A PROJECT REPORT

on

DRIVER DROWSINESS DETECTION SYSTEM USING MACHINE LEARNING

Submitted to

G.L. BAJAJ INSTITUTE OF TECHNOLOGY AND MANAGEMENT

In Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER SCIENCE & ENGINEERING

BY

JAY MISHRA	ROLL NO:1901920100134
HIMANSHU JAIN	ROLL NO:1901920100123
HARSH BHAGWANI	ROLL NO:1901920100112

UNDER THE GUIDANCE OF

Mr. Sachin Chawla

G.L. BAJAJ INSTITUTE OF TECHNOLOGY AND MANAGEMENT
GREATER NOIDA-201310

January 2022

G.L. BAJAJ INSTITUTE OF TECHNOLOGY AND MANAGEMENT

Greater Noida, U.P.- 201310

CERTIFICATE

This is certify that the project entitled

Driver Drowsiness Detection System

Using Machine Learning

Submitted by

JAY MISHRA	ROLL NUMBER:19019210100134
HIMANSHU JAIN	ROLL NUMBER :1901920100123
HARSH BHAGWANI	ROLL NUMBER:1901920100112

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Sci- ence & Engineering) at G.L. Bajaj Institute Of Technology and Management,,Greater Noida. This work is done during year 2020-2021, under our guidance.

Date:

Mr.Sachin Chawla
Project Guide

ACKNOWLEDGEMENT

We are profoundly grateful to Mr. Sachin Chawla for his expert guidance and continuousencouragement throughout to see that this project rights its target since its commencement to its completion.

JAY MISHRA(1901920100134)
HIMANSHU JAIN(1901920100123)
HARSH BHAGWANI(1901920100112)

ABSTRACT

This document is a review report on the research conducted and the project made in the field of computer engineering to develop a system for driver drowsiness detection to prevent accidents from happening because of driver fatigue and sleepiness. The report proposed the results and solutions on the limited implementation of the various techniques that are introduced in the project. Whereas the implementation of the project give the real world idea of how the system works and what changes can be done in order to improve the utility of the overall system.

Furthermore, the paper states the overview of the observations made by the authors in order to help further optimization in the mentioned field to achieve the utility at a better efficiency for a safer road.

Keywords—Driver drowsiness; eye detection; yawn detection; blink pattern; fatigue

Contents

List of Figures

Chapter 1

Introduction

1.1 PURPOSE

1.1.1 HUMAN PSYCHOLOGY WITH CURRENT TECHNOLOGY

Humans have always invented machines and devised techniques to ease and protect their lives, for mundane activities like traveling to work, or for more interesting purposes like aircraft travel. With the advancement in technology, modes of transportation kept on advancing and our dependency on it started increasing exponentially. It has greatly affected our lives as we know it. Now, we can travel to places at a pace that even our grandparents wouldn't have thought possible. In modern times, almost everyone in this world uses some sort of transportation every day. Some people are rich enough to have their own vehicles while others use public transportation. However, there are some rules and codes of conduct for those who drive irrespective of their social status. One of them is staying alert and active while driving.

Neglecting our duties towards safer travel has enabled hundreds of thousands of tragedies to get associated with this wonderful invention every year. It may seem like a trivial thing to most folks but following rules and regulations on the road is of utmost importance. While on road, an automobile wields the most power and in irresponsible hands, it can be destructive and sometimes, that carelessness can harm lives even of the people on the road. One kind of carelessness is not admitting when we are too tired to drive. In order to monitor and prevent a destructive outcome from such negligence, many researchers have written research papers on driver drowsiness detection systems. But at times, some of the points and observations made by the system are not accurate enough. Hence, to provide data and another perspective on the problem at hand, in order to improve their implementations and to further optimize the solution, this project has been done.

1.1.2 FACTS & STATISTICS

Our current statistics reveal that just in 2015 in India alone, 148,707 people died due to car related accidents. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem as its implementation is very difficult and costly.

1.2 INTENDED AUDIENCE

The intended audience for this document are the development team, the project evaluation jury, and other tech-savvy enthusiasts who wish to further work on the project.

1.3 PRODUCT SCOPE

There are many products out there that provide the measure of fatigue level in the drivers which are implemented in many vehicles. The driver drowsiness detection system provides the similar functionality but with better results and additional benefits. Also, it alerts the user on reaching a certain saturation point of the drowsiness measure.

1.4 PROBLEM DEFINITION

Fatigue is a safety problem that has not yet been deeply tackled by any country in the world mainly because of its nature. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative.

Chapter 2

Literature Survey

2.1 SYSTEM REVIEW

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need of such application and felt that there is a decent extent of progress in this field too.

2.2 TECHNOLOGY USED

a) PYTHON

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace.Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

b) COMMAND PROMPT

Command Prompt is a command line interpreter application available in most Windows operating systems. It's used to execute entered commands. Most of those commands automate tasks via scripts and batch files, perform advanced administrative functions, and troubleshoot or solve certain kinds of Windows issues.

c) IMAGE PROCESSING

In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images.

d) MACHINE LEARNING

Machine learning is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions .

Chapter 3

Software Requirements Specification

3.1 Python:

- Python 3

3.2 Libraries

- OpenCV
- Tensorflow
- Keras
- Pygame
- Numpy,etc.

3.3 Operating System

- Windows or Ubuntu

Hardware Requirements Specification

I. Laptop with basic hardware.
II. Webcam

Chapter 4

Requirement Analysis

4.1 Python: Python is the basis of the program that we wrote. It utilizes many of the python libraries.

4.2 Libraries:

- Numpy: Pre-requisite for Dlib
- Tensorflow:Performs backend operations.
- Pygame: Used for sounding the alarm
- Keras:To build our classification model.
- Imutils: Convenient functions written for Opencv.
- Opencv: Used to get the video stream from the webcam, etc.

4.3 OS: Program is tested on Windows 10 build 1903 and PopOS 19.04

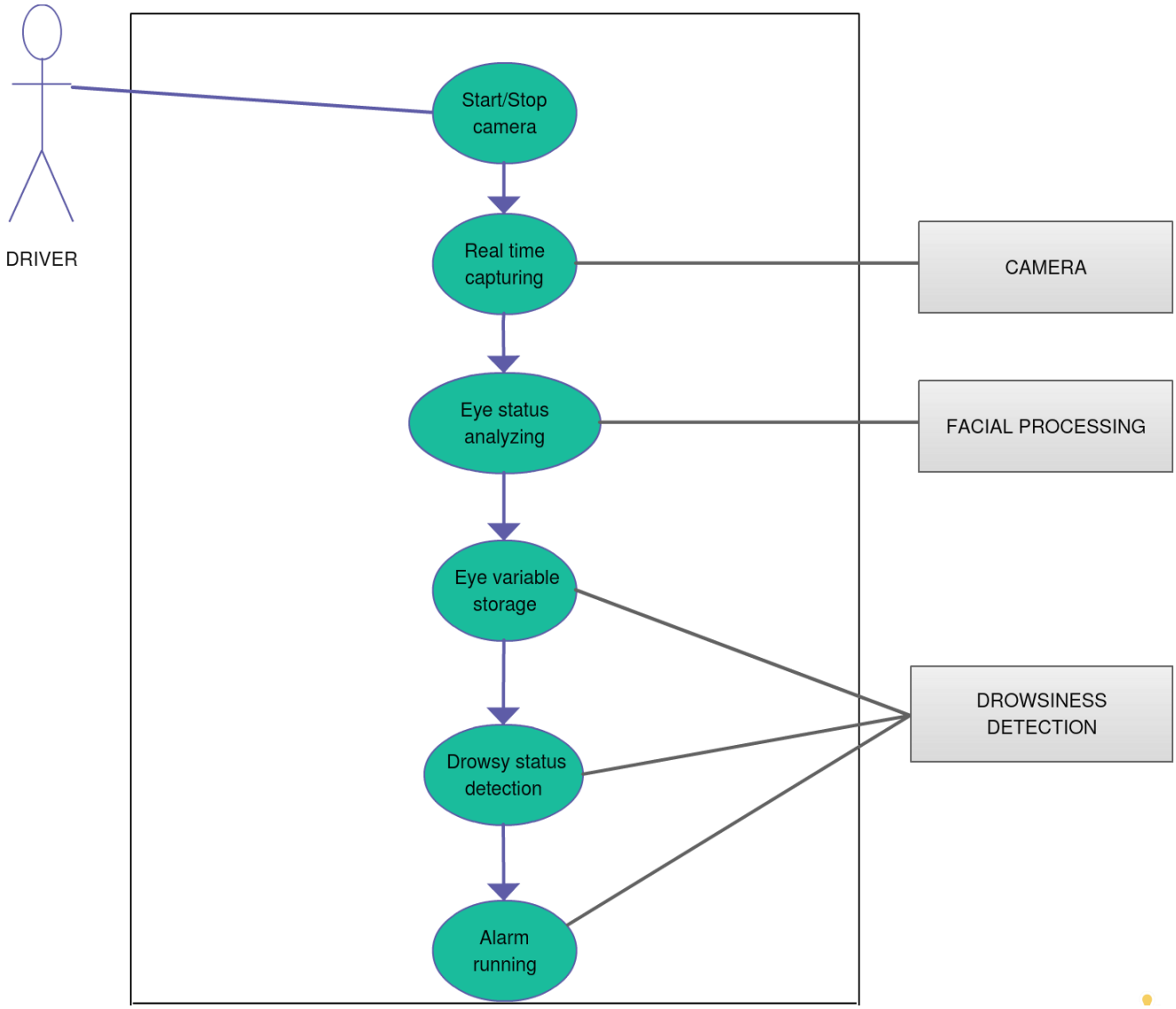
4.3 Laptop: Used to run our code.

