Report On

Driver Drowsiness Detection

Submitted in partial fulfillment of the requirements of the Mini project in Semester VI of Third Year Artificial Intelligence and Data Science Engineering

by

Shlok Gaikwad	(Roll No. 04)
Priyanshu Kamble	(Roll No. 07)
Dhruv Mewada	(Roll No. 11)

Mentor
Prof. Sneha Yadav

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(A.Y. 2022-23)

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the Mini Project entitled **“Driver Drowsiness Detection”** is a Bonafide work of **Shlok Gaikwad (04), Priyanshu Kamble (07), Dhruv Mewada (11)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in Semester VI of Third Year **“Artificial Intelligence and Data Science”.**

Prof. Sneha Yadav
Guide

Prof. Sejal D’mello
Deputy HOD AI & DS

Dr. Vikas Gupta
HOD AI &DS

Dr. H.V. Vankudre
Principal

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

Mini Project Approval

This Mini Project entitled **“Driver Drowsiness Detection”** by **Shlok Gaikwad (04), Priyanshu Kamble (07), Dhruv Mewada (11)** is approved for the degree of **Bachelor of Engineering** in Semester VI of Third Year **Artificial Intelligence and Data Science.**

Examiners

1……………………………………….
(Internal Examiner Name & Sign)

2………………………………………….
(External Examiner name & Sign)

Date: 28/04/2023

Place: Vasai

Contents

Abstract

Driver drowsiness is a significant cause of accidents on the road, leading to fatalities and injuries. In this context, a driver drowsiness detection system using OpenCV can be an effective solution to detect when a driver is becoming drowsy and alert them to take a break, enhancing road safety. This system uses OpenCV's pre-trained models for facial landmark detection to track key features on the driver's face, such as the eyes, eyebrows, and mouth. By analyzing the position and movement of these features, the system can accurately detect when the driver is becoming drowsy and trigger an alert. The system uses techniques such as eye aspect ratio (EAR) calculation and yawn detection to detect driver drowsiness accurately. Additionally, the system can store data on the driver's drowsiness level and driving behavior, providing valuable insights for analysis and reporting. This project proposes to implement a driver drowsiness detection system using OpenCV and evaluate its performance under different driving conditions. The implementation of such a system can be a cost-effective and efficient way to detect driver drowsiness and enhance overall safety on the roads.

Acknowledgment

We would like to express our sincere gratitude to our advisor **Prof. Sneha yadav** for the continuous support of our study and research, for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped us in all the time of research and writing of this thesis. We could not have imagined having a better advisor and mentor for our study.

INTRODUCTION

1.1 Introduction

Driver drowsiness is a major cause of accidents on the road, leading to fatalities and injuries. According to research, drowsy driving can impair a driver's reaction time, attention span, and decision-making ability. Therefore, it is crucial to detect and prevent driver drowsiness to ensure road safety. A driver drowsiness detection system can be an effective solution to detect when a driver is becoming drowsy and alert them to take a break.OpenCV, an open-source computer vision library, provides tools and pre-trained models for facial landmark detection, which can be used to track key features on the driver's face, such as the eyes, eyebrows, and mouth. By analyzing the position and movement of these features, the system can accurately detect when the driver is becoming drowsy and trigger an alert to prompt the driver to take a break.In this context, this project proposes a driver drowsiness detection system using OpenCV. This system can be integrated into a vehicle to detect drowsiness in real-time and provide alerts to the driver, enhancing road safety. The system will use techniques such as eye aspect ratio (EAR) calculation and yawn detection to detect driver drowsiness accurately. Additionally, the system can store data on the driver's drowsiness level and driving behavior, providing valuable insights for analysis and reporting.Overall, a driver drowsiness detection system using OpenCV can be a cost-effective and efficient way to detect driver drowsiness and enhance overall safety on the roads. This project aims to implement such a system and evaluate its performance under different driving conditions.

1.2 Problem Statement

Driver drowsiness is a significant issue that can result in accidents, injuries, and fatalities on the road. Drowsiness is a state of impaired cognitive and physical function, which can occur due to various factors such as sleep deprivation, medication, and alcohol consumption.The problem statement for driver's drowsiness detection is to develop a system that can accurately detect driver drowsiness in real-time. This system should take into account various factors such as the driver's behavior, eye movements, and facial expressions to determine their level of drowsiness. It should also be able to differentiate between normal driving behavior and drowsy driving behavior.The system should be able to alert the driver when they are getting drowsy and provide suggestions to help them stay awake and alert. Additionally, the system should also notify relevant authorities or emergency services if it detects a severely drowsy driver who is at high risk of causing an accident.

1.3 Scope

The scope of a driver's drowsiness detection system is to develop a system that can detect driver drowsiness in real-time and provide alerts and suggestions to prevent accidents caused by drowsy driving. The system should use various sensors and technologies such as cameras, infrared sensors, and accelerometers to monitor the driver's behavior, eye movements, and facial expressions. The system should be able to differentiate between normal driving behavior and drowsy driving behavior and accurately detect the level of drowsiness. It should also be able to adapt to different driving conditions and environments and work in various lighting conditions and weather conditions.The system should provide visual, auditory, and haptic feedback to alert the driver when they are getting drowsy and provide suggestions to help them stay awake and alert. The system should have a user-friendly interface that is easy to use and understand and should be able to notify relevant authorities or emergency services if it detects a severely drowsy driver who is at high risk of causing an accident.The scope of a driver's drowsiness detection system is to improve road safety by detecting and preventing accidents caused by drowsy driving. The system can be used in various settings such as personal vehicles, commercial vehicles, and public transportation to ensure the safety of all road users.

LITERATURE SURVEY

2.1 Survey of Existing System/SRS

The current detection system uses a random-cost iris sensor for drowsy driving and an easy-to-deploy sensor model, which is particularly suitable for measuring impaired cycling and hand position. These sensors are often used in the active safety system of cars to identify the driver. fatigue is a major problem in stopping traffic accidents. The main point of this approach is to develop a prototype of the sensor blocks to act as a platform for the integration of various sensors in the wheel. The bike slowed down or stopped to calculate the condition. An existing identical system uses an adaptive driver to detect drowsiness. Truck drivers and shift workers are at the highest risk of falling asleep while driving. Most of the accidents happen due to drunkenness. This survey is ready to understand the requirements and conditions of the entire population and as such we tried to go through various websites and applications and searched for basic information.

2.2 Limitation Existing system or Research gap

Some limitations and research gaps in the existing driver's drowsiness detection systems , While the existing systems are quite accurate in detecting drowsiness, there is still room for improvement. For example, some systems may have false positives, i.e., they may detect drowsiness when the driver is actually alert. Many systems require calibration for each driver, which can be time-consuming and inconvenient. There is a need for more robust and generalizable models that can work with different drivers without calibration. Systems may be affected by environmental factors such as lighting conditions, weather, and road conditions.

