================] - ETA: 0s - loss: 1.5089 - accuracy: 0.4117
Epoch 00008: val_loss did not improve from 1.36719
755/755 [==============================] - 236s 312ms/step - loss: 1.5089 - accuracy: 0.41
17 - val_loss: 1.3890 - val_accuracy: 0.4782 - lr: 0.0010
Epoch 9/25
755/755 [==============================] - ETA: 0s - loss: 1.4841 - accuracy: 0.4263
Epoch 00009: val_loss improved from 1.36719 to 1.33090, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 238s 315ms/step - loss: 1.4841 - accuracy: 0.42
63 - val_loss: 1.3309 - val_accuracy: 0.4946 - lr: 0.0010
Epoch 10/25
755/755 [==============================] - ETA: 0s - loss: 1.4559 - accuracy: 0.4377
Epoch 00010: val_loss improved from 1.33090 to 1.26966, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 243s 321ms/step - loss: 1.4559 - accuracy: 0.43
77 - val_loss: 1.2697 - val_accuracy: 0.5141 - lr: 0.0010
Epoch 11/25
755/755 [==============================] - ETA: 0s - loss: 1.4443 - accuracy: 0.4429
Epoch 00011: val_loss improved from 1.26966 to 1.24982, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 254s 336ms/step - loss: 1.4443 - accuracy: 0.44
29 - val_loss: 1.2498 - val_accuracy: 0.5131 - lr: 0.0010
Epoch 12/25
755/755 [==============================] - ETA: 0s - loss: 1.4278 - accuracy: 0.4456
Epoch 00012: val_loss improved from 1.24982 to 1.22843, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 248s 328ms/step - loss: 1.4278 - accuracy: 0.44
56 - val_loss: 1.2284 - val_accuracy: 0.5272 - lr: 0.0010
Epoch 13/25
755/755 [==============================] - ETA: 0s - loss: 1.4146 - accuracy: 0.4556
Epoch 00013: val_loss did not improve from 1.22843
755/755 [==============================] - 237s 315ms/step - loss: 1.4146 - accuracy: 0.45
56 - val_loss: 1.2962 - val_accuracy: 0.5037 - lr: 0.0010
```

```
Epoch 14/25
755/755 [==============================] - ETA: 0s - loss: 1.4005 - accuracy: 0.4646
Epoch 00014: val_loss improved from 1.22843 to 1.19430, saving model to Users/Naina/Stress
\Emotion_little_vgg.h5
755/755 [==============================] - 236s 312ms/step - loss: 1.4005 - accuracy: 0.46
46 - val_loss: 1.1943 - val_accuracy: 0.5363 - lr: 0.0010
Epoch 15/25
755/755 [==============================] - ETA: 0s - loss: 1.3962 - accuracy: 0.4652
Epoch 00015: val_loss did not improve from 1.19430
755/755 [==============================] - 237s 314ms/step - loss: 1.3962 - accuracy: 0.46
52 - val_loss: 1.2040 - val_accuracy: 0.5333 - lr: 0.0010
Epoch 16/25
755/755 [==============================] - ETA: 0s - loss: 1.3802 - accuracy: 0.4693
Epoch 00016: val_loss did not improve from 1.19430
755/755 [==============================] - 247s 328ms/step - loss: 1.3802 - accuracy: 0.46
93 - val_loss: 1.1965 - val_accuracy: 0.5309 - lr: 0.0010
Epoch 17/25
755/755 [==============================] - ETA: 0s - loss: 1.3783 - accuracy: 0.4745Restor
ing model weights from the end of the best epoch: 14.

Epoch 00017: val_loss did not improve from 1.19430

Epoch 00017: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
755/755 [==============================] - 274s 363ms/step - loss: 1.3783 - accuracy: 0.47
45 - val_loss: 1.2185 - val_accuracy: 0.5323 - lr: 0.0010
Epoch 00017: early stopping
```

In [6]:

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```

Optimizer : Adam



Testing :)

In [1]:

```
#from keras.preprocessing.image import img_to_array
import cv2
from tensorflow.keras.utils import img_to_array
from keras.models import load_model
import numpy as np
# loading files
haar_file="haarcascade_frontalface_default.xml"
emotion_model='_mini_XCEPTION.102-0.66.hdf5'

cascade=cv2.CascadeClassifier(haar_file)
emotion_classifier=load_model(emotion_model,compile=True)
emotion_names=["angry","disgust","fear", "happy", "sad", "surprise","neutral"]
#frame=cv2.imread('images/disgust_face.jpeg')
#frame=cv2.imread('images/happy_face.jpeg')
frame=cv2.imread('images/sad_face.png')
#frame=cv2.imread('images/me_happy2.jpg')
gray_frame=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
faces=cascade.detectMultiScale(gray_frame,1.5,5)
text=[]
for (x,y,w,h) in faces:
    roi=gray_frame[y:y+h,x:x+w]
    roi=cv2.resize(roi,(64,64))
    roi=roi.astype("float")/255.0
    roi=img_to_array(roi)
    roi=np.expand_dims(roi,axis=0)

    predicted_emotion=emotion_classifier.predict(roi)[0]
    probab=np.max(predicted_emotion)
    label=emotion_names[predicted_emotion.argmax()]
    percen=predicted_emotion*100
    for j in range(7):
        text.append(emotion_names[j]+"  :  "+str(percen[j]))
    for i in range(7):
        #cv2.putText(frame,text[i],(5,i*30+15),cv2.FONT_HERSHEY_SIMPLEX,0.8,(0,255,255),2
        print(text[i])
    cv2.putText(frame,label,(x,y-10),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),1)
```

