

A

Semester Project-III

Report

On

**“DRIVER DROWSINESS
DETECTION”**

By

1. Neha Dhananjay Patil [21107033]
2. Ruchita Lalit Mahajan [21107034]
3. Perizad Atik Shaikh [21107035]
4. Rajshri Ananda Bedse [21107045]



R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

**Department of Artificial Intelligence and
Machine Learning**

The Shirpur Education Society's
R. C. Patel Institute of Technology, Shirpur - 425405.

[2023-24]

A
Semester Project-III Report
On

“DRIVER DROWSINESS DETECTION”

In partial fulfillment of requirements for the degree of
Bachelor of Technology

In
Department of Artificial Intelligence and Machine Learning

Submitted By

1. Neha Dhananjay Patil (211107033)
2. Ruchita Lalit Mahajan (211107034)
3. Perizad Atik Shaikh (211107035)
4. Rajshri Ananda Bedse (211107045)

Under the Guidance of
Prof. Shailendra. M. Pardeshi.



R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

The Shirpur Education Society's
R. C. Patel Institute of Technology, Shirpur - 425405.

Department of Artificial Intelligence and Machine
Learning

[2023-24]



R. C. PATEL
INSTITUTE OF TECHNOLOGY
An Autonomous Institute

The Shirpur Education Society's

R. C. Patel Institute of Technology
Shirpur, Dist. Dhule (M.S.)

Department of Artificial Intelligence and
Machine Learning

CERTIFICATE

This is to certify that the Semester Project-III entitled “**Driver Drowsiness Detection**” has been carried out by team:

- 1. Neha Dhananjay Patil**
- 2. Ruchita Lalit Mahajan**
- 3. Perizad Atik Shaikh**
- 4. Rajshri Ananda Bedse**

under the guidance of **Prof. Shailendra.M.Pardeshi.** in partial fulfillment of the requirement for the degree of Bachelor of Technology in Department of Artificial Intelligence and Machine Learning (Semester-V) of Dr. Babasaheb Ambedkar Technological University, Lonere during the academic year 2023-24.

Date:

Place: Shirpur

Guide

Prof. Shailendra. M. Pardeshi.

Semester Project-III Coordinator

Prof. Priyanka Lanjewar

H.O.D.

Prof. Dr. R. B. Wagh

Director

Prof. Dr. J. B. Patil

ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our Project Guide **Prof. Shailendra. M. Pardeshi.** for their valuable inputs, adorable guidance, encouragement, whole-hearted co-operation and constructive criticism throughout the duration of our project.

We deeply express our sincere thanks to our Head of Department **Dr. R. B. Wagh** sir for encouraging and allowing us to present the project on the topic “**Driver Drowsiness Detection** “ at our department premises for the partial fulfilment of the requirements leading to the award of B-Tech degree.

We are also grateful to honourable Principal **Dr. J. B. Patil**, for his kind support and guidance. Last but not the least I express my sincere thanks to all of my friends who have patiently extended all sorts of help for accomplishing this undertaking, Thank for your kind support.

Project Team:

1. Neha Dhananjay Patil
2. Ruchita Lalit Mahajan
3. Perizad Atik Shaikh
4. Rajshri Ananda Bedse

Chapter No.	Topic	Page No.
	ABSTRACT	03
1.	INTRODUCTION	04
1.1	Importance of Driver Drowsiness Detection	05
1.2	What is Driver Drowsiness Detection	06
1.3	Techniques Involved	08
1.4	Architecture	09
1.5	Need for Drowsiness Detection	10
2.	RELATED CONCEPTS	11
2.1	OpenCv	11
2.2	Dlib	12
2.3	Numpy	13
2.4	TensorsFlow	13
2.5	Keras	14
2.6	Pygame	15
3.	SOFTWARE AND HARDWARE	16
3.1	Hardware Requirements	16
3.2	software Requirements	16
4.	IMPLEMENTATION	17
4.1	Implementation Details	17
4.2	Algorithm used	17
4.3	Flowchart	18
4.4	Steps for Drowsiness Detection	18
4.5	Code	22
4.6	Output	28
	CONCLUSION	
	REFERNCES/BIBLIOGRAPHY	

Fig. No	Figure	Page No.
1.2	Figure 1 Image of driver drowsiness	07
1.3	Figure 2 Eye tracking and plot generation	09
1.3	Figure 3 Eye tracking	09
1.5	Figure 4 Need for drowsiness detection	10
4.1	Figure 5: Folder of Drowsiness files	17
4.3	Figure 6 Flowchart	18
4.5	Figure 7 Output1	28
4.5	Figure 8 Output2	28
4.5	Figure 9 Output3	29
4.5	Figure 10 Output4	29
4.5	Figure 11 Output5	30
4.5	Figure 12 Output6	30
4.5	Figure 13 Output7	31
4.5	Figure 14 Output8	31