2.3 Mini Project Contribution

The contribution of driver's drowsiness detection to society is significant in terms of enhancing road safety, reducing accidents, and saving lives. Drowsy driving is a major cause of road accidents, and detecting drowsiness in drivers can prevent accidents caused by driver fatigue. By detecting drowsiness in drivers, drowsiness detection systems can alert drivers to take a break, change their driving behavior, or take other measures to avoid accidents. This can help prevent accidents and save lives, as drowsy driving can be just as dangerous as drunk driving.

PROPOSED SYSTEM

3.1 Introduction

The proposed system for driver drowsiness detection aims to accurately detect when a driver is becoming drowsy and alert them to take a break, enhancing road safety. The system will use OpenCV's pre-trained models for facial landmark detection to track key features on the driver's face, such as the eyes, eyebrows, and mouth. By analyzing the position and movement of these features, the system can accurately detect when the driver is becoming drowsy and trigger an alert.The proposed system will use the following techniques to detect driver drowsiness accurately..Overall, the proposed system using OpenCV can be a cost-effective and efficient way to detect driver drowsiness and improve road safety. It can be easily integrated with existing vehicle systems and customized to suit the specific needs of different drivers and driving conditions.The proposed system will provide alerts to the driver in real-time, encouraging them to take a break and avoid accidents. Additionally, the system can store data on the driver's drowsiness level and driving behavior, providing valuable insights for analysis and reporting.ImplementationWhen the driver is driving, the driver's face is captured by a camera and it is converted into a video stream. The application then analyzes the video to detect drowsiness and also checks the level of drowsiness. In this stage, the main parts which should be considered for analysis are: the driver's face tracking, and recognition of key regions of the face based on eye closure and yawning . Finally, if the drowsiness is detected, a warning voice alert is given. High-Level System Architecture consisting of the input to the model and the preprocessing and evaluation in stages. Stage 1 involves pre-processing of video streams for human face tracking. Stage 2 involves extraction of facial key regions such as eyes and mouth. Stage 3 involves detection of drowsiness symptoms like eye closure, blinking, and yawning.

3.2 Architecture

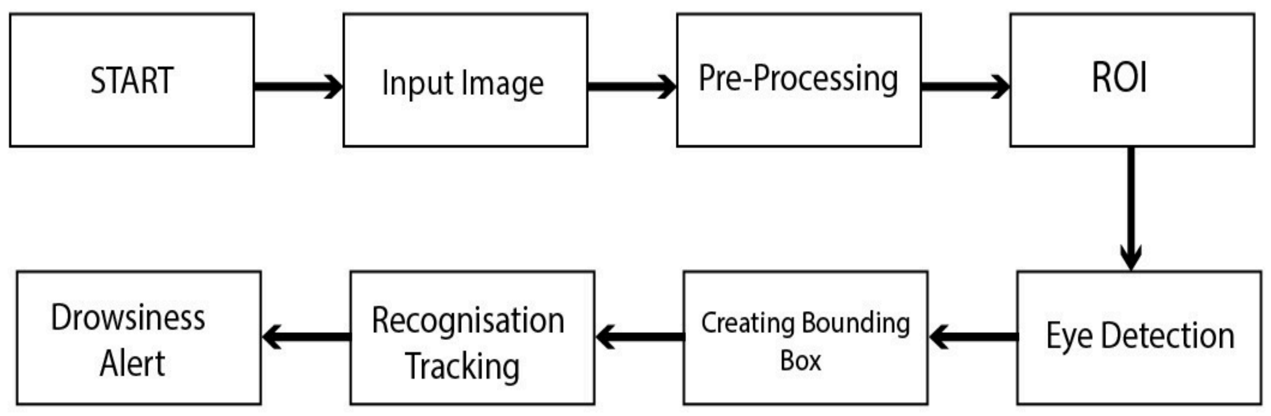


Fig 1:- Architecture

3.3 Details of Hardware/Software Used:-

1.Hardware Requirement:

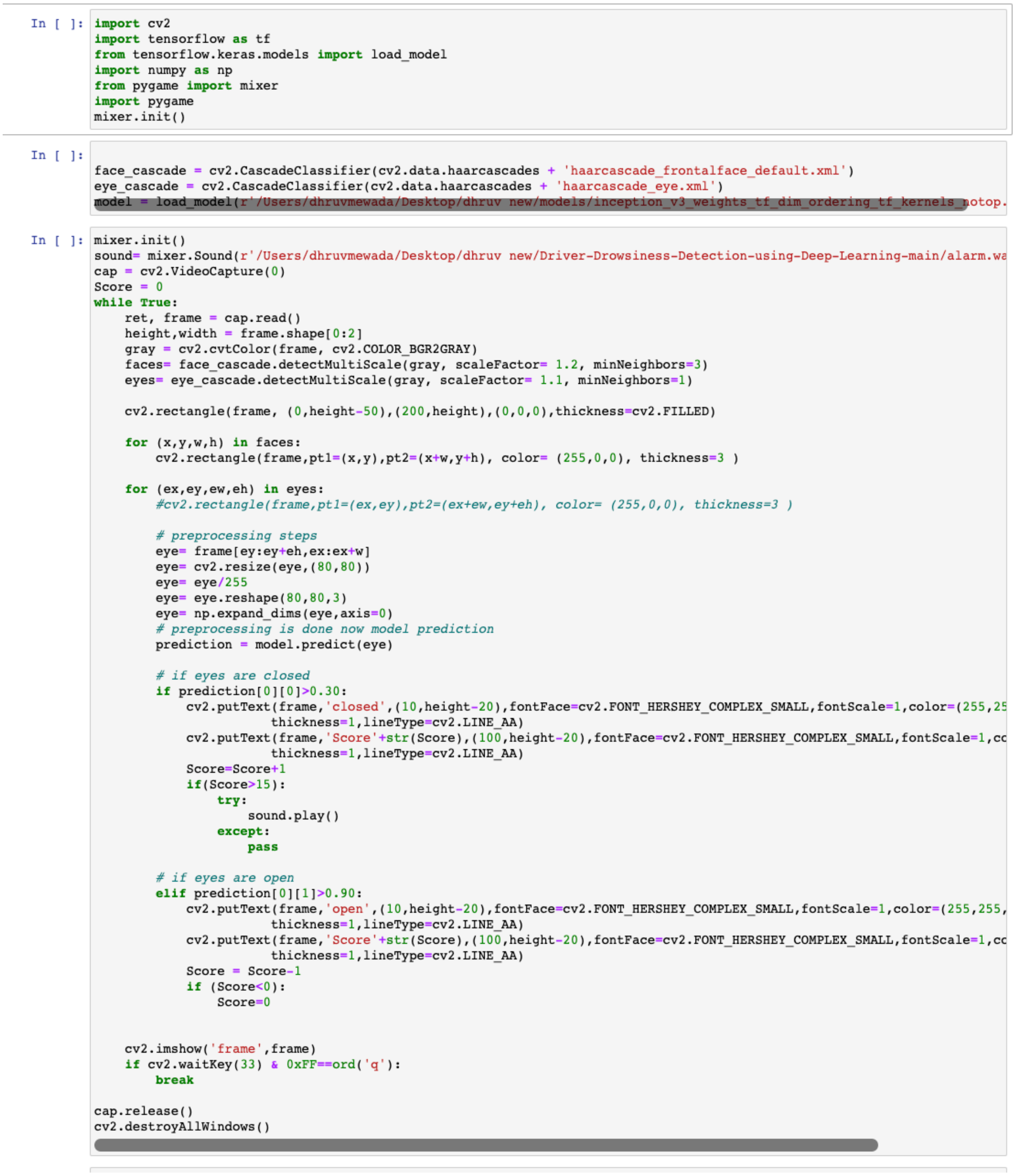
- I7 Processor Based Computer or higher
- Memory: 8 GB RAM
- Hard Drive: 256 GB
- Monitor
- Internet Connection

