Deep Learning based Driver Drowsiness Detection

B.E. project report submitted in partial fulfilment
of the requirements of the degree of

Information Technology

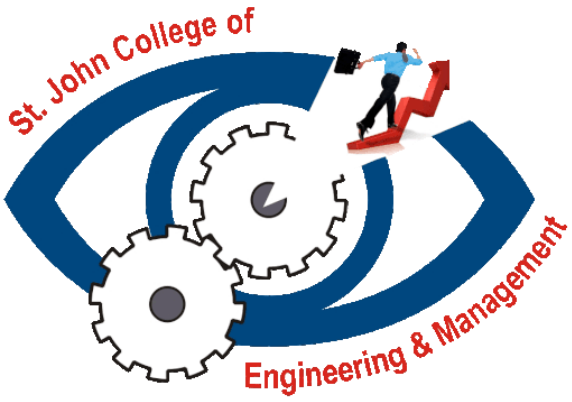
By

Santoshi Sabat 61

Parth Patel 51

Chirag Pavesha 54

Supervisor
Ms. Shraddha More
Assistant Professor



Department of Information Technology
St. John College of Engineering and Management,
VevoorRoad, Palghar(E),401404

University of Mumbai
2021–2022

CERTIFICATE

This is to certify that the project entitled **"Deep Learning based Driver Drowsiness Detection"** is a bonafide work of **Santoshi Sabat(PID No. 1184002)**, **Parth Patel (PID No EU1184017), and Chirag Pavesha (PID No. 1184005)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of B.E. in Information Technology.

Ms. Shraddha More

Project Guide

Mr. Arun

Head of Department

Dr. G.V. Mulgund

Principal

Project Report Approval for B. E.

The project report entitled **Deep Learning based Driver Drowsiness Detection** by **Santoshi Sabat, Parth Patel, Chirag Pavesha** is approved for the degree of **Bachelor of Engineering** in **Information Technology.**

Examiners:

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Santoshi Sabat (1184002)

Parth Patel (1184017)

Chirag Pavesha (1184005)

Date:

Abstract

Driver drowsiness and distraction are now widely recognized as important contributors to deadly road accidents around the world. This typically occurs when a driver has not slept enough, but it can also occur as a result of untreated sleep problems, drugs, alcohol consumption, or shift employment. As a result, driver monitoring and identification are becoming increasingly important features of car safety systems. Head position, gaze direction, yawning, and eye state analysis are among the essential aspects. This research proposes a driver drowsiness detection system that uses eye blink and mouth opening and closing counts to detect drowsiness. When the driver's eyes are closed for an extended period of time, an alert sound is generated to notify him. Furthermore, the vehicle's owner is notified by e-mail if the driver is observed to be napping off more than a few times, in order to certify that the driver is taking certain steps to avoid falling asleep. The output of the system proposed in the paper on Deep learning technology of Dlib which uses CNN (Convolutional Neural Network) as its base algorithm for accurate detection, OpenCV, and Raspberry Pi environments with a mounted camera for the same, show that system achieves good result when it comes to drowsiness detection, reducing the overall number of accidents on the streets. For Realtime video input, the proposed method had a 96% of accuracy.

Keywords: CNN, Deep Learning, Dlib, Drowsiness Detection, Eye blink, OpenCV, Raspberry Pi.

Table of Contents

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Network
CV	Computer Vision
ECG	Electrocardiography
EAR	Eye Aspect Ratio
EEG	Electroencephalography
HMM	Hidden Markov Model
kNN	K Nearest Neighbor
MAR	Mouth Aspect Ratio
ML	Machine Learning
OS	Operating System
WBS	Work Breakdown Structure
WHO	World Health Organization

Chapter 1
Introduction

1.1 Introduction

One of the major contributing factors to motor vehicle accidents is driver drowsiness. Drowsy drivers are more likely to cause a serious accident because they are unable to react quickly enough to dangerous situations. To avoid such an accident, it's crucial to identify drowsiness as soon as possible and precisely . According to the survey, 846 people died in 2021 as a result of sleepy driving. Between 2021 and 2018, sleepy driving was blamed for an estimated 2,363,031 crashes each year. On a yearly basis, car crashes account for around 996 deaths, 49,000 injuries, and 51,000 property damage . Drowsy driving happens when a driver is exhausted while driving, making it impossible for him to remain awake. This frequently happens when the driver doesn't get enough sleep. This can also happen if he/she has sleep problems like insomnia and even shift work sleep disorder (SWSD). This resulted in a collision with another vehicle for the drowsy driver. A human's drowsiness is identified by a few distinct motions and facial expressions such as the brief moment of eye closing, the mouth opens in a yawn, the jaw dips, and the neck tilts. This research concentrates on detecting tiredness and classifying a driver as drowsy by observing the eyes and mouth. The video input can be acquired by placing the camera on the vehicle's dashboard for real-time application of the model and can easily collect the driver's face, hands, upper torse and even through non-tinted spectacles in the video stream. The Dlib model has been taught to recognize 68 facial landmark points . The features of drowsiness are retrieved, and the driver is notified if drowsiness is recognized. The model is not bound to have any prior knowledge about the person who will be using or testing the system.

