**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

**Project Final Report**

**for**

**Drowsiness Detection System**

***By***

**Hammad Ahmed Tahir CU/FA17-BCS-025/ISB**

***Supervisor*Ms Shehla Saif**

Submission Date: 05-08-2020

***Bachelor of Science in Computer Science (2017-2021)***

# 

[Overview:](#_tk4cto8kj4pj)

[Constituents of the Project:](#_4ays1jd1kslz)

[The Dataset:](#_f5sykih7188r)

[The Model Architecture:](#_ijfzw27rkrly)

[Tools and Technologies:](#_m6yas9mdf16y)

[Contents of the File Attached:](#_pnh22e7nvwhz)

[Working:](#_lbtdmpidr9ea)

[Step 1 - Get image input:](#_1wn4iker3gp)

[Step 2 - Identify Face in the image:](#_ij3a7hqvqlur)

[Step 3 - Detect eyes from RIO and pass it to Classifier:](#_eowiyl5yx6sg)

[Step 4 - Classify open/closed eyes:](#_g6lryy7nnjvr)

[Step 5 - Calculate Score to Check Drowsiness:](#_e12ntjtzyrl2)

[Working Snapshots:](#_7alqsaaeplvr)

[Link to Git Repo of code:](#_g2rovavk3oam)

# 

# Overview:

With this Python venture, we will make a Drowsiness Detection gadget. An incalculable number of individuals drive on the parkway day and night. Cab drivers, transport drivers, truck drivers and individuals voyaging significant distance experience the ill effects of absence of rest. Because of which it turns out to be risky to drive when feeling languid.

Most mishaps occur because of the sleepiness of the driver. Along these lines, to forestall these mishaps we will construct a framework utilizing Python, OpenCV, and Keras which will alarm the driver when he feels sluggish/lazy.

Drowsiness detection is a safety technology capable of preventing accidents caused by drivers who fall asleep while driving.

# Constituents of the Project:

Let me explain the datasets, model architecture and technologies used to develop this project.

## The Dataset:

The dataset for this project contains 7000 images of human eyes distributed under the respective labels of open and closed. The dataset has been manually cleaned and gotten rid of the unwanted images. The data comprises of images in various lighting conditions.

## The Model Architecture:

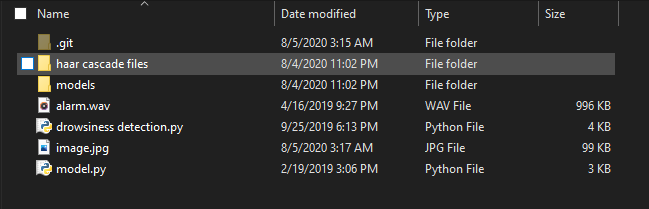
The model used here is build using Convolutional Neural Network (CNN) with the assistance of Keras. A CNN works proficiently well for the purpose of image classification. It contains an input, output and a hidden (intermediate) layer, that in turn contains various other layers. On such layers, a convolution operation is performed using a filter that multiplies 2D matrix on layer and filter. Luckily I had gotten my hands on the trained model on the above mentioned dataset.

## Tools and Technologies:

Following tools and technologies are used while developing the project.

* Python 3.6
* Anaconda 3
* Spyder (IDE)
* OpenCV
* Keras
* Tenserflow (CPU)
* Pygame

## Contents of the File Attached:



# Working:

## Step 1 - Get image input:

The webcam is used to get the image input. An infinite loop is made to capture every frame. For this purpose, OpenCV is used. The methods **cv2.VideoCapture(0)** and **cap.read()** are used to access camera and capture frames respectively.

## Step 2 - Identify Face in the image:

The files inside the directory haar cascade files contain the xml files required to detect face and eyes.As the OpenCV method for object detection requires grayscale input, the images from the webcam must first be converted into grayscale. After that haar cascade classifier is used to detect faces. The classifier is set using the line, **face = cv2.CascadeClassifier(‘ path to our haar cascade xml file’).** For face detection, we use, **faces = face.detectMultiScale(gray).** It returns an detection array of x,y coordinates, height and width. Now we can iterate over faces to draw boundary boxes. Hence creating a Region of Interest (ROI).

## Step 3 - Detect eyes from RIO and pass it to Classifier:

The same process done to detect face, is also used to detect eyes using a different cascade classifier, one for left and eye other for the right. The data for left and right eyes is then transferred to CNN classifier that tells whether the eyes are open or closed.

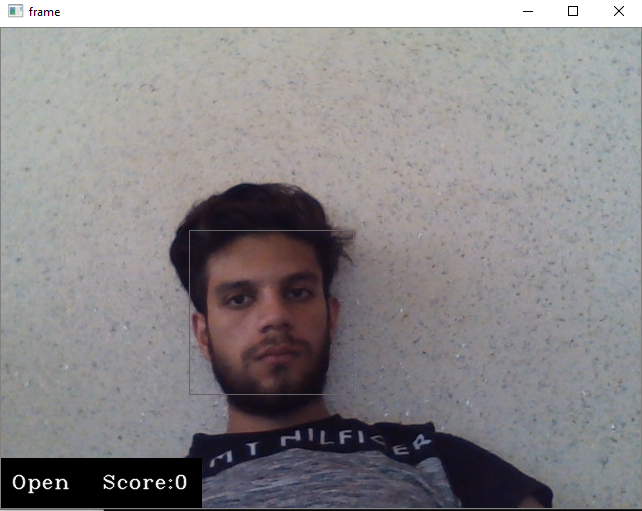
## Step 4 - Classify open/closed eyes:

The CNN classifier is used for this purpose. But first we need to convert our image data into grayscale, resize it to 24\*24 pixels, and later normalise for satisfactory convergence. Now we load our model and feed the data to it. If it returns 1, the eyes are open and if 0, the eyes are closed.

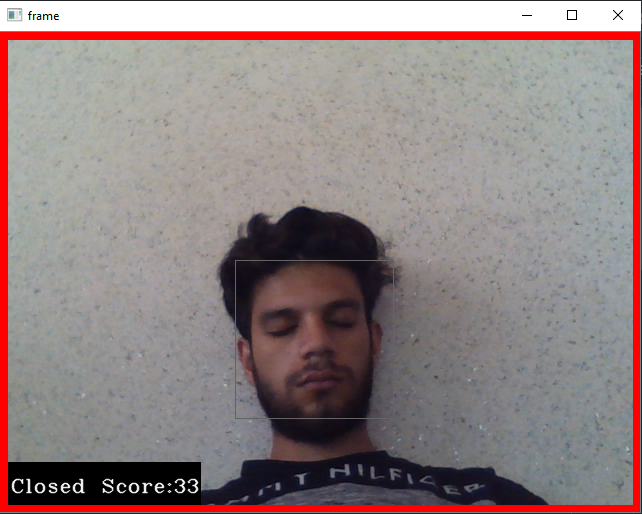
## Step 5 - Calculate Score to Check Drowsiness:

The score determines the period for which the person has closed his/her eyes. It’s threshold is 15. If the value goes any higher than 15, the alarm beeps and keeps on beeping until the person opens his/her eyes and the score decreases down to 0.

# Working Snapshots:



**Fig-1: Eyes Open, not beeping**



**Fig-2: Eyes Closed, Red border, Alarm beeping**

# Link to Git Repo of code:

<https://github.com/HammadAhm3d/Drowsiness_Detection_Python.git>

## 