Predicting the Attention Rate of Patients by Monitoring their Emotions

- Name: Akshara Shukla
- Class: AI43
- Genuine Challenge_01

Introduction

This project consists of creating a report on aiding doctors in large multinational hospitals to understand different patient behaviors. Fundamentally, the project goal is to showcase the power of Artificial Intelligence and Machine Learning in healthcare and how to provide a better patient experience by classifying them on their emotions, mainly the four basic emotions (happy, sad, angry, and relaxing) and accompanying the ones that are in pain and require immediate or more attention and care. The system is Areya and it aims at providing some portrait of notification to the doctors about patients who illustrate negative emotions i.e., angry and sad.

This notebook consists the testing of the dataset FER-2013 in real time by using the default camera attached in my pc. The algorithm chosen was building a convolutional neural network.

Setup

We start with importing needed/required libraries.

In [2]:

```
from keras.models import load_model  #loading the saved model from pc
from time import sleep
from keras.preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2 #for conducting computer vision
import numpy as np #for rescaling and normalising face pixels detected from camera
```

We will start our testing by first loading a **haarcascade_frontalface_default.xml**. This is an xml file which is provided by cv2 library for detecting face in any video frame. This file was taken from the github repository of cv2. Following, loading our model from the path defined on computer.

In [3]:

```
#file to detect a face in our video frame
face_classifier = cv2.CascadeClassifier(r'C:\Users\PC\Evernote\Logs\emotion_recognition\haarcascade_frontalface_default.xml')

#loading the model
model =load_model(r'C:\Users\PC\Evernote\Logs\emotion_recognition\my_h5_model.h5')
```

Explicitly specifying the different emotions which will be exhibited by patients.

In [4]:

```
emotions = ['Angry','Happy','Relaxed','Sad']
```

In order to get the camera turned on, using the .VideoCapture() function provided by cv2 library and setting the paramter to 0, since I want to turn my default camera from my pc on and assigning it to the variable cam.

In [5]:

```
cam = cv2.VideoCapture(0)
```

Specifying the detailed steps which will be carried while the camera is turned on. Following will be the steps carried out in brief:

- It will essentially start by first detecting if a face is presented in the camera frame which is carried out with the help of cam.read() function. Additionally, it will be saving the face and forming a rectangle around it which will have our region of interest i.e. the facial expression of an individual or the patients in our project.
- After detecting the region of interest, it will be normalizing the pixel values to our default image dimensions of our dataframe, which is 48x48. And predicting the emotions of our patients with their attention rate.
- When emotions displayed aren't Happy and Relaxed, a message of "Needs Immediate Attention" would pop up.
- In case of no face getting detected, the model would return a 'No face detected' message.

In [6]:

```
while True:
    # Grab a single frame of video
    ret, frame = cam.read() #read the face from the camera/frame
    #labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY) #convert it into grayscale(train data =
 grayscale)
    #putting neighbors as 5 which is similar to the hyperparamter of knn, where k = 5
    #so if our model is getting confused in detecting the emotion, it can understand the 5
nearest labels and then further predict
    faces = face_classifier.detectMultiScale(gray,1.3,5) #scale down our frame/input img b
ecause in the webcam, the pixel values will be very high

    #draw a rectange around our face
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        #in our region of interest, we want to convert it into gray to take it as an input
        roi_gray = gray[y:y+h,x:x+w]
        #resizing it to the original size of our img
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
    # rect,face,image = face_detector(frame)

        #when there is a face detected
        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi, axis = 0)

        # make a prediction on the ROI, then lookup the class
            preds = model.predict(roi)[0]
            label=emotions[preds.argmax()]
            label_position = (x,y)
            #using cv2.font_hershey_simplex to diplay the text on the camera with color gr
een and size of 3.
            cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
            #Inserting the conditions to display the message when emotions aren't Happy an
d Relaxed on the camera
            if(label != 'Happy' and label != 'Relaxed'):
                cv2.putText(frame,'Needs Immediate Attention',(20,60),cv2.FONT_HERSHEY_SIM
PLEX,2,(0,255,0),3)
            else:
                cv2.putText(frame,'',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
        else:
            cv2.putText(frame,'No Face Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0
),3)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

The below chunk helps to close the window which opens when the above chunk is running.

In [7]:

```
cam.release()
cv2.destroyAllWindows()
```

Conclusion

To conclude, the haarcasade file from the cv2 library worked perfectly in detecting if a face is present in the frame or not. It didn't get distracted by any light dimness or excessive brightness. Our model did a great job in detecting emotions mainly Happy, as seen from the confusion matrix, all the emotions were merely classified as Happy. Having said this, the goal for future implementation is to improve the performance in detecting other emotions.