This research provides deep learning strategy for observing driver drowsiness based on computer vision. This strategy adopts the input stream from driver's face and divides drowsiness into three categories (normal, yawn, and drowsy). This model's major benefit is that it is compact enough to be implemented on an handheld microcontroller with camera embedded on it while maintaining decent accuracy. A compressed model is important for integrating a driver's tiredness detection system into a normal car. Whereas a person who can fall asleep at any moment, having a real-time classifier for sleepiness detection that consumes little to no electricity and can be effortlessly installed on a car.

1.2 Problem Formulation

Current drowsiness detection systems monitoring the driver's condition requires complex computation and expensive equipment, not comfortable to wear during driving and is not suitable for driving conditions. A drowsiness detection system which use a camera placed in front of the driver is more suitable to be use but the physical signs that will indicate drowsiness need to be located first in order to come up with a drowsiness detection algorithm that is reliable and accurate. Lighting intensity and while the driver tilt their face left or right are the problems occur during detection of eyes and mouth region. Therefore, this project aims to analyze all the previous research and method, hence propose a method to detect drowsiness by using video or webcam. It analyzes the video images that have been recorded and come up with a system that can analyze each frame of the video.

1.3 Description

According to National Highway Traffic Safety Administration (NHTSA), drowsy driving leads to over 71,000 injuries, 1,500 deaths, and $12.5 billion in monetary losses per year. The main issue that needs to be addressed is that the driver should be warned well before time to take precautionary measures. This system describes a non-intrusive approach that takes into consideration the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) to quantify drowsiness by locating, monitoring, and analyzing both eyes and mouth, even in the dark. The real-time EAR is compared with the initial EAR of the drive and the MAR is compared with a set threshold of 20. The system alerts he user by issuing audio warnings through espeak and by alarm.

1.4 Motivation

As the death due to road accidents are increasing day by day, the threat that it could not only happen to us and our dear ones because of our fault but also because of the people who are driving the cars around us on the road due to lack of concentration or just overworking and sleep deprivation. We could not prevent the inevitable fate of the person but we could avoid it and can be safe on our side by developing a system which is easy to install and can potentially avoid the accidents to some extend by focusing on the behavioral approach of the person

while driving because when a person is feeling lack of concentration due to some reason their face could tilt, the position of head could change and the fact that the facial feature could also tell the story . So due to the alarming rate of the accidents which could have been avoided gave us the motivation to undertake this project.

1.3 Objectives

The objectives are as follows:

- To get the image of facial features such as eyes, mouth and the whole head position.
- To perform neural network algorithm for facial recognition.
- To extract the facial data for future use.
- To implement the system using Deep learning and IoT
- To show accuracy of the model.

1.4 Scope

There are many products out there that provide the measure of drowsiness 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. The system uses CNN which requires less pre-processing and has the ability to handle large, unstructured data e.g. image classification, speech recognition.

Chapter 2

Literature Survey

2.1 A deep learning based approach to detect drowsy driver in real time

In 2019, Anshul Pinto et al. [1] projected a system that is built on convolutional neural network for a range of machine learning and predictive tasks, and it takes a picture of an eye as input and classifies it as open or closed. The performance of CNN was tested using eye blink detection algorithm and a formula called EAR as per results section. On average, the CNN technique was 93.3 percent accurate, while the EAR approach was 80.4 percent accurate. The system was developed using a yawn detection mechanism that might be linked with the status of the eyes to act as a metric for detecting the driver's alertness.

2.2 Driver Drowsiness Classification based on Eye Blink and Head Movement Features using the kNN Algorithm

In 2020, Mariella Dreißig et al. [2] presented a system in which the eyelid movement was studied using classification kNN algorithm on dataset which contain 134 hours of recording and with Karolinska Sleepiness Scale (KSS) values it became easy to determine the drowsiness. The upper bound of k was selected as 1000 based on the distinct data points. The identification of appropriate features was a key component of the k-NN-based classification. With a crucial decision being made on the value of k the accuracy of the model changes is the value of k is not chosen upto the mark. Different number of Head movements and eye features were detected which served as a basis for designing the model.