```
     cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
cv2.imwrite('images/result.jpg', frame)
```

```
WARNING:tensorflow:Error in loading the saved optimizer state. As a result, your model is
starting with a freshly initialized optimizer.
1/1 [==============================] - 2s 2s/step
angry : 21.7511
disgust : 0.054468527
fear : 5.193165
happy : 0.0013106188
sad : 70.66278
surprise : 0.06525041
neutral : 2.2719183
```

Out[1]:

```
True
```

In [ ]:

Recomendation ;)

In [9]:

```
from playsound import playsound
```

In [14]:

```
#labels = ["happy", "angry", "fear", "disgust", "sad", "surprised", "neutral"]
#label = "sad"
tips = {"fear":["Drink water","Get a good night's sleep","Eat wholesome meals","Go for a
        "angry":["Repeat gentle phrases to yourself","Take a walk","Use visualization to
        "sad":["Do things you enjoy (or used to)","Get quality exercise","Eat a nutritiou
       }
website_links  =  {"fear":["https://www.businessinsider.in/science/health/heres-how-to-take
                  "angry":["https://www.thehotline.org/resources/how-to-cool-off-when-your
                  "sad":["https://www.vandrevalafoundation.com/","https://www.healthline.c
                 }
youtube_links = {"fear":["https://www.youtube.com/watch?v=IAODG6KaNBc"],
                  "angry":["https://www.youtube.com/watch?v=P6aPg3YBvBQ"],
                  "sad":["https://www.youtube.com/watch?v=P6aPg3YBvBQ"]
                 }
song_links  =  {"fear":["https://www.youtube.com/watch?v=GyA8ccqwp-4&feature=youtu.be","htt
               "angry":["https://www.youtube.com/watch?v=e74wLJ_KRes&feature=youtu.be","ht
               "sad":["https://www.youtube.com/watch?v=25ROFXjoaAU&feature=youtu.be","http
               "happy":["https://www.youtube.com/watch?v=vGZhMIXH62M","https://www.youtube
              }
 tunes = {"fear":'fear.mp3',
          "angry":'angry.mp3',
          "sad":'sad.mp3'
          }

if (label == "happy"):
    # songs
    print("Here are some song suggestions for your mood:")
    for s in song_links.get('happy'):
        print(s)

elif (label == "angry"):
    # songs
    print("Here are some song suggestions for your mood:")
    for s in song_links.get('angry'):
        print(s)
    # tips
    print("Here are some tips to help you feel better:")
    for i in tips.get('angry'):
        print("-> "+i)
    # resources
    print("Here are some resources that you may find beneficial:")
    for j in website_links.get('angry'):
        print(j)
    for k in youtube_links.get('angry'):
        print(k)
    # tunes
   # print("Here's a tune that will help you calm down.")
    #playsound(tunes.get('angry'))

elif (label == "fear"):
    # songs
    print("Here are some song suggestions for your mood:")
    for s in song_links.get('fear'):
        print(s)
    # tips
    print("Here are some tips to help you feel better:")
    for i in tips.get('fear'):
        print("-> "+i)
    # resources
    print("Here are some resources that you may find beneficial:")
    for j in website_links.get('fear'):
        print(j)
    for k in youtube_links.get('fear'):
        print(k)
    # tunes
    #print("Here's a tune that will make you feel better.")
    #playsound(tunes.get('fear'))

elif (label == "sad"):
    # songs
    print("Here are some song suggestions for your mood:")
    for s in song_links.get('sad'):
        print(s)
    # tips
    print("Here are some tips to help you feel better:")
    for i in tips.get('sad'):
        print("-> "+i)
    # resources
    print("Here are some resources that you may find beneficial:")
    for j in website_links.get('sad'):
        print(j)
    for k in youtube_links.get('sad'):
        print(k)
    # tunes
    #print("Listen to a tune that will soothe you.")
    #playsound(tunes.get('sad'))
```

```
Here are some song suggestions for your mood:
https://www.youtube.com/watch?v=25ROFXjoaAU&feature=youtu.be
https://www.youtube.com/watch?v=BzE1mX4Px0I
Here are some tips to help you feel better:
-> Do things you enjoy (or used to)
-> Get quality exercise
-> Eat a nutritious diet
-> Challenge negative thinking
Here are some resources that you may find beneficial:
https://www.vandrevalafoundation.com/
https://www.healthline.com/health/depression/recognizing-symptoms#fatigue
https://www.youtube.com/watch?v=P6aPg3YBvBQ
```

```
In [ ]:
```