2.Software Requirement:

- Windows 10
- Programming language: Python
- IDE used: Jupyter Notebook
- Technology used: Opencv ,Tkinter, Dlib, Numpy.

3.4.Experiment and Results for Validation and Verification:

Main.ipynb:-

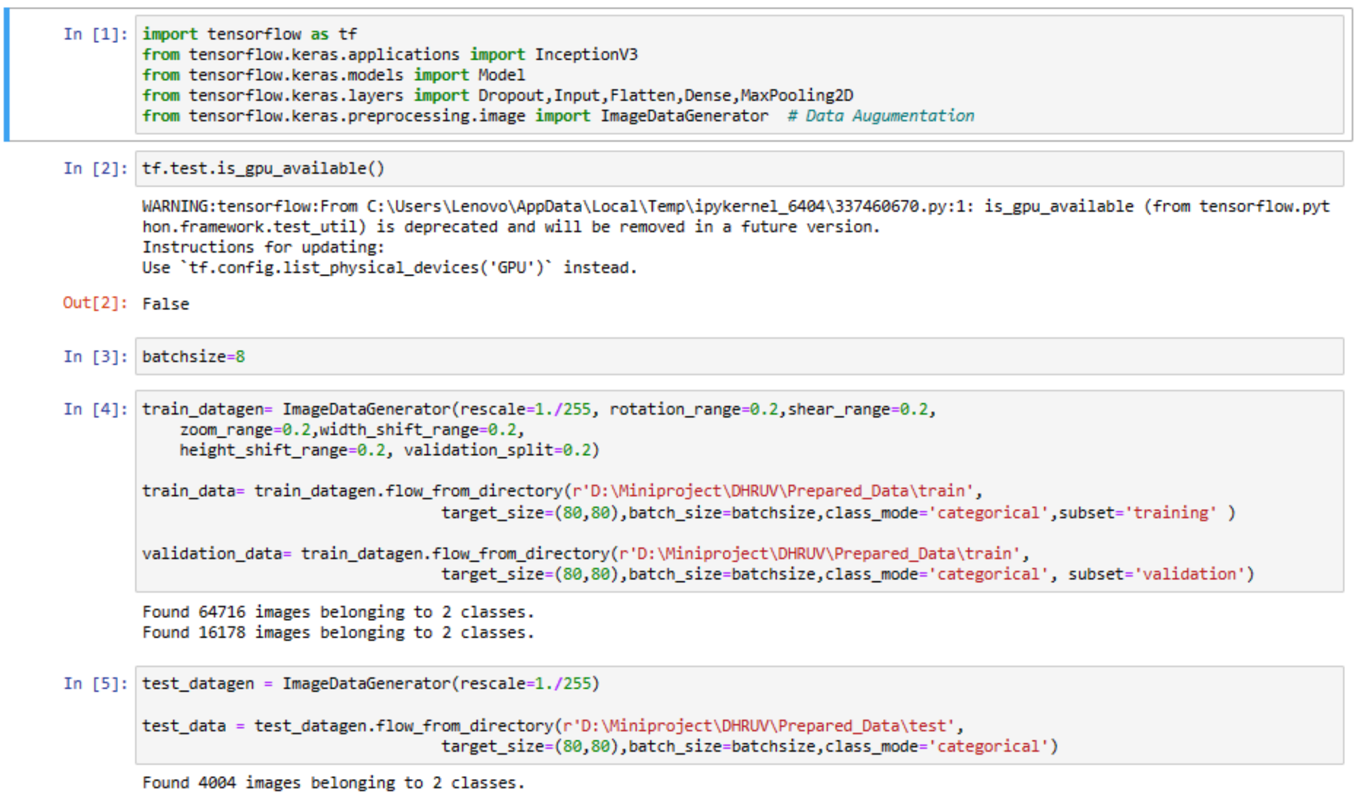


```
In [ ]: import cv2
        import tensorflow as tf
        from tensorflow.keras.models import load_model
        import numpy as np
        from pygame import mixer
        import pygame
        mixer.init()
```