2.3 Driver Drowsiness Detection System Based on Visual Features

In 2018, Fouzia et al. [3] presented a system sleepiness monitoring framework founded on a shape predictor algorithm that detects and counts the blink rate of the eyes before detecting drowsiness in real time. Image processing algorithms are used to gather information about the eye status, which deliver a noninvasive method of detecting drowsiness without causing any discomfort or inconvenience. The same framework needs to be used for implementing yawn detection which further would hail the drowsiness detection.

2.4 Driver Distraction Detection using Deep Learning and Computer Vision

In 2018, Kusuma.S1 et al.[4] presented a system which detects the driver's drowsiness by using deep learning and computer vision. Whenever the driver is not concentrating, an alarm ring's. Image recognition is made possible through a model called deep convolutional neural networks (CNN). Deep learning techniques could be used to reduce the training time.

2.5 Robust Drowsiness Detection for Vehicle Driver using Deep Convolutional Neural Network

In 2020, Saifuddin Saif, et al. [5] suggested a Convolutional Neural Network (CNN) technique for detecting microsleep and tiredness using a machine learning algorithm. In the research done, the mounted camera detects the driver's face points, which is then fed to the detecting algorithm to determine a person's drowsiness status. The model gets trained on the dataset of open and closed eyes in different lighting conditions as it is only trained to detect the tiredness based on eye blink and other signs of drowsiness are missed. Given the system provides a live detection and feedback on the subject matter of eyes for sleepiness. It focuses on the development of instantaneous drowsiness detection system to avoid the catastrophic consequence of driver due to accident. This method creates a variety of driver faces and a model to detect sleepy driver with 93 percent enrolments on a single feature, while different symptoms of tiredness, such as yawning and even head tilting, should be monitored by the system.

No.	Paper Title	Journal	Conclusion	Research Gaps
[1]	**A Deep Learning Approach to Detect Drowsy Drivers in Real Time**		System uses convolutional neural network for variety of machine learning and predictive tasks & taking in an image of an eye as the input and it classifies whether the eye is open or closed. The results section evaluated the performance of CNN with a blink detection algorithm that uses a metric called EAR. The CNN approach gave 93.3% accuracy on an average whereas the EAR approach gave 80.4% accuracy.	Implementing a mechanism to detect yawns, which could be combined with the state of the eyes to serve as a metric to detect the vigilance of the driver.
[2]	**Driver Drowsiness Classification based on Eye Blink and Head Movement Features using the kNN Algorithm**		In this paper the eyelid movement was studied using classification kNN algorithm on dataset which contain 134 hours of recording and with Karolinska Sleepiness Scale (KSS) values it became easy to determine the drowsiness. The upper bound of k was selected as 1000 based on the number of datapoints. Multiple Head Movements and blink features were detected which served as a basis for designing the model	In kNN output completely relies on nearest neighbors which may not be good choice and if the value of k is chosen wrong the classification would not give satisfactory result.
[3]	**Driver Drowsiness Detection System Based on Visual Features**		In this paper, a Drowsiness detection framework based on shape predictor algorithm, detects the eyes, and also counts the eye blink rate followed by drowsiness detection at real time. The details about the eye status is obtained through image processing algorithms, which offer a noninvasive approach to detect drowsiness without any annoyance and interference	The detection of yawning of the driver would be implemented using same frame work for detecting further details about the drowsiness of driver.

Table 2.1: Literature Review 1

[4]	**Driver Distraction Detection using Deep Learning and Computer Vision**		In this system it detects the driver's drowsiness by using deep learning and computer vision. Whenever the driver is not concentrating, an alarm ring's. Image recognition is made possible through a model called deep convolutional neural networks (CNN).	Deep learning techniques could be used to reduce the training time.We should use containerized services like kubernetes and docker to ship them as a single entity
[5]	**Robust Drowsiness Detection for Vehicle Driver using Deep Convolutional Neural Network**		In this paper driver drowsiness is detected by extracting facial features initially and then performs face alignment CNN based machine learning approach and achieved accuracy rate of 83% and pass through CNN based deep learning model.	Proposed research performed extensive experimentation where accuracy rate of 98.97% was achieved using frame rate of 35 fps which is higher comparing with previous research results. Experimental results reveal the effectiveness of the proposed methodology.

Table 2.2: Literature Review 2

Chapter 3

Requirement Analysis

3.1 Functional Requirements

Eye Aspect ratio Calculation

Mouth Aspect Ratio Calculation

Yawn Calculation

Head tilt Detection

Arithmetic Formula

3.2 Non Functional Requirements

Python Implementation

3.3 Specific Requirements

3.3 Hardware and Software Requirements: Training a Machine Learning Program has minimum requirement and some IoT hardware for measuring parameters.

3.3.1 Hardware Requirements:

1. Raspberry Pi Board
2. Camera
3. Buzzer
4. Min. RAM 512MB
5. Power supply

3.3.2 Software Requirements:

1. Python Environment 3.7 or later with the required libraries
2. PyCharm
3. Raspberry Pi OS

3.3.3 UML Diagram:

3.3.3 Use Case Diagram:

A use case diagram is a graphical depiction of a user's possible interaction with a system. A use case diagram shows various cases and different types of users the system has and will often be accompanied by other types of diagram as well.

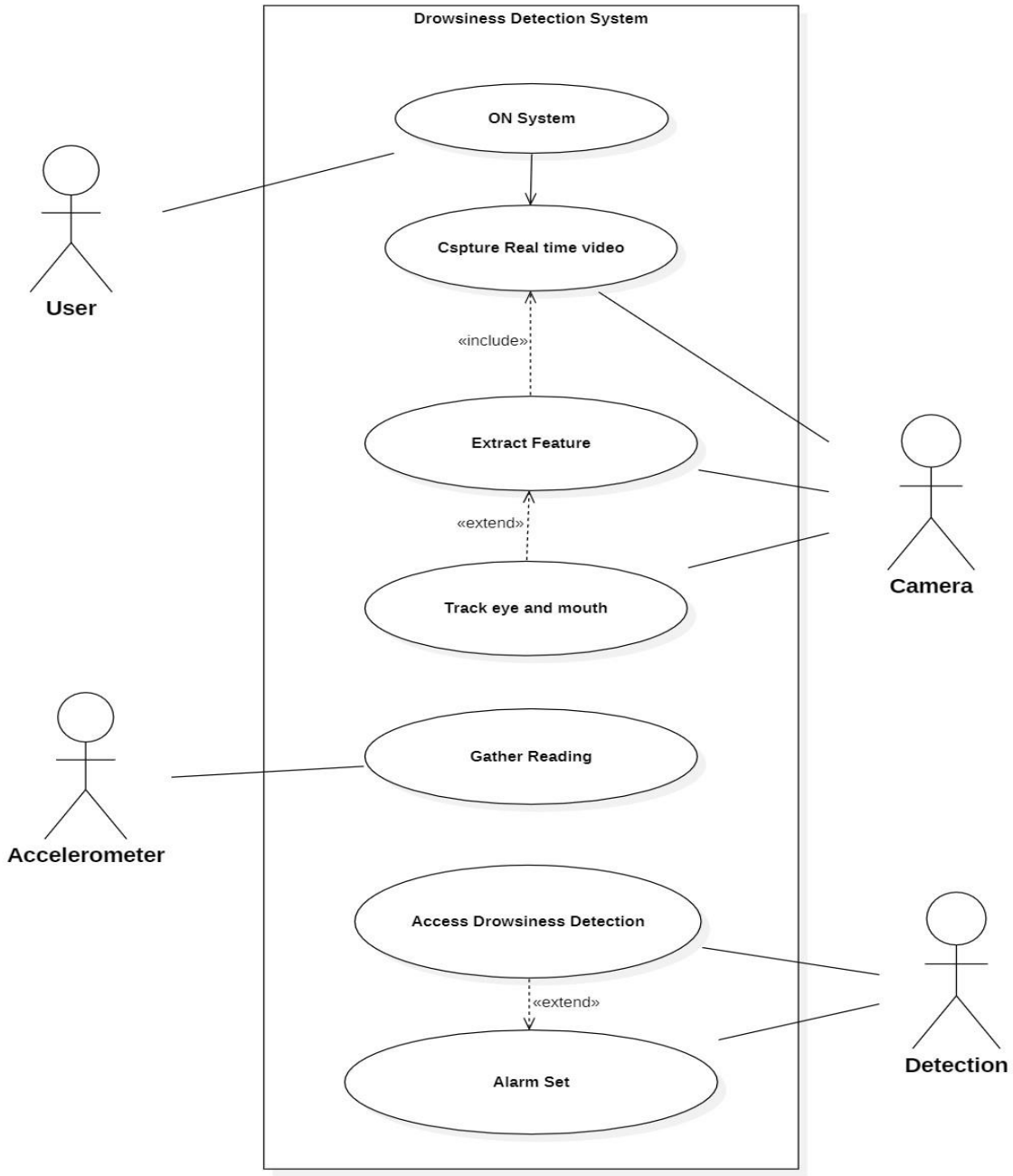


Figure 3.3.1- Use Case Diagram

Chapter 4
Analysis Modeling

4.1 Activity Diagram:

