**Real time detection of driver drowsiness and alerting using OpenCV**

**Introduction**

This study shows a surveillance system developed to detect and alert driver about the presence of drowsiness. This system works adequately under different lighting conditions, already trained dataset was used. haar cascade classifier is used to detect faces

**Dataset used:**

7000 images of people in different lighting conditions were taken and trained, the images consist of open and closed eyes.

This dataset was already trained using CNN in keras, and we got it from a public source and applied it in our source code.

**Methods:**

**Prerequisites:**

The requirements for this python project are a webcam, and python version (3.6 recommended)

Installation of below necessary packages should be done before starting the project.

1. OpenCV – used for face and eye detection.
2. Tenser flow – keras uses TensorFlow as backend.
3. Keras – used to build classification model.
4. Pygame – used to play alarm sound.

Let’s understand how our code works:

**Step-1: We took image as input from a camera**

Here webcam was used to images as input. Here we made infinite loop to access webcam which means that will capture each frame.

We used this method provided by OpenCV, **cv2.VideoCapture(0)** to access the camera and to set the capture object(cap), **cap.read()** to read each frame and stored image variable in a frame variable.

**Step-2: We detected face in the image and created a region of interest**

Here to detect the face in the image, we converted image into grayscale as the OpenCV algorithm for object detection takes Gray images in the input. We don’t need colour information to detect the objects. We used haar cascade classifier to detect faces. To set our classifier we used **face = cv2.CascadeClassifier(‘ path to our haar cascade xml file’)**. Then we performed the detection using **faces = face.detectMultiScale(gray)**. Which returns an array of detections with coordinates (x,y), and width of the boundary box of the object, and height.

So that now we can iterate the faces and draw boundary boxes for each face.

**Step-3: We detected the eyes from region of interest and feed it to the classifier**

procedure used to detect faces is used to detect eyes. We set cascade classifier for eyes in left eye(**leye)** and right eye(reye) respectively then detect the eyes using **left\_eye = leye.detectMultiScale(gray)**. To extract eye data from the full image we extracted the boundary of the eye and then we pulled out the eye image form the frame using this code.

**Step-4: classifier is used to categorize whether eyes are open or closed**

Here CNN classifier used for predicting the eye status. Certain operations are done to feed our image into the model because the model needs the correct dimensions to start with. we convert the colour image into grayscale using **r\_eye = cv2.cvtColor(r\_eye, cv2.COLOR\_BGR2GRAY)**.

Then we resized the image to 24\*24 pixels as our model was trained on 24\*24 pixel images **cv2.resize(r\_eye, (24,24))**.

Then we normalized the data for better convergence r**\_eye = r\_eye/255**(all values will be between 0-1). We expand the dimensions to feed into our classifier. Loaded our model using **model = load\_model(‘models/cnnCat2.h5’).** then we predicted each eye with our model **lpred = model.predict\_classes(l\_eye)**. If the value of lpred[0] = 1, it states that eyes or open, if value of lpred[0] = 0 then, it states that eyes are closed.

**Step-5: we calculated score to check whether person is drowsy**

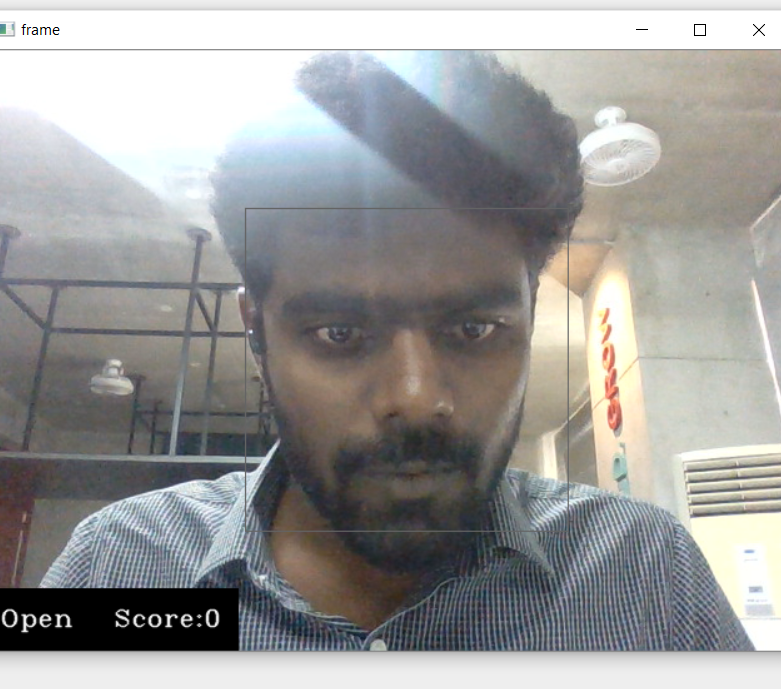
score is basically a value used to determine hoe long the person has closed his eyes. If both eyes are closed, then we will keep on increase the score and when eyes are open, we decrease the score.

The result is drawn on the screen using cv2.putText() function which display real time status of the person.

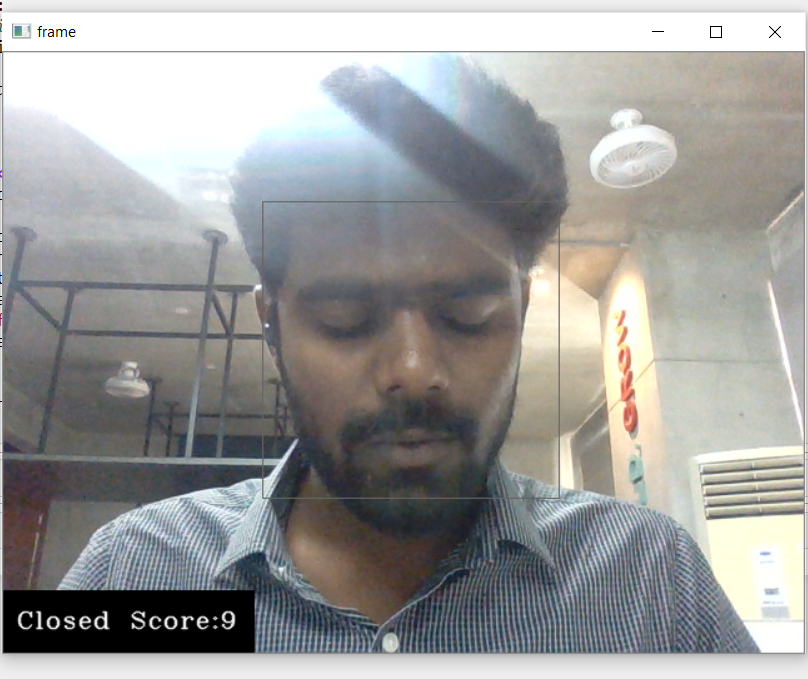
A threshold is defined for example if score becomes greater than 15 that means the person’s eyes are closed for a long period of time. This is when the alarm beep will played using **sound.play()**

**Results:**

**Open eye**



**Closed eye:**



**Sleep alert:**