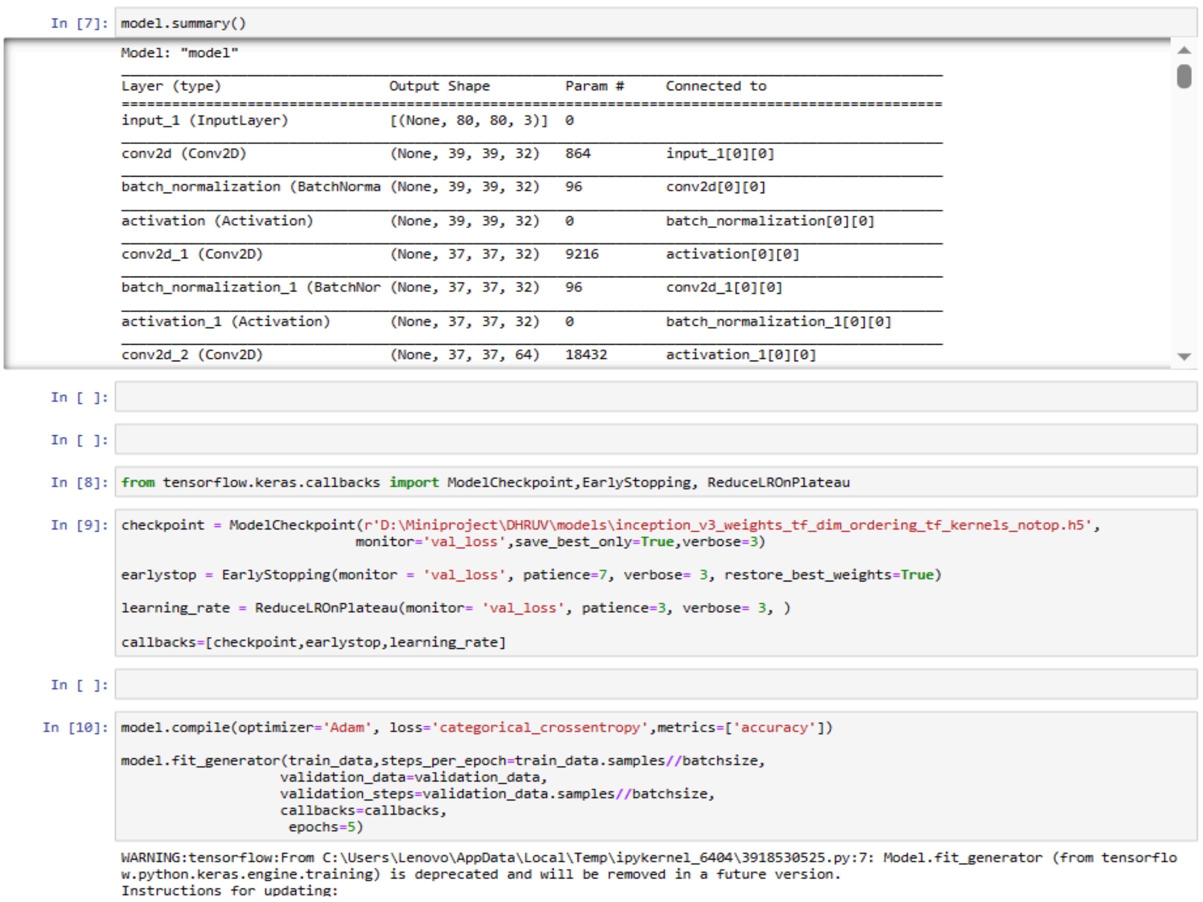
```
In [ ]:
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
        model = load_model(r'/Users/dhruvmewada/Desktop/dhruv new/models/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.
```

```
In [ ]: mixer.init()
        sound= mixer.Sound(r'/Users/dhruvmewada/Desktop/dhruv new/Driver-Drowsiness-Detection-using-Deep-Learning-main/alarm.wa
        cap = cv2.VideoCapture(0)
        Score = 0
        while True:
            ret, frame = cap.read()
            height,width = frame.shape[0:2]
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
            eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)

            cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )

            for (ex,ey,ew,eh) in eyes:
                #cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3 )

                # preprocessing steps
                eye= frame[ey:ey+eh,ex:ex+w]
                eye= cv2.resize(eye,(80,80))
                eye= eye/255
                eye= eye.reshape(80,80,3)
                eye= np.expand_dims(eye,axis=0)
                # preprocessing is done now model prediction
                prediction = model.predict(eye)

                # if eyes are closed
                if prediction[0][0]>0.30:
                    cv2.putText(frame,'closed',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,25
                                thickness=1,lineType=cv2.LINE_AA)
                    cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,cc
                                thickness=1,lineType=cv2.LINE_AA)
                    Score=Score+1
                    if(Score>15):
                        try:
                            sound.play()
                        except:
                            pass

                # if eyes are open
                elif prediction[0][1]>0.90:
                    cv2.putText(frame,'open',(10,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,
                                thickness=1,lineType=cv2.LINE_AA)
                    cv2.putText(frame,'Score'+str(Score),(100,height-20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,cc
                                thickness=1,lineType=cv2.LINE_AA)
                    Score = Score-1
                    if (Score<0):
                        Score=0


            cv2.imshow('frame',frame)
            if cv2.waitKey(33) & 0xFF==ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()
```

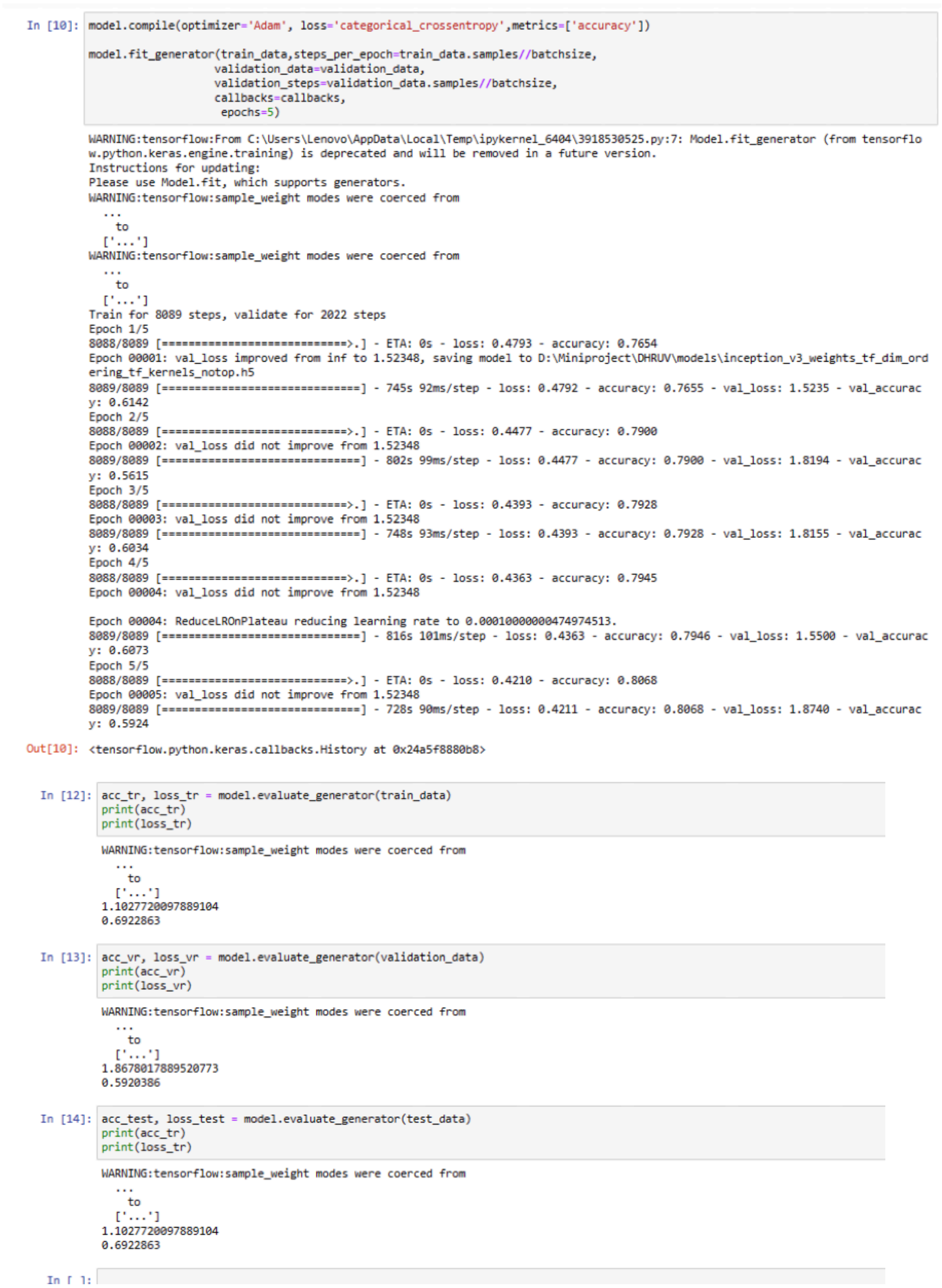
model_training.ipynb:-



```
In [1]: import tensorflow as tf
        from tensorflow.keras.applications import InceptionV3
        from tensorflow.keras.models import Model
        from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
        from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augumentation

In [2]: tf.test.is_gpu_available()

        WARNING:tensorflow:From C:\Users\Lenovo\AppData\Local\Temp\ipykernel_6404\337460670.py:1: is_gpu_available (from tensorflow.pyt
        hon.framework.test_util) is deprecated and will be removed in a future version.
        Instructions for updating:
        Use `tf.config.list_physical_devices('GPU')` instead.

Out[2]: False

In [3]: batchsize=8

In [4]: train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
            zoom_range=0.2,width_shift_range=0.2,
            height_shift_range=0.2, validation_split=0.2)

        train_data= train_datagen.flow_from_directory(r'D:\Miniproject\DHRUV\Prepared_Data\train',
                                        target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )

        validation_data= train_datagen.flow_from_directory(r'D:\Miniproject\DHRUV\Prepared_Data\train',
                                        target_size=(80,80),batch_size=batchsize,class_mode='categorical', subset='validation')

        Found 64716 images belonging to 2 classes.
        Found 16178 images belonging to 2 classes.

In [5]: test_datagen = ImageDataGenerator(rescale=1./255)

        test_data = test_datagen.flow_from_directory(r'D:\Miniproject\DHRUV\Prepared_Data\test',
                                        target_size=(80,80),batch_size=batchsize,class_mode='categorical')

        Found 4004 images belonging to 2 classes.
```