ABSTRACT

On an average 1200 road accidents record daily in India out of which 400 leads to direct death and rest gets affected badly. The major reason of these accidents is drowsiness caused by both sleep and alcohol. Due to driving for long time or intoxication, drivers might feel sleepy which the biggest distraction for them while driving is. This distraction might cost death of driver and other passengers in the vehicle and at the same time it also causes death of people in the other vehicles and pedestrians too. This mistake of one person on road would take their own life and also takes lives of other and put respective families in sorrow and tough situations.

To prevent such accidents propose a system which alerts the driver if he/she feels drowsy. To accomplish this implement the solution using computer-vision based machine learning model. The driver's face is detected by face recognition algorithm continuously using a camera and the face of the driver is captured. The face of the driver is given as input to a classification algorithm which is trained with a data set of images of drowsy and non-drowsy faces. The algorithm uses landmark detection to classify the face as drowsy or not drowsy. If the driver's face is drowsy, a voice alert is generated by the system. This alert can make the driver aware that he/she is feeling drowsy and the necessary actions can then be taken by the driver. This can be used in any vehicle on the road to ensure safety of the people who are travelling and prevent accidents which are caused due to the drowsiness of the driver.

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

Driver drowsiness detection is a car safety technology which prevents accidents when the driver is getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads, driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects related, up to 50% on certain roads. Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related crashes. The development of technologies for detecting or preventing drowsiness at the wheel is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its effect. Drowsiness detection is one of those common problem needed to be solved to prevent road accidents. In recent time's automobile fatigue connected crashes have very enlarged.

1.1 Importance of Driver Drowsiness Detection

Driver drowsiness detection is of significant importance in the field of transportation and road safety. The primary goal of drowsiness detection systems is to identify signs of driver fatigue or sleepiness and alert the driver or trigger safety measures to prevent accidents. Here are some key reasons highlighting the importance of driver drowsiness detection:

Prevention of Accidents: Drowsy driving is a major cause of road accidents. When a driver is fatigued or drowsy, reaction times slow down, attention becomes impaired, and the risk of accidents increases. Drowsiness detection systems can help prevent accidents by alerting drivers in real-time, allowing them to take corrective action or pull over for rest.

Enhanced Road Safety: Detecting and addressing drowsiness contributes to overall road safety. By identifying signs of fatigue early on, these systems help reduce the likelihood of accidents caused by impaired driving.

Protection of Lives and Property: Accidents resulting from drowsy driving can lead to loss of lives and property damage. Drowsiness detection systems play a crucial role in protecting not only the driver but also passengers, pedestrians, and other road users.

Driver Health and Well-being: Continuous monitoring of driver alertness promotes better health and well-being. It encourages responsible driving habits and discourages behaviors that may jeopardize the safety of the driver and others on the road.

Compliance with Regulations: In some regions, there are regulations and standards related to driver fatigue and drowsiness detection. Adhering to these regulations is essential for both individual drivers and organizations involved in transportation.

Increased Productivity in Commercial Transportation: For commercial drivers, especially in the trucking and logistics industry, drowsiness detection systems can contribute to increased productivity. By reducing the risk of accidents and promoting safer driving practices, these systems help ensure timely and efficient transportation of goods.

Integration with Advanced Driver Assistance Systems (ADAS): Drowsiness detection systems are often integrated into advanced driver assistance systems, providing a comprehensive approach to road safety. These integrated systems can include features like lane departure warnings, automatic emergency braking, and adaptive cruise control.

Insurance Benefits: Some insurance companies offer benefits or discounts to drivers or organizations that implement safety measures, including drowsiness detection systems. This can incentivize the adoption of such technologies.

Overall, the importance of driver drowsiness detection lies in its potential to save lives, prevent accidents, enhance road safety, and contribute to the overall well-being of drivers and other road users. As technology continues to advance, the integration of such systems into vehicles is becoming increasingly common and is expected to have a positive impact on transportation safety.

1.2 What is Driver Drowsiness Detection?

Driver's inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and

distraction, however, might have the same effects, that is decreased driving performance, longer reaction time, and an increased risk of crash involvement.

Based on acquisition of video from the camera that is in front of driver perform real-time processing of an incoming video stream in order to infer the driver's level of fatigue if the drowsiness is estimated then it will give the alert by sensing the eyes. A new approach towards automobile safety and security with autonomous region primarily based automatic automotive system is projected during this conception. A drowsy driver detection system and a traffic detection system with external vehicle intrusion dodging primarily based conception. So as to attenuate these problems, we've incorporated driver alert system by watching each the driver's eyes.



Figure 1: Image of driver drowsiness

In this figure 1. the person who is driving the care is feeling the drowsy

1.3 Techniques involved

Eye tracking and detection of face through videos:

Initially endeavoring to generate a preview of the driver through the web camera of the laptop. The driver's preview image is being captured simultaneously by the web camera and the camera forms to give us a preview image. The camera now starts to record the video and automatically saves it in the backend so that it can be analyzed. The recorded video is in grey, HSV color and original form. These different types of recording is because it helps to give us data in all the frames- grey scale or black and white, colored frame and original frame. After the images, videos, frames are being recorded and saved, the eyes are detected from them and then tracked.

Sounding the alarm:

At first trying to generate a preview of the driver through the web camera of the laptop. The driver's preview image is being captured simultaneously by the web camera and the camera forms to give us a preview image. The camera now starts to record the video and automatically saves it in the backend so that it can be analyzed. The recorded video is in grey, HSV color and original form. These different types of recording is because it helps to give us data in all the frames- grey scale or black and white, colored frame and original frame. After the images, videos, frames are being recorded and saved, the eyes are detected from them and then tracked.

Eye tracking and plot generation:

The eye tracking involves to calculate the value for the eye aspect ratio and gives the figures and plots to detect the data being recorded.

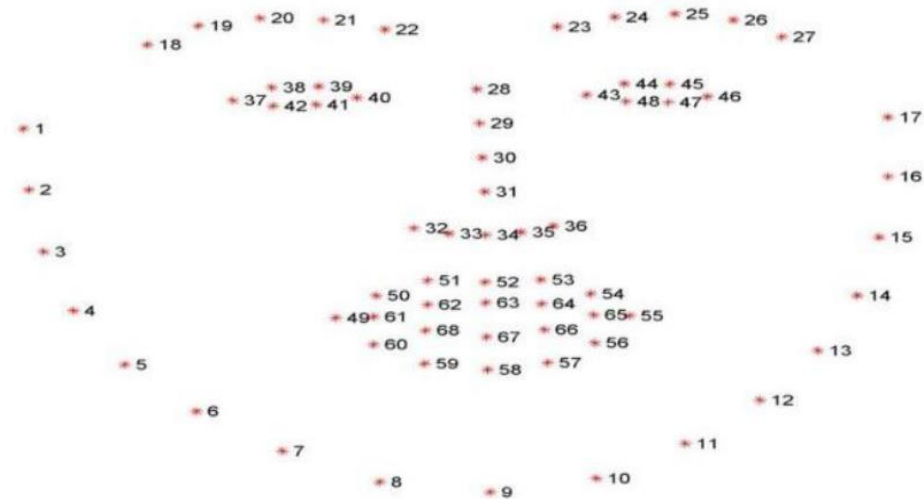


Figure 2: Eye tracking and plot generation

The above figure 2 marks all the regions on the face and gives the proper outlining of the face to be focused on. The numbers marked 37,38,39,40,41,42,43,44,45,46,47,48 are the main region of interest for us in this project.

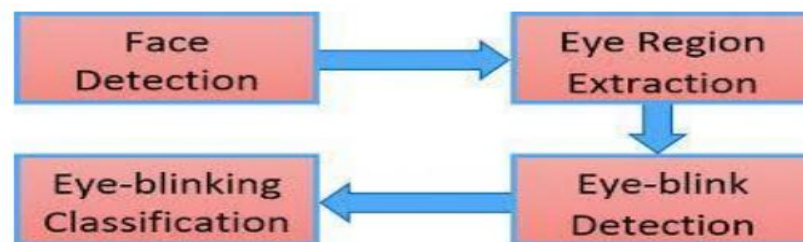


Figure 3: Eye tracking

1.4 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 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. A Relu activation function is used in all the layers except the output layer in which it use Softmax.

1.5 Need for drowsiness detection:

- Drowsiness Detection System is necessary.
- It will protect the driver while driving and will ask him to drive safely.
- No risk of death or any accident will be there.
- The driver will feel safe in the night too while driving.
- Driver will get alert with it.

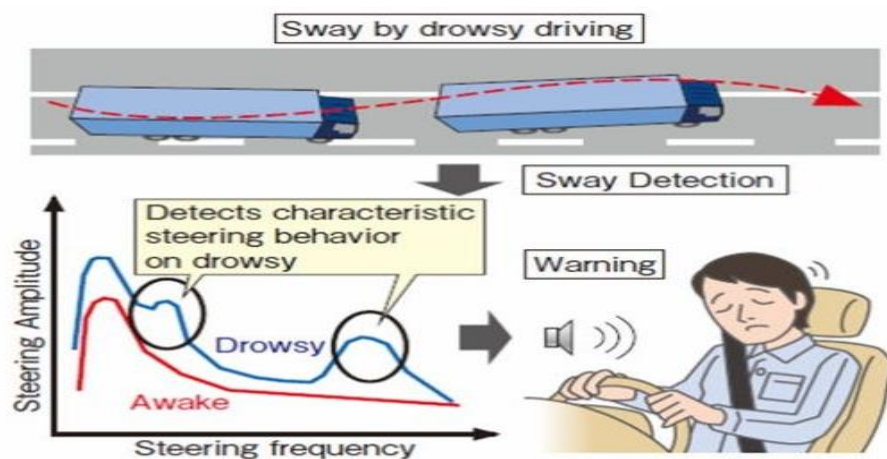


Figure 4: Need for drowsiness detection

CHAPTER 2

RELATED CONCEPT

2. LIBRARIES REQUIRED

The requirement for this Python project is a webcam through which it 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.

- I. OpenCV
- II. dlib
- III. Numpy
- IV. TensorFlow
- V. Keras
- VI. Pygame

2.1 OpenCV

OpenCV, or Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. It was originally developed by Intel and later supported by Willow Garage and Itseez (which later became a part of Intel again). OpenCV is widely used for various applications in computer vision, image processing, and machine learning.

Computer Vision and Image Processing: OpenCV provides a wide range of tools and functions for computer vision and image processing tasks. It includes algorithms for image and video analysis, feature extraction, object recognition, and more.

Open Source and Cross-Platform: OpenCV is an open-source project, meaning that its source code is freely available to the public. It is also cross-platform, supporting various operating systems such as Windows, Linux, macOS, Android, and iOS.

Rich Set of Functions and Algorithms: The library offers a comprehensive set of functions and algorithms for various tasks, including image manipulation, geometric transformations, object tracking, machine learning, camera calibration, and stereo vision.

Programming Language Support: OpenCV is primarily written in C++ but provides interfaces for multiple programming languages, including Python, Java, and MATLAB. This makes it accessible to a wide range of developers with different language preferences.

Community and Documentation: OpenCV has a vibrant and active community of developers and researchers who contribute to its development and improvement. The library is well-documented, with extensive documentation, tutorials, and resources available for users.

2.2 DLib

Dlib contains a wide range of machine learning algorithms. All designed to be highly modular, quick to execute, and simple to use via a clean and modern C++ API. It is used in a wide range of applications including robotics, embedded devices, mobile phones, and large high performance computing environments. There are a number of well-known machine learning libraries. However, many of these libraries focus on providing a good environment for doing research using languages other than C++. Two examples of this kind of project are the Shogun (Sonnenburg et al., 2006) and Torch (Collobert and Bengio, 2001) toolkits which, while they are implemented in C++, are not focused on providing support for developing machine learning software in that language. Instead they are primarily intended to be used with languages like R, Python, Matlab, or Lua. Then there are toolkits such as Shark (Igel et al., 2008) and dlib-ml which are explicitly targeted at users who wish to develop software in C++. Given

these considerations, dlib-ml attempts to help fill some of the gaps in tool support not already filled by libraries such as Shark. It is hoped that these efforts will prove useful for researchers and engineers who wish to develop machine learning software in this language.

2.3 Numpy

NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages. The NumPy library contains multidimensional array and matrix data structures (you'll find more information about this in later sections). It provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

2.4 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. See the TensorFlow

documentation for complete details on the broader TensorFlow system. TensorFlow APIs are arranged hierarchically, with the high-level APIs built on the low-level APIs. Machine learning researchers use the low-level APIs to create and explore new machine learning algorithms. In this class, you will use a high-level API named `tf.keras` to define and train machine learning models and to make predictions. `tf.keras` is the TensorFlow variant of the open-source Keras API.

2.5 Keras

Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation. Keras is relatively easy to learn and work with because it provides a python frontend with a high level of abstraction while having the option of multiple back-ends for computation purposes. This makes Keras slower than other deep learning frameworks, but extremely beginner-friendly.

Keras allows you to switch between different back ends. The frameworks supported by Keras are:

- Tensorflow
- Theano
- PlaidML
- MXNet
- CNTK (Microsoft Cognitive Toolkit)

Out of these five frameworks, TensorFlow has adopted Keras as its official high-level API. Keras is embedded in TensorFlow and can be used to perform deep learning fast as it provides inbuilt modules for all neural network computations. At the same time, computation involving tensors,

computation graphs, sessions, etc can be custom made using the Tensorflow Core API, which gives you total flexibility and control over your application and lets you implement your ideas in a relatively short time.

2.6 Pygame

Pygame is a popular Python library for game development. It provides a basic interface for designing games, including managing graphics, sound, and user interaction. Pete Shinnars created Pygame after its development stagnated. It has been a community effort since 2000[8] and is distributed under the GNU Lesser General Public License for free software. Python must be installed on your machine before you can use Pygame. Pygame may be installed using pip, the Python package manager, once Python is installed.

CHAPTER 3

SOFTWARE AND HARDWARE

3. SOFTWARE AND HARDWARE REQUIREMENT

3.1 Hardware Requirements

System: Windows or Mac PC x86 64-bit CP

(Intel AMD architecture)

RAM: Minimum 4 GB

Hard Disk: Maximum 500 GB

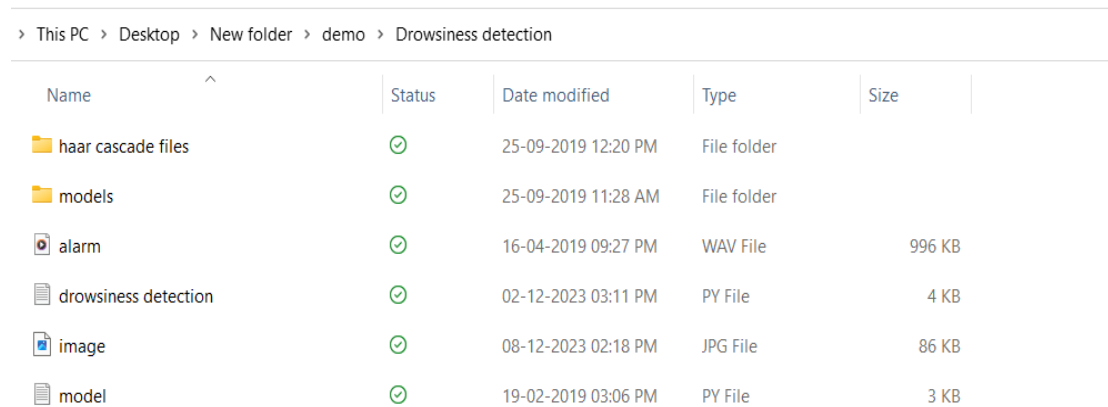
3.2 Software Requirement

Operating System: Windows 10/11.

IDE: Jupyter Notebook or any python IDE

CHAPTER 4 IMPLEMENTATION

4.1 IMPLEMENTATION DETAILS



The screenshot shows a Windows File Explorer window with the address bar displaying the path: > This PC > Desktop > New folder > demo > Drowsiness detection. The main area contains a table of files and folders.

Name	Status	Date modified	Type	Size
haar cascade files	✓	25-09-2019 12:20 PM	File folder	
models	✓	25-09-2019 11:28 AM	File folder	
alarm	✓	16-04-2019 09:27 PM	WAV File	996 KB
drowsiness detection	✓	02-12-2023 03:11 PM	PY File	4 KB
image	✓	08-12-2023 02:18 PM	JPG File	86 KB
model	✓	19-02-2019 03:06 PM	PY File	3 KB

Figure 5: Folder of Drowsiness files

- In Figure 5 the “haar cascade files” folder consists of the xml files that are needed to detect objects from the image. In this case, it is detecting the face and eyes of the person.
- The models folder contains our model file “cnnCat2.h5” which was trained on convolutional neural networks.
- Which have an audio clip “alarm.wav” which is played when the person is feeling drowsy.
- “Model.py” file contains the program through which the classification model built by training.
- “Drowsiness detection.py” is the main file of our project. To start the detection procedure, this file has to run.

4.2 Algorithm used

Step 1- Look for faces in the input video being recorded.

Step 2 - Test the input faces and capture them as images for processing them.

Step 3 - If the eyes get little drowsy sound an alarm.

Step 4 - Keep the eyes tracked through the video being saved at the backend.

Step 5 - Generate the graphs and plots showing the score of drowsiness.

The above algorithm is applied in our project

4.3 Flowchart

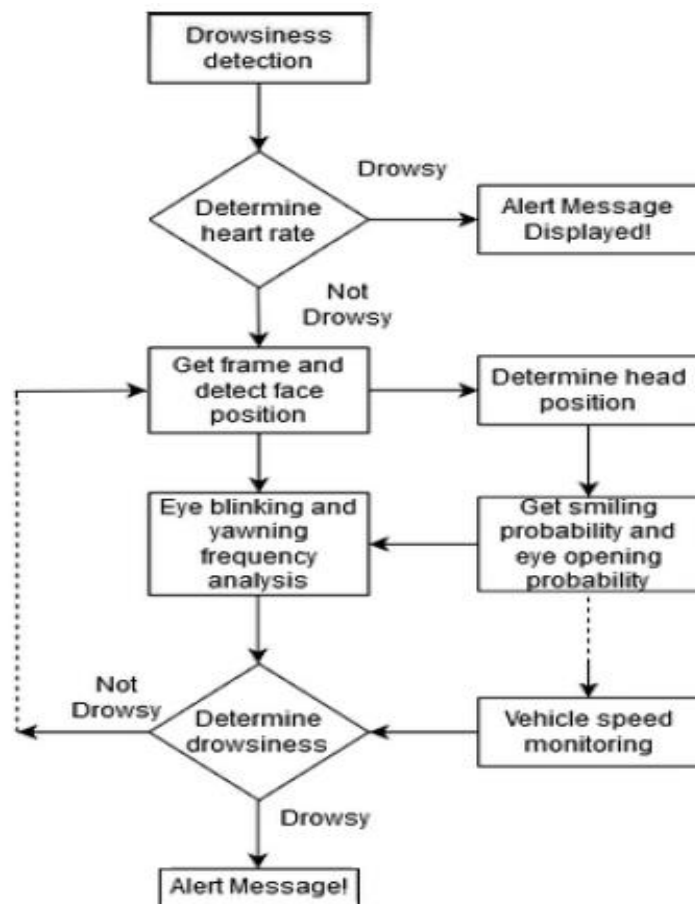


Figure 6: System architecture

- a) The chart depicts how the full function of the drowsiness detection is carried out.
- b) It describes the importance of each step that is required to complete the detection of drowsiness.
- c) As it seems that how first the brightness and contrast level of the camera is adjusted.
- d) Then the face is detected.
- e) If it is successfully previewed then only the further step is taken.
- f) The eye detection takes place.
- g) The decision for proper eye detection is taken and then the eye region is focused and extracted.
- h) The eyes are determined whether they are closed or opened.
- i) The drowsiness is calculated through the data being stored and saved.
- j) The drowsiness is judged now.
- k) If the drowsiness is drowsy then alarm rings loud and alert is given to him.
- l) This is how it works.

4.4 Steps for drowsiness detection

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 ]
```

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()**

4.4 CODE

#Driver 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('haarcascade_files/haarcascade_frontalface_alt.xml')
face=cv2.CascadeClassifier('haarcascade_files/haarcascade_frontalface_alt.xml')
leye=cv2.CascadeClassifier('haarcascade_files/haarcascade_lefteye_2splits.xml')
reye=cv2.CascadeClassifier('haarcascade_files/haarcascade_righteye_2splits.xml')

lbl = ['Close', 'Open']
model = load_model('models/cnn_cat2.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(r_eye)
        if rpred.any() > 0.5:
            lbl = 'Open'

```

```

else:
    lbl = 'Closed'
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(l_eye)
    if lpred.any() > 0.5:
        lbl = 'Open'
    else:
        lbl = 'Closed'

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

        cv2.imwrite(os.path.join(path, 'image.jpg'), frame)

        try:

            sound.play()

        except:

            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_mode=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=(24,1)),

    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',m
etrics=['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)

```

4.5 OUTPUTS

These are the following outputs of our project .



Figure 7 : Output 1

In figure 7: Person Eyes are open so it shows that the given score is not updating i.e 0

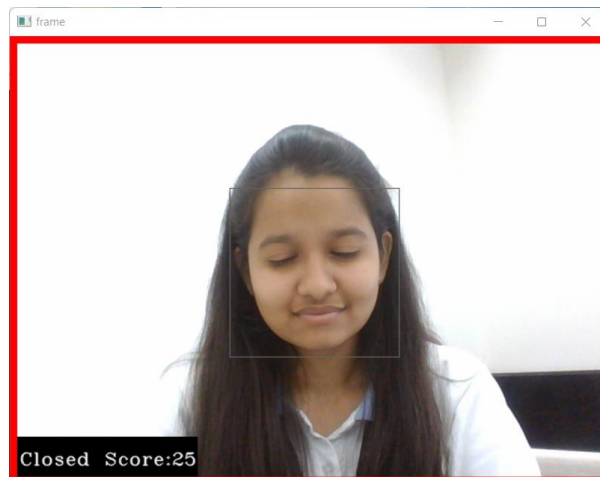


Figure 8 : Output 2

In figure 8: Person Eyes are closed so it shows that the given score is updating i.e 25

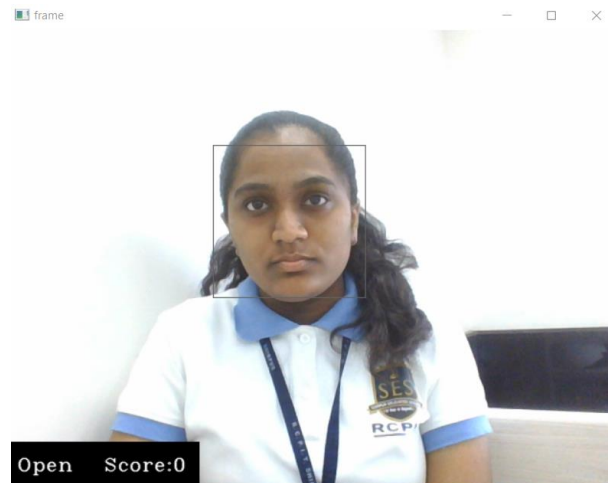


Figure 9: Output 3

In figure 9: Person Eyes are open so it shows that the given score is not updating i.e 0

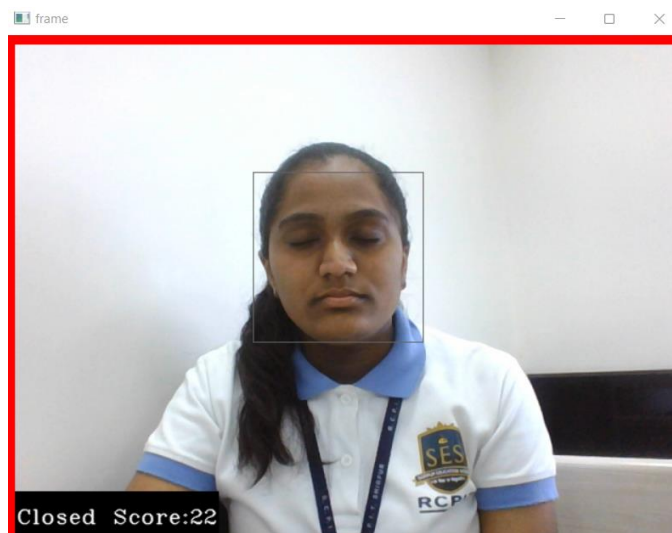


Figure 10 : Output 4

In figure 10: Person Eyes are closed so it shows that the given score is updating i.e 22

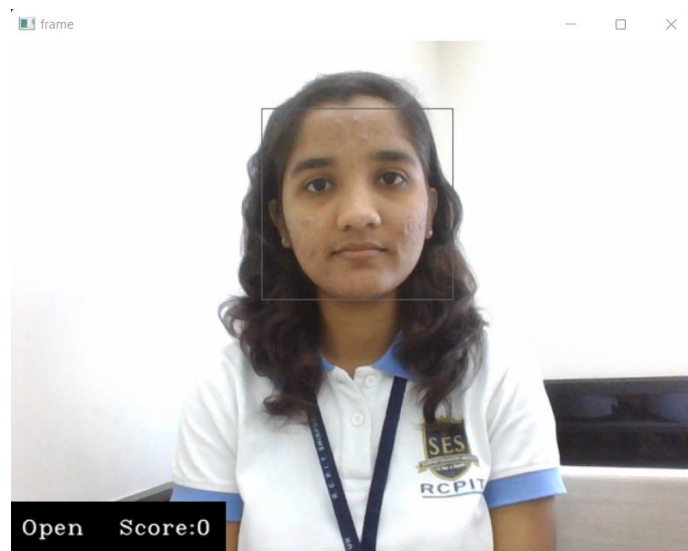


Figure 11: Output 5

In figure 11: Person Eyes are open so it shows that the given score is not updating i.e 0

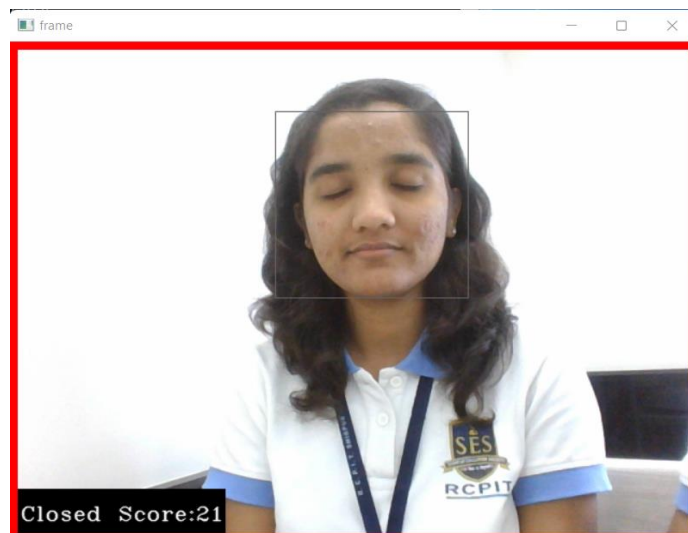


Figure 12: Output 6

In figure 12: Person Eyes are closed so it shows that the given score is updating i.e 21

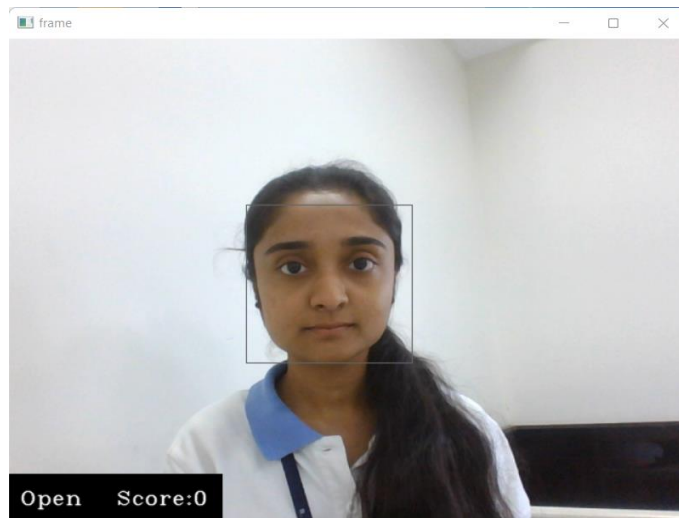


Figure 13: Output 7

In figure 13: Person Eyes are open so it shows that the given score is not updating i.e 0

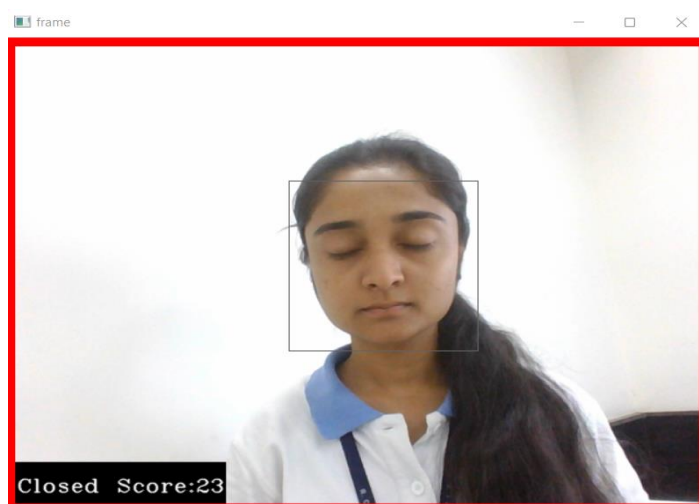


Figure 14 : Output 8

In figure 14: Person Eyes are closed so it shows that the given score is updating i.e 23

CONCLUSION

Driver drowsiness detection systems represent a valuable tool in the ongoing efforts to enhance road safety. While they are not a panacea and must be used in conjunction with other safety measures, these systems have the potential to save lives by providing timely alerts and encouraging proactive driver interventions. As technology continues to advance, it is expected that drowsiness detection systems will become even more sophisticated and integrated into the broader landscape of smart and safe transportation. Hence driver drowsiness detection is a very important part for the driver while he drives the car. So every driver should have a drowsiness detection system in his vehicle so that he can be almost safe and secure from road accidents, deaths from road accidents. A sleep of even a second can take the life of the driver. Then why to take any risk? This risk can be prevented when the driver uses the drowsiness detection system.

REFERENCES/BIBLIOGRAPHY

1. *Driver Monitoring and Drowsiness Detection by Steering Signal Analysis* Originally published: 2007
Author: Tobias Altmüller
2. *Drowsiness Detection Using Image Processing*
Author: Hanojhan Rajahrajaasingh
3. *Driver Drowsiness Detection: Systems and Solutions*
Author : Aleksandar Čolić, Borko Furht, and Oge Marques
4. <https://www.geeksforgeeks.org/python-opencv-drowsiness-detection/>
5. <https://github.com/topics/drowsiness-detection>
6. <https://learnopencv.com/driver-drowsiness-detection-using-mediapipe-in-python>
7. <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.nauto.com%2Fblog%2Fintroducing-driver-drowsiness-alerts&psig=AOvVaw0M2PMDTPbxgxNPv045Bs2E&ust=1702575258724000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLCrnaT5jIMDFQAAAAAdAAAAABAD>
8. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6174048/>