4.4 Webcam: Used to get the video feed.

Chapter 5

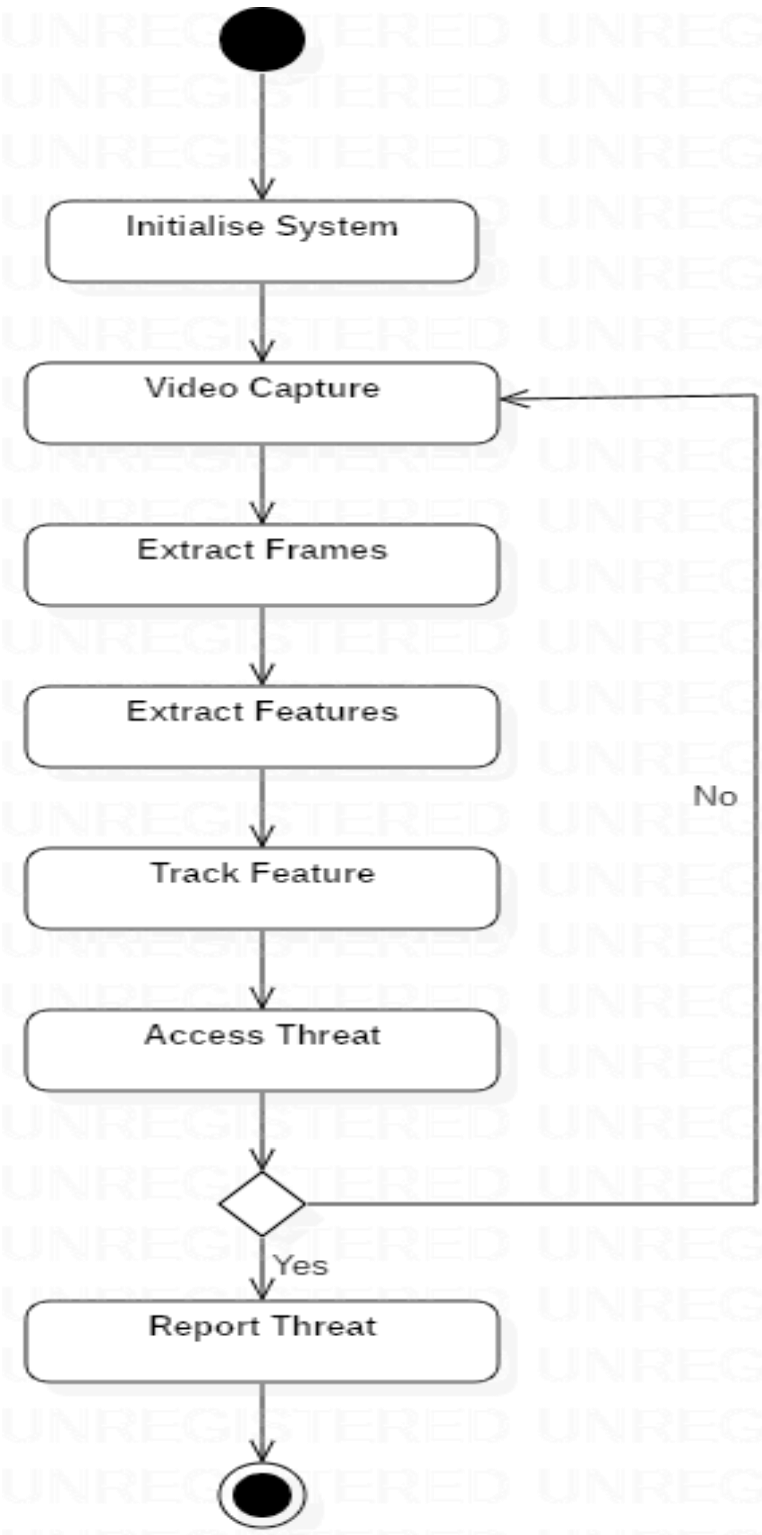
System Design