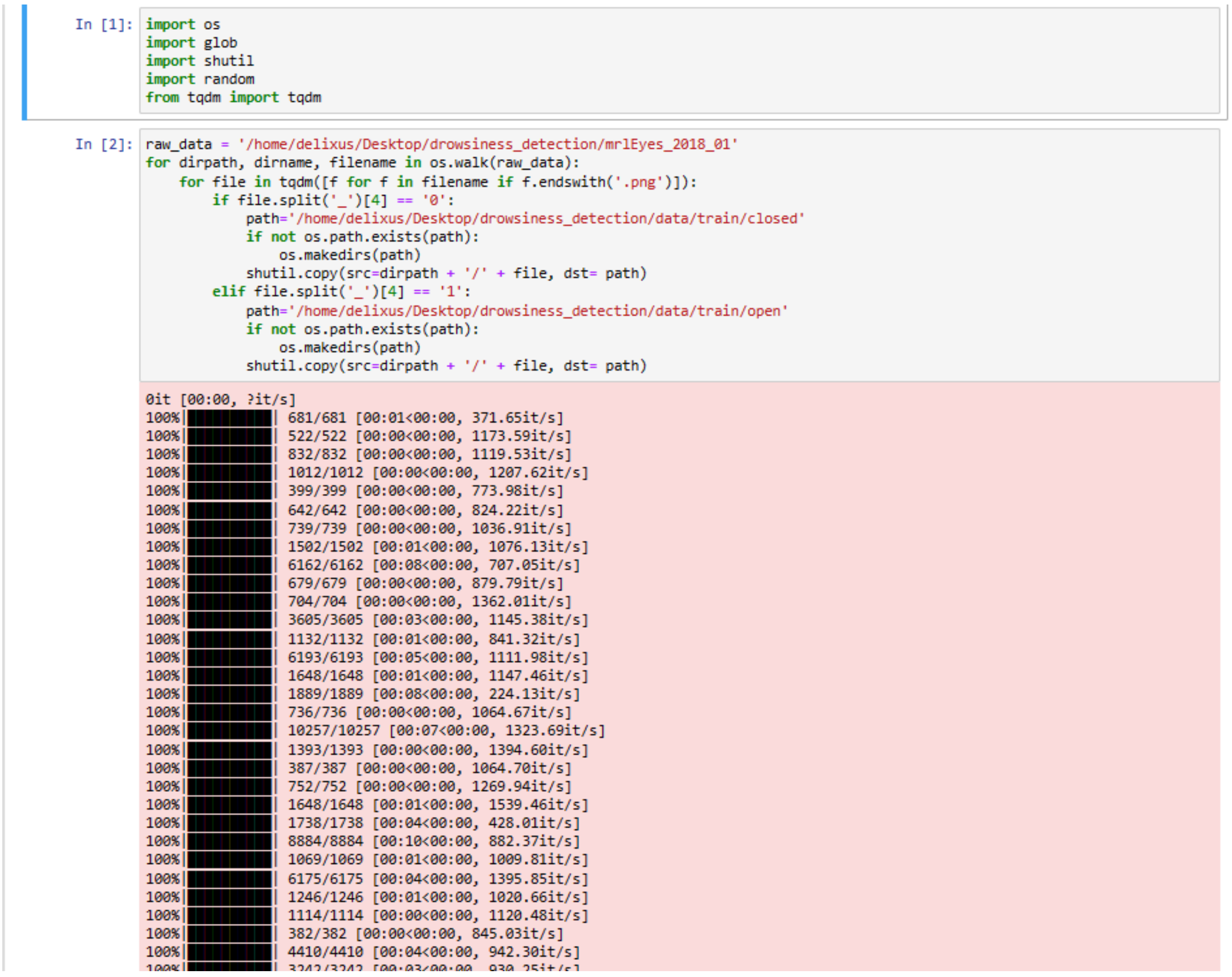


```
In [7]: model.summary()

        Model: "model"
        __________________________________________________________________________________________________
        Layer (type)                    Output Shape         Param #     Connected to
        ==================================================================================================
        input_1 (InputLayer)            [(None, 80, 80, 3)]  0
        __________________________________________________________________________________________________
        conv2d (Conv2D)                 (None, 39, 39, 32)   864         input_1[0][0]
        __________________________________________________________________________________________________
        batch_normalization (BatchNorma (None, 39, 39, 32)   96          conv2d[0][0]
        __________________________________________________________________________________________________
        activation (Activation)         (None, 39, 39, 32)   0           batch_normalization[0][0]
        __________________________________________________________________________________________________
        conv2d_1 (Conv2D)               (None, 37, 37, 32)   9216        activation[0][0]
        __________________________________________________________________________________________________
        batch_normalization_1 (BatchNor (None, 37, 37, 32)   96          conv2d_1[0][0]
        __________________________________________________________________________________________________
        activation_1 (Activation)       (None, 37, 37, 32)   0           batch_normalization_1[0][0]
        __________________________________________________________________________________________________
        conv2d_2 (Conv2D)               (None, 37, 37, 64)   18432       activation_1[0][0]

In [ ]:

In [ ]:

In [8]: from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping, ReduceLROnPlateau

In [9]: checkpoint = ModelCheckpoint(r'D:\Miniproject\DHRUV\models\inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5',
                                    monitor='val_loss',save_best_only=True,verbose=3)

        earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_best_weights=True)

        learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

        callbacks=[checkpoint,earlystop,learning_rate]

In [ ]:

In [10]: model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])

         model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                             validation_data=validation_data,
                             validation_steps=validation_data.samples//batchsize,
                             callbacks=callbacks,
                              epochs=5)

         WARNING:tensorflow:From C:\Users\Lenovo\AppData\Local\Temp\ipykernel_6404\3918530525.py:7: Model.fit_generator (from tensorflo
         w.python.keras.engine.training) is deprecated and will be removed in a future version.
         Instructions for updating:
```



```
In [10]: model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])

         model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                             validation_data=validation_data,
                             validation_steps=validation_data.samples//batchsize,
                             callbacks=callbacks,
                              epochs=5)

         WARNING:tensorflow:From C:\Users\Lenovo\AppData\Local\Temp\ipykernel_6404\3918530525.py:7: Model.fit_generator (from tensorflo
         w.python.keras.engine.training) is deprecated and will be removed in a future version.
         Instructions for updating:
         Please use Model.fit, which supports generators.
         WARNING:tensorflow:sample_weight modes were coerced from
           ...
             to
           ['...']
         WARNING:tensorflow:sample_weight modes were coerced from
           ...
             to
           ['...']
         Train for 8089 steps, validate for 2022 steps
         Epoch 1/5
         8088/8089 [============================>.] - ETA: 0s - loss: 0.4793 - accuracy: 0.7654
         Epoch 00001: val_loss improved from inf to 1.52348, saving model to D:\Miniproject\DHRUV\models\inception_v3_weights_tf_dim_ord
         ering_tf_kernels_notop.h5
         8089/8089 [==============================] - 745s 92ms/step - loss: 0.4792 - accuracy: 0.7655 - val_loss: 1.5235 - val_accurac
         y: 0.6142
         Epoch 2/5
         8088/8089 [============================>.] - ETA: 0s - loss: 0.4477 - accuracy: 0.7900
         Epoch 00002: val_loss did not improve from 1.52348
         8089/8089 [==============================] - 802s 99ms/step - loss: 0.4477 - accuracy: 0.7900 - val_loss: 1.8194 - val_accurac
         y: 0.5615
         Epoch 3/5
         8088/8089 [============================>.] - ETA: 0s - loss: 0.4393 - accuracy: 0.7928
         Epoch 00003: val_loss did not improve from 1.52348
         8089/8089 [==============================] - 748s 93ms/step - loss: 0.4393 - accuracy: 0.7928 - val_loss: 1.8155 - val_accurac
         y: 0.6034
         Epoch 4/5
         8088/8089 [============================>.] - ETA: 0s - loss: 0.4363 - accuracy: 0.7945
         Epoch 00004: val_loss did not improve from 1.52348

         Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
         8089/8089 [==============================] - 816s 101ms/step - loss: 0.4363 - accuracy: 0.7946 - val_loss: 1.5500 - val_accurac
         y: 0.6073
         Epoch 5/5
         8088/8089 [============================>.] - ETA: 0s - loss: 0.4210 - accuracy: 0.8068
         Epoch 00005: val_loss did not improve from 1.52348
         8089/8089 [==============================] - 728s 90ms/step - loss: 0.4211 - accuracy: 0.8068 - val_loss: 1.8740 - val_accurac
         y: 0.5924

Out[10]: <tensorflow.python.keras.callbacks.History at 0x24a5f8880b8>

In [12]: acc_tr, loss_tr = model.evaluate_generator(train_data)
         print(acc_tr)
         print(loss_tr)

         WARNING:tensorflow:sample_weight modes were coerced from
           ...
             to
           ['...']
         1.1027720097889104
         0.6922863

In [13]: acc_vr, loss_vr = model.evaluate_generator(validation_data)
         print(acc_vr)
         print(loss_vr)

         WARNING:tensorflow:sample_weight modes were coerced from
           ...
             to
           ['...']
         1.8678017889520773
         0.5920386

In [14]: acc_test, loss_test = model.evaluate_generator(test_data)
         print(acc_tr)
         print(loss_tr)

         WARNING:tensorflow:sample_weight modes were coerced from
           ...
             to
           ['...']
         1.1027720097889104
         0.6922863

In [ ]:
```

Data_preparation.py:-



```
In [1]: import os
        import glob
        import shutil
        import random
        from tqdm import tqdm
```

```
In [2]: raw_data = '/home/delixus/Desktop/drowsiness_detection/mrlEyes_2018_01'
        for dirpath, dirname, filename in os.walk(raw_data):
            for file in tqdm([f for f in filename if f.endswith('.png')]):
                if file.split('_')[4] == '0':
                    path='/home/delixus/Desktop/drowsiness_detection/data/train/closed'
                    if not os.path.exists(path):
                        os.makedirs(path)
                    shutil.copy(src=dirpath + '/' + file, dst= path)
                elif file.split('_')[4] == '1':
                    path='/home/delixus/Desktop/drowsiness_detection/data/train/open'
                    if not os.path.exists(path):
                        os.makedirs(path)
                    shutil.copy(src=dirpath + '/' + file, dst= path)
```

```
0it [00:00, ?it/s]
100%|██████████| 681/681 [00:01<00:00, 371.65it/s]
100%|██████████| 522/522 [00:00<00:00, 1173.59it/s]
100%|██████████| 832/832 [00:00<00:00, 1119.53it/s]
100%|██████████| 1012/1012 [00:00<00:00, 1207.62it/s]
100%|██████████| 399/399 [00:00<00:00, 773.98it/s]
100%|██████████| 642/642 [00:00<00:00, 824.22it/s]
100%|██████████| 739/739 [00:00<00:00, 1036.91it/s]
100%|██████████| 1502/1502 [00:01<00:00, 1076.13it/s]
100%|██████████| 6162/6162 [00:08<00:00, 707.05it/s]
100%|██████████| 679/679 [00:00<00:00, 879.79it/s]
100%|██████████| 704/704 [00:00<00:00, 1362.01it/s]
100%|██████████| 3605/3605 [00:03<00:00, 1145.38it/s]
100%|██████████| 1132/1132 [00:01<00:00, 841.32it/s]
100%|██████████| 6193/6193 [00:05<00:00, 1111.98it/s]
100%|██████████| 1648/1648 [00:01<00:00, 1147.46it/s]
100%|██████████| 1889/1889 [00:08<00:00, 224.13it/s]
100%|██████████| 736/736 [00:00<00:00, 1064.67it/s]
100%|██████████| 10257/10257 [00:07<00:00, 1323.69it/s]
100%|██████████| 1393/1393 [00:00<00:00, 1394.60it/s]
100%|██████████| 387/387 [00:00<00:00, 1064.70it/s]
100%|██████████| 752/752 [00:00<00:00, 1269.94it/s]
100%|██████████| 1648/1648 [00:01<00:00, 1539.46it/s]
100%|██████████| 1738/1738 [00:04<00:00, 428.01it/s]
100%|██████████| 8884/8884 [00:10<00:00, 882.37it/s]
100%|██████████| 1069/1069 [00:01<00:00, 1009.81it/s]
100%|██████████| 6175/6175 [00:04<00:00, 1395.85it/s]
100%|██████████| 1246/1246 [00:01<00:00, 1020.66it/s]
100%|██████████| 1114/1114 [00:00<00:00, 1120.48it/s]
100%|██████████| 382/382 [00:00<00:00, 845.03it/s]
100%|██████████| 4410/4410 [00:04<00:00, 942.30it/s]
100%|██████████| 3242/3242 [00:03<00:00, 930.25it/s]
```

```
100%|██████████| 1114/1114 [00:00<00:00, 1120.48it/s]
100%|██████████| 382/382 [00:00<00:00, 845.03it/s]
100%|██████████| 4410/4410 [00:04<00:00, 942.30it/s]
100%|██████████| 3242/3242 [00:03<00:00, 930.25it/s]
100%|██████████| 8728/8728 [00:07<00:00, 1148.81it/s]
100%|██████████| 736/736 [00:00<00:00, 988.15it/s]
100%|██████████| 665/665 [00:00<00:00, 1435.48it/s]
100%|██████████| 987/987 [00:00<00:00, 1477.41it/s]
100%|██████████| 1384/1384 [00:01<00:00, 1250.77it/s]
100%|██████████| 624/624 [00:00<00:00, 1259.39it/s]
```

In [5]:
```python
def create_test_closed(source, destination, percent):
    '''
    divides closed eyes images into given percent and moves from
    source to destination.

    Arguments:
    source(path): path of source directory
    destination(path): path of destination directory
    percent(float): percent of data to be divided(range: 0 to 1)
    '''
    path, dirs, files_closed = next(os.walk(source))
    file_count_closed = len(files_closed)
    percentage = file_count_closed * percent
    to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

    for f in enumerate(to_move):
        if not os.path.exists(destination):
            os.makedirs(destination)
        shutil.move(f[1], destination)
    print(f'moved {int(percentage)} images to the destination successfully.')
```

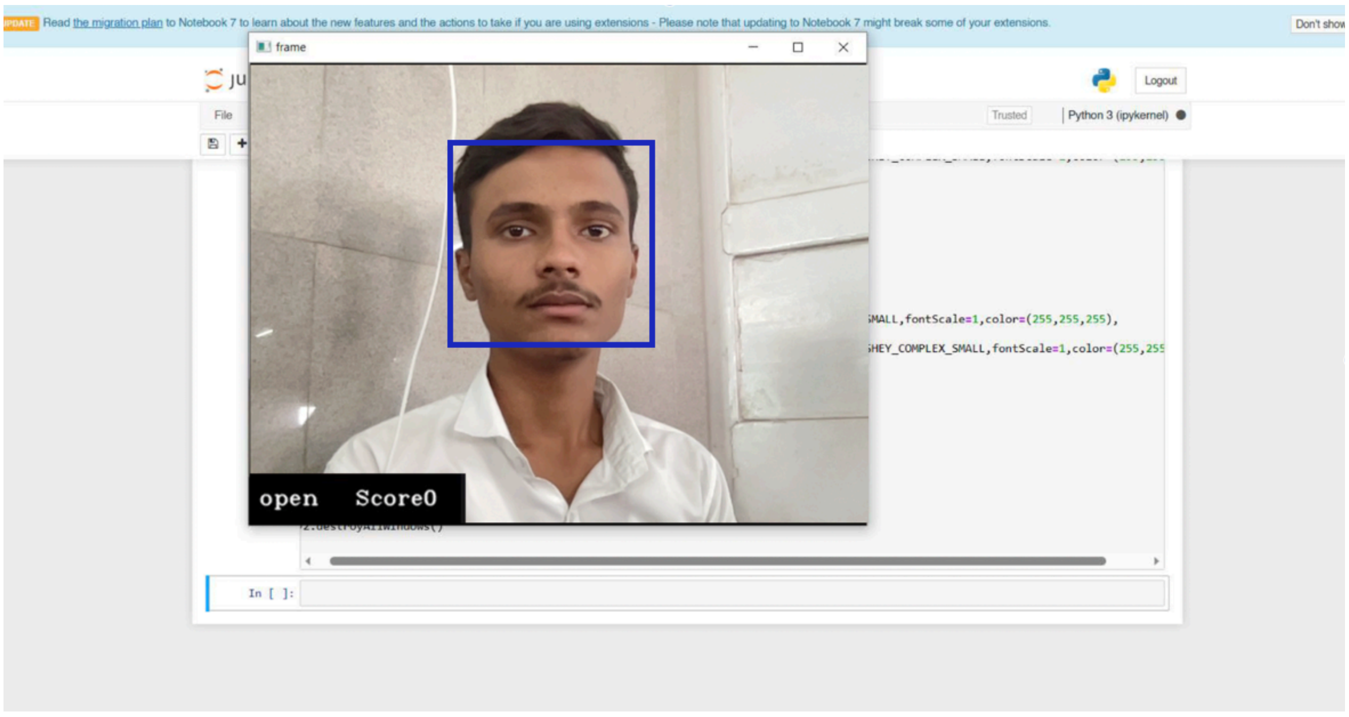
In [6]:
```python
def create_test_open(source, destination, percent):
    '''
    divides open eyes images into given percent and moves from
    source to destination.

    Arguments:
    source(path): path of source directory
    destination(path): path of destination directory
    percent(float): percent of data to be divided(range: 0 to 1)
    '''
    path, dirs, files_open = next(os.walk(source))
    file_count_open = len(files_open)
    percentage = file_count_open * percent
    to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

    for f in enumerate(to_move):
        if not os.path.exists(destination):
            os.makedirs(destination)
        shutil.move(f[1], destination)
    print(f'moved {int(percentage)} images to the destination successfully.')
```

In [7]:
```python
create_test_closed('/home/delixus/Desktop/drowsiness_detection/data/train/closed',
                   '/home/delixus/Desktop/drowsiness_detection/data/test/closed',
                   0.2)
```
```
moved 8389 images to the destination successfully.
```

In [8]:
```python
create_test_open('/home/delixus/Desktop/drowsiness_detection/data/train/open',
                 '/home/delixus/Desktop/drowsiness_detection/data/test/open',
                 0.2)
```
```
moved 8590 images to the destination successfully.
```

Results:-

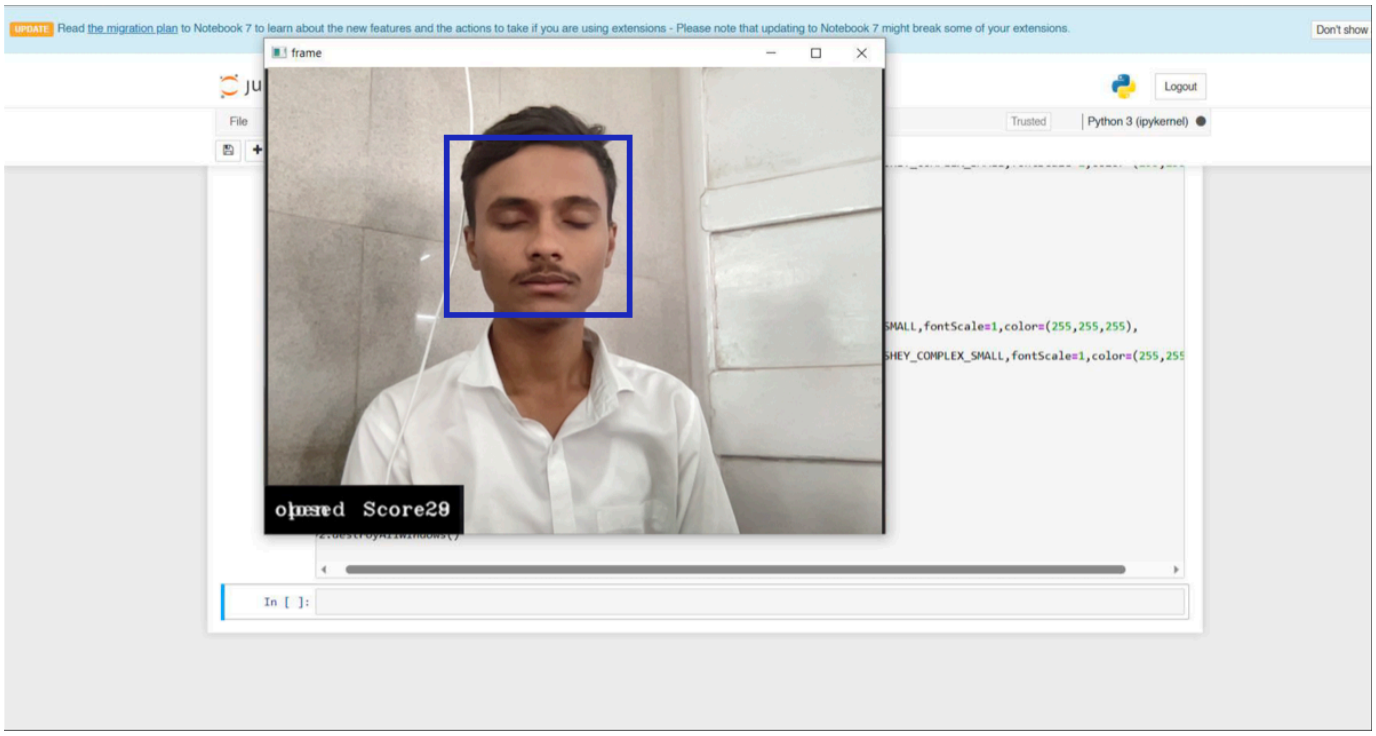
1. **When the eyes are open.**



UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

```
SMALL,fontScale=1,color=(255,255,255),
SHEY_COMPLEX_SMALL,fontScale=1,color=(255,255
```

2. **When the eyes are closed, and the score is above 20 then it will start giving alert.**



UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

```
SMALL,fontScale=1,color=(255,255,255),
SHEY_COMPLEX_SMALL,fontScale=1,color=(255,255
```

3.5 Analysis

This project is built using python programming language and a highly evolving field which is computer vision, it allows computers to watch or give vision capability .Overall, the proposed system has the potential to improve road safety by detecting drowsiness in drivers in real-time. However, it would need to be designed and implemented carefully to ensure accuracy, reliability, usability, integration, and privacy. The system's cost would also need to be considered, especially for commercial applications.

3.6 Conclusion And Future Work

Conclusion:-

In conclusion, driver drowsiness is a serious problem that can lead to accidents on the road. A driver drowsiness detection system using OpenCV can be an effective solution to detect and alert drivers when they become drowsy. OpenCV provides a wide range of tools and pre-trained models for facial landmark detection, which can be used to track key features on the driver's face, such as the eyes, eyebrows, and mouth. By measuring the eye aspect ratio (EAR) and detecting yawns, the system can accurately detect when the driver is becoming drowsy and trigger an alert to prompt the driver to take a break.Moreover, the implementation of such a system can be customized to suit the specific needs of different drivers and driving conditions. It can be easily integrated with existing vehicle systems and can provide valuable data on the driver's drowsiness level and driving behavior.Overall, a driver drowsiness detection system using OpenCV can help to reduce the risk of accidents caused by driver drowsiness and improve road safety. The implementation of such a system can be a cost-effective and efficient way to detect driver drowsiness and enhance overall safety on the roads.

FutureWork:-

The model can be improved incrementally by using other parameters like blink rate, yawning, state of the car, etc. If all these parameters are used it can improve the accuracy by a lot. We plan to further work on the project by adding a sensor to track the heart rate in order to prevent accidents caused due to sudden heart attacks to drivers. Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when the user is asleep and stop the video accordingly. It can also be used in applications that prevent users from sleeping.

4.REFERENCES

4.1 Published Paper /Camera Ready Paper/ Business pitch/proof of concept

[1] Kyong Hee Lee, Whui Kim, Hyun Kyun Choi, Byung Tae Jan.“A Study on Feature Extraction Methods Used to Estimate a Driver’s Level of Drowsiness”, IEEE, February 2019.

[2] Srinivas Batchu, S Praveen Kumar, “Driver DrowsinessDetection to Reduce the Major Road Accidents in Automotive Vehicles”, IRJET, Volume 02 Issue 01, April 2015.

[3] Hardeep Singh, J S Bhatia and Jasbir Kaur, “Eye Tracking based Driver Fatigue Monitoring and Warning System”, IEEE, January 2011.

[4] Fouzia, Roopalakshmi R, Jayantkumar A Rathod, Ashwitha S,Supriya K, “Driver Drowsiness Detection System Based on Visual Features.” , IEEE, April 2018.

[5] Varsha E Dahiphale, Satyanarayana R, “A Real-Time Computer Vision System for Continuous Face Detection and Tracking”, IJCA, Volume 122 Number 18, July 2015.

[6] SaeidFazli, Parisa Esfehani, “Tracking Eye State for Fatigue Detection”, ICACEE, November 2012. Gao Zhenhai, Le DinhDat, Hu Hongyu, Yu Ziwen, Wu Xinyu, “Driver Drowsiness Detection Based on Time Series Analysis of Steering Wheel Angular Velocity”, IEEE, January 2017.