5.1 USE CASE DIAGRAM



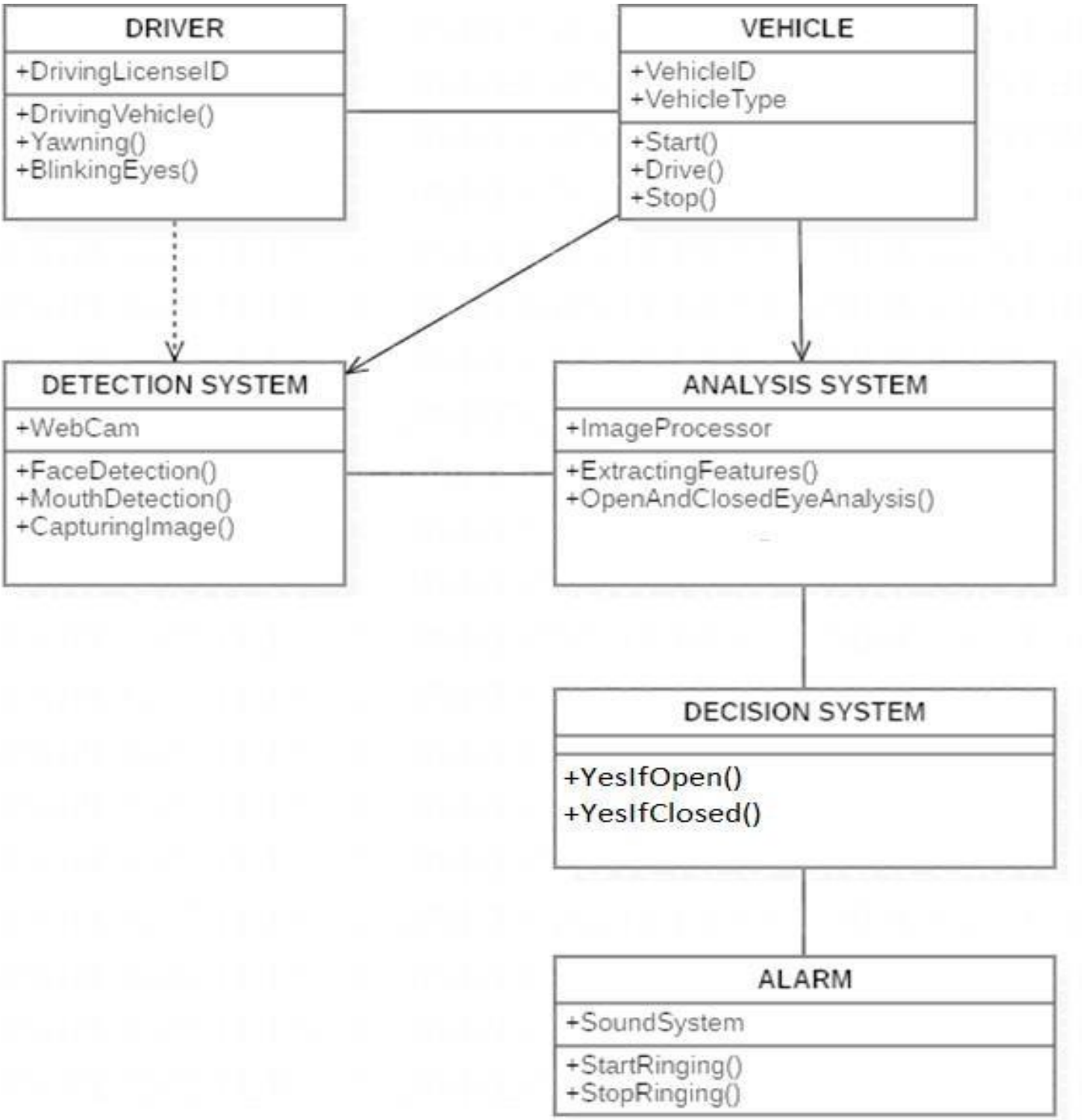
5.1 BASIC IDEA OF IMPLEMENTATION OF PROJECT IN REAL TIME

5.2 ACTIVITY DIAGRAM



5.2 FLOW CHART FOR DRIVER DROWSINESS DETECTION SYSTEM

CLASS DIAGRAM



5.3 FUNCTIONAL IMPLEMENTATION OF PROJECT

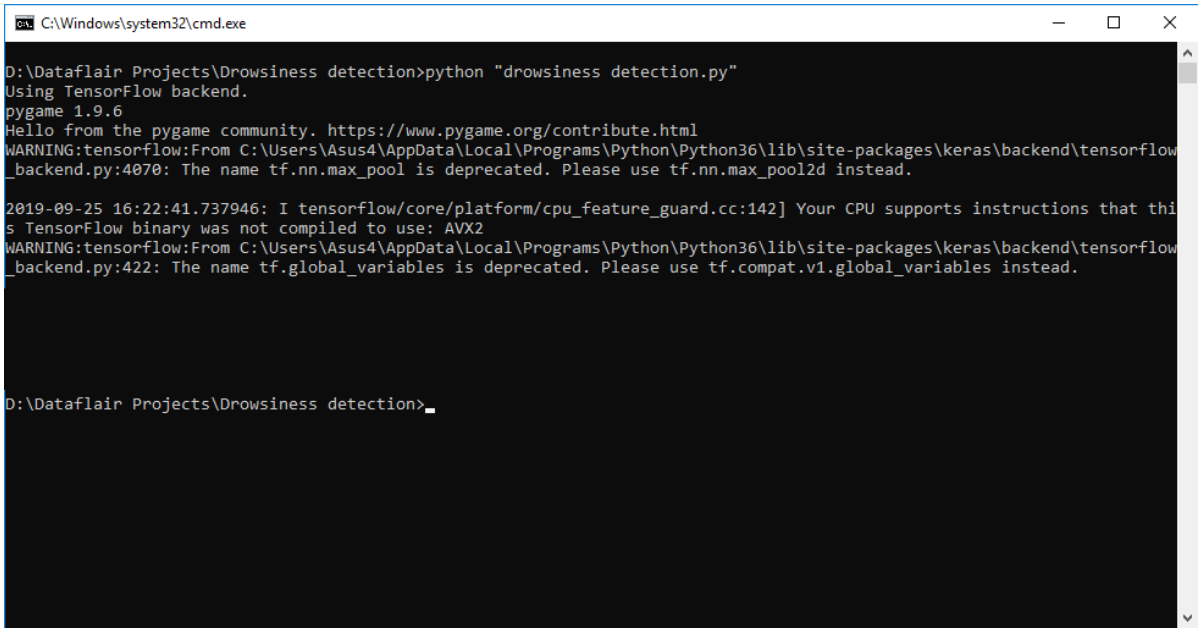
Chapter 6

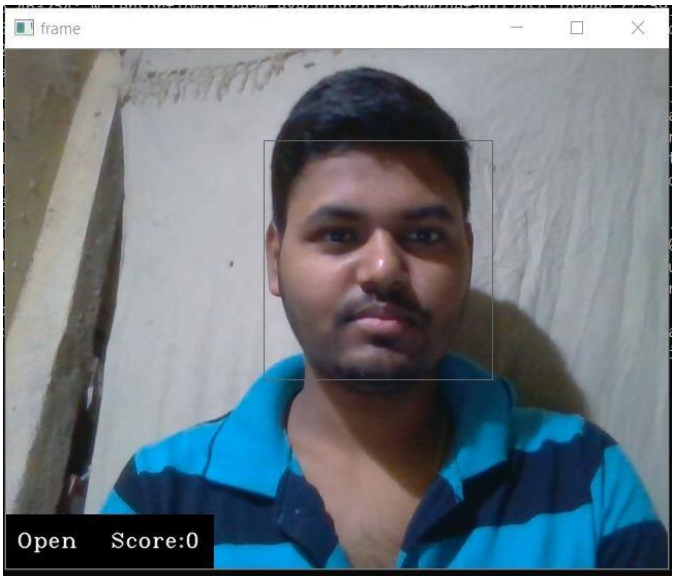
System Testing

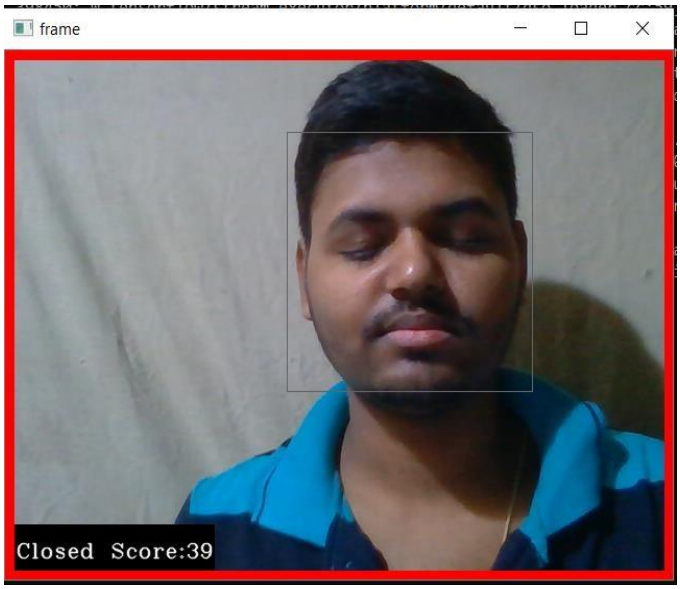
6.1 Test Cases and Test Results

Test ID	Test Case Title	Test Condition	System Behavior	Expected Result
T01	NSGY	Straight Face, Good Light	Open	Open
T02	YTGN	Straight Face, Good Light,	Closed	Closed

Example Screenshot:







Note: Testing is performed manually.

Chapter 7

Project Planning

7.1 SYSTEM MODEL

The framework is created utilizing the incremental model. The center model of the framework is first created and afterwards augmented in this way in the wake of testing at each turn. The underlying undertaking skeleton was refined into expanding levels of ability.

At the following incremental level, it might incorporate new execution backing and improvement.

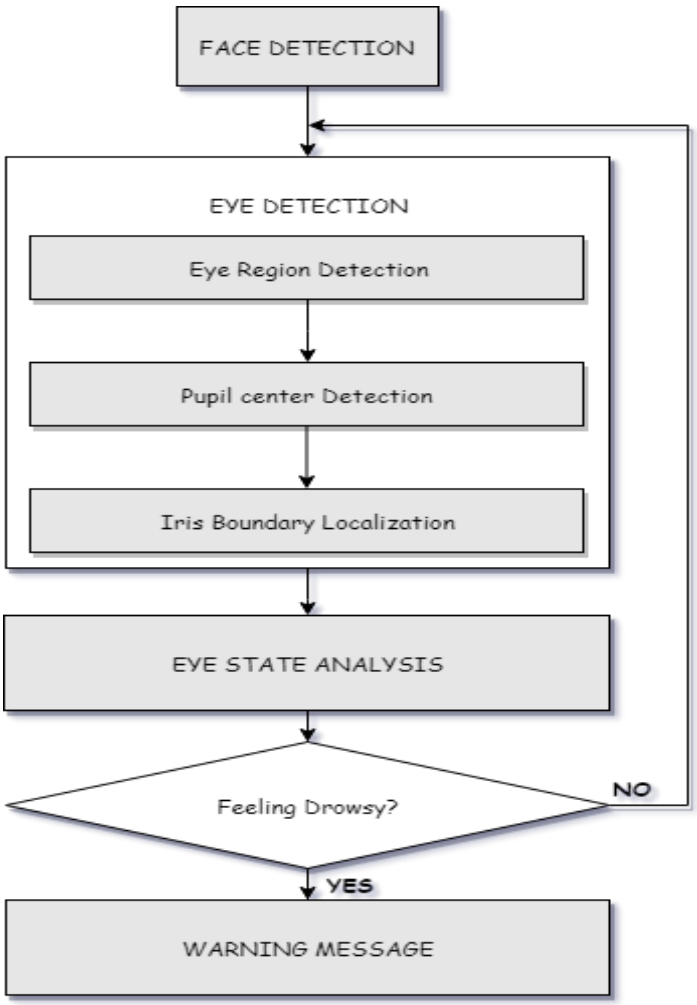


Figure 4: Block diagram

Chapter 8

Driver Drowsiness Detection System

8.1 About the Python Project

In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed'. The approach we will be using for this Python project is as follows :

Step 1 – Take image as input from a camera.
Step 2 – Detect the face in the image and create a Region of Interest (ROI).
Step 3 – Detect the eyes from ROI and feed it to the classifier.
Step 4 – Classifier will categorize whether eyes are open or closed.
Step 5 – Calculate score to check whether the person is drowsy.

8.2 The Dataset

The dataset used for this model is created by us. To create the dataset, we wrote a script that captures eyes from a camera and stores in our local disk. We separated them into their respective labels 'Open' or 'Closed'. The data was manually cleaned by removing the unwanted images which were not necessary for building the model. The data comprises around 7000 images of people's eyes under different lighting conditions. After training the model on our dataset, we have attached the final weights and model architecture file "models/cnnCat2.h5".

Now, you can use this model to classify if a person's eye is open or closed.

8.3 The Model Architecture

The model we used is built with Keras using Convolutional Neural Networks (CNN). A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple numbers of layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

The CNN model architecture consists of the following layers:

- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 32 nodes, kernel size 3
- Convolutional layer; 64 nodes, kernel size 3
- Fully connected layer; 128 nodes

The final layer is also a fully connected layer with 2 nodes. In all the layers, a Relu activation function is used except the output layer in which we used Softmax.

8.4 Prerequisites

The requirement for this Python project is a webcam through which we will capture images. You need to have Python (3.6 version recommended) installed on your system, then using pip, you can install the necessary packages.

OpenCV – pip install opencv-python (face and eye detection).
TensorFlow – pip install tensorflow (keras uses TensorFlow as backend).
Keras – pip install keras (to build our classification model).
Pygame – pip install pygame (to play alarm sound).

The “haar cascade files” folder consists of the xml files that are needed to detect objects from the image. In our case, we are detecting the face and eyes of the person.
The models folder contains our model file “cnnCat2.h5” which was trained on convolutional neural networks. We have an audio clip “alarm.wav” which is played when the person is feeling drowsy.
“Model.py” file contains the program through which we built our classification model by training on our dataset. You could see the implementation of convolutional neural network in this file.
“Drowsiness detection.py” is the main file of our project. To start the detection procedure, we have to run this file.

8.5 Algorithm

Let’s now understand how our algorithm works step by step.

Step 1 – Take Image as Input from a Camera

With a webcam, we will take images as input. So to access the webcam, we made an infinite loop that will capture each frame. We use the method provided by OpenCV, cv2.VideoCapture(0) to access the camera and set the capture object (cap). cap.read() will read each frame and we store the image in a frame variable.

Step 2 – Detect Face in the Image and Create a Region of Interest (ROI)

To detect the face in the image, we need to first convert the image into grayscale as the OpenCV algorithm for object detection takes gray images in the input. We don’t need color information to detect the objects. We will be using haar cascade classifier to detect faces. This line is used to set our classifier face = cv2.CascadeClassifier(‘ path to our haar cascade xml file’). Then we perform the detection using faces = face.detectMultiScale(gray). It returns an array of detections with x,y coordinates, and height, the width of the boundary box of the object. Now we can iterate over the faces and draw boundary boxes for each face.

```
for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x,y), (x+w, y+h), (100,100,100), 1 )
```

Step 3 – Detect the eyes from ROI and feed it to the classifier

The same procedure to detect faces is used to detect eyes. First, we set the cascade classifier for eyes in leye and reye respectively then detect the eyes using left_eye = leye.detectMultiScale(gray). Now we need to extract only the eyes data from the full image. This can be achieved by extracting the boundary box of the eye and then we can pull out the eye image from the frame with this code.

```
l_eye = frame[ y : y+h, x : x+w ]
```

l_eye only contains the image data of the eye. This will be fed into our CNN classifier which will predict if eyes are open or closed. Similarly, we will be extracting the right eye into r_eye.

Step 4 – Classifier will Categorize whether Eyes are Open or Closed

We are using CNN classifier for predicting the eye status. To feed our image into the model, we need to perform certain operations because the model needs the correct dimensions to start with. First, we convert the color image into grayscale using r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY). Then, we resize the image to 24*24 pixels as our model was trained on 24*24 pixel images cv2.resize(r_eye, (24,24)). We normalize our data for better convergence r_eye = r_eye/255 (All values will be between 0-1). Expand the

dimensions to feed into our classifier. We loaded our model using model = load_model('models/cnnCat2.h5') . Now we predict each eye with our model

lpred = model.predict_classes(l_eye). If the value of lpred[0] = 1, it states that eyes are open, if value of lpred[0] = 0 then, it states that eyes are closed.

Step 5 – Calculate Score to Check whether Person is Drowsy

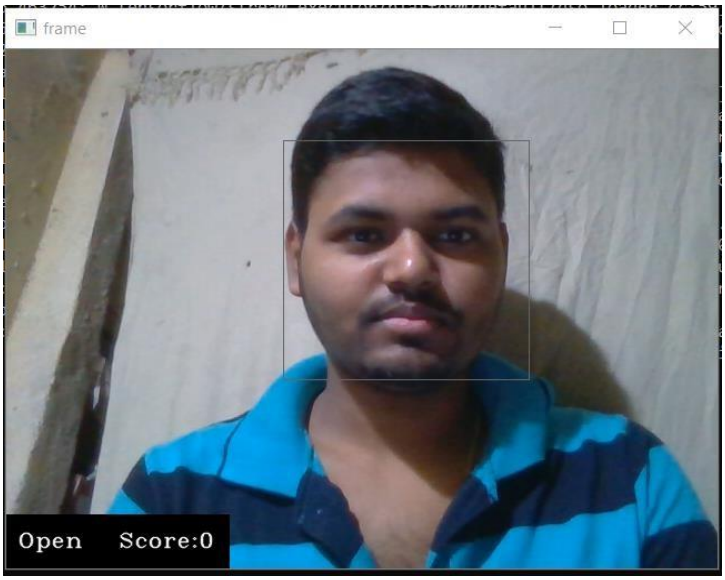
The score is basically a value we will use to determine how long the person has closed his eyes. So if both eyes are closed, we will keep on increasing score and when eyes are open, we decrease the score. We are drawing the result on the screen using cv2.putText() function which will display real time status of the person.

cv2.putText(frame, "Open", (10, height-20), font, 1, (255,255,255), 1, cv2.LINE_AA)
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 we beep the alarm using sound.play()

Chapter 9

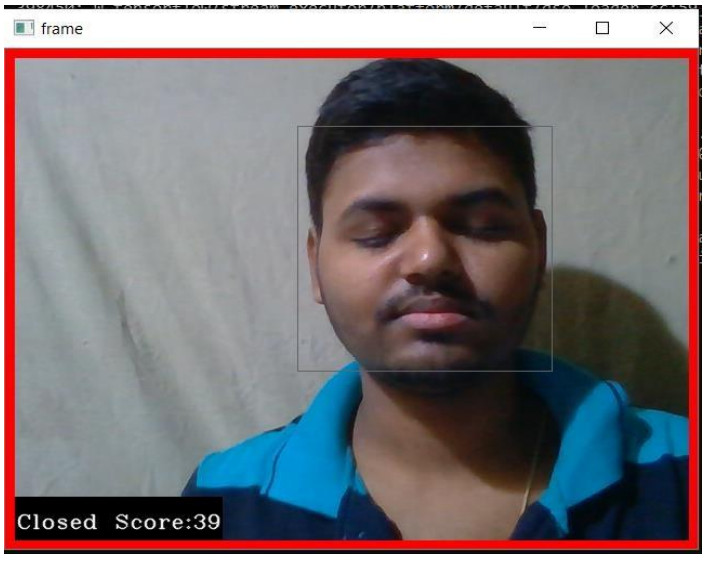
Screenshots of Project

9.1 Awaken Person:



Open eyes state

9.2 Sleeping Person:



Close eyes state

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

It completely meets the objectives and requirements of the system. The framework has achieved an unfaltering state where all the bugs have been disposed of. The framework cognizant clients who are familiar with the framework and comprehend it's focal points and the fact that it takes care of the issue of stressing out for individuals having fatigue-related issues to inform them about the drowsiness level while driving.

10.2 Future Scope

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 application that prevents user from sleeping.

References

[1] COMPUTATIONALLY EFFICIENT FACE DETECTION; B. SCHLKOPF-A. BLAKE, S. ROMDHANI, AND P. TORR.

[2] USE OF THE HOUGH TRANSFORMATION TO DETECT LINES AND CURVES IN PICTURE; R. DUDA AND P. E. HART.

[3] JAIN, "FACE DETECTION IN COLOR IMAGES; R. L. HSU, M. ABDEL-MOTTALEB, AND A. K. JAIN.

[4] OPEN/CLOSED EYE ANALYSIS FOR DROWSINESS DETECTION; P.R. TABRIZI AND R. A. ZOROOFI.

[5] http://ncrb.gov.in/StatPublications/ADSI/ADSI2015/chapter1A%20traffic%20accidents.pdf

[6] http://www.jotr.in/text.asp?2013/6/1/1/118718

[7] http://dlib.net/face_landmark_detection_ex.cpp.html

Appendix

"drowsiness detection.py"

```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time


mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')



lbl=['Close','Open']

model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]

while(True):
 ret, frame = cap.read()
 height,width = frame.shape[:2]

  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
  left_eye = leye.detectMultiScale(gray)
right_eye =  reye.detectMultiScale(gray)

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

 for (x,y,w,h) in faces:
 cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

 for (x,y,w,h) in right_eye:
  r_eye=frame[y:y+h,x:x+w]
  count=count+1
```

```
r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
  r_eye = cv2.resize(r_eye,(24,24))



r_eye= r_eye/255
r_eye=  r_eye.reshape(24,24,-1)
r_eye = np.expand_dims(r_eye,axis=0)
rpred = model.predict_classes(r_eye)
if(rpred[0]==1):
lbl='Open'
if(rpred[0]==0):
lbl='Closed'

break

for (x,y,w,h) in left_eye:
l_eye=frame[y:y+h,x:x+w]
count=count+1
 l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
 l_eye = cv2.resize(l_eye,(24,24))
l_eye= l_eye/255
l_eye=l_eye.reshape(24,24,-1)
l_eye = np.expand_dims(l_eye,axis=0)
lpred = model.predict_classes(l_eye)
if(lpred[0]==1):
lbl='Open'
if(lpred[0]==0):
lbl='Closed'

break

if(rpred[0]==0 and lpred[0]==0):
score=score+1
cv2.putText(frame,"Closed",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)

# if(rpred[0]==1 or lpred[0]==1):
else:
score=score-1
cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)


if(score<0):
score=0
cv2.putText(frame,'Score:'+str(score),(100,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)

if(score>15):
 #person is feeling sleepy so we beep the alarm
cv2.imwrite(os.path.join(path,'image.jpg'),frame)
try:
```

```
        sound.play()



    except:  # isplaying = False
    pass
if(thicc<16):
    thicc= thicc+2
else:
    thicc=thicc-2



      if(thicc<2):
        thicc=2
      cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
      cv2.imshow('frame',frame)

     if cv2.waitKey(1) & 0xFF == ord('q'):
      break
    cap.release()
    cv2.destroyAllWindows()
```

"model.py"

```
import os
from keras.preprocessing import image
import matplotlib.pyplot as plt
import numpy as np
from keras.utils.np_utils import to_categorical
import random,shutil
from keras.models import Sequential
from keras.layers import Dropout,Conv2D,Flatten,Dense, MaxPooling2D, BatchNormalization
from keras.models import load_model


def generator(dir, gen=image.ImageDataGenerator(rescale=1./255),
shuffle=True,batch_size=1,target_size=(24,24),class_mode='categorical' ):

    return
gen.flow_from_directory(dir,batch_size=batch_size,shuffle=shuffle,color_mode='grayscale',class_m
ode=class_mode,target_size=target_size)

BS= 32
TS=(24,24)
train_batch= generator('data/train',shuffle=True, batch_size=BS,target_size=TS)
valid_batch= generator('data/valid',shuffle=True, batch_size=BS,target_size=TS)
SPE= len(train_batch.classes)//BS
VS = len(valid_batch.classes)//BS
print(SPE,VS)


# img,labels= next(train_batch)
# print(img.shape)

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(24,24,1)),
    MaxPooling2D(pool_size=(1,1)),
    Conv2D(32,(3,3),activation='relu'),
    MaxPooling2D(pool_size=(1,1)),
#32 convolution filters used each of size 3x3
#again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1,1)),

#64 convolution filters used each of size 3x3
#choose the best features via pooling

#randomly turn neurons on and off to improve convergence
    Dropout(0.25),
#flatten since too many dimensions, we only want a classification output
    Flatten(),
#fully connected to get all relevant data
    Dense(128, activation='relu'),
#one more dropout for convergence' sake :)
    Dropout(0.5),
```

```
#output a softmax to squash the matrix into output probabilities
   Dense(2, activation='softmax')
])

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

model.fit_generator(train_batch,        validation_data=valid_batch,epochs=15,steps_per_epoch=SPE
,validation_steps=VS)

model.save('models/cnnCat2.h5', overwrite=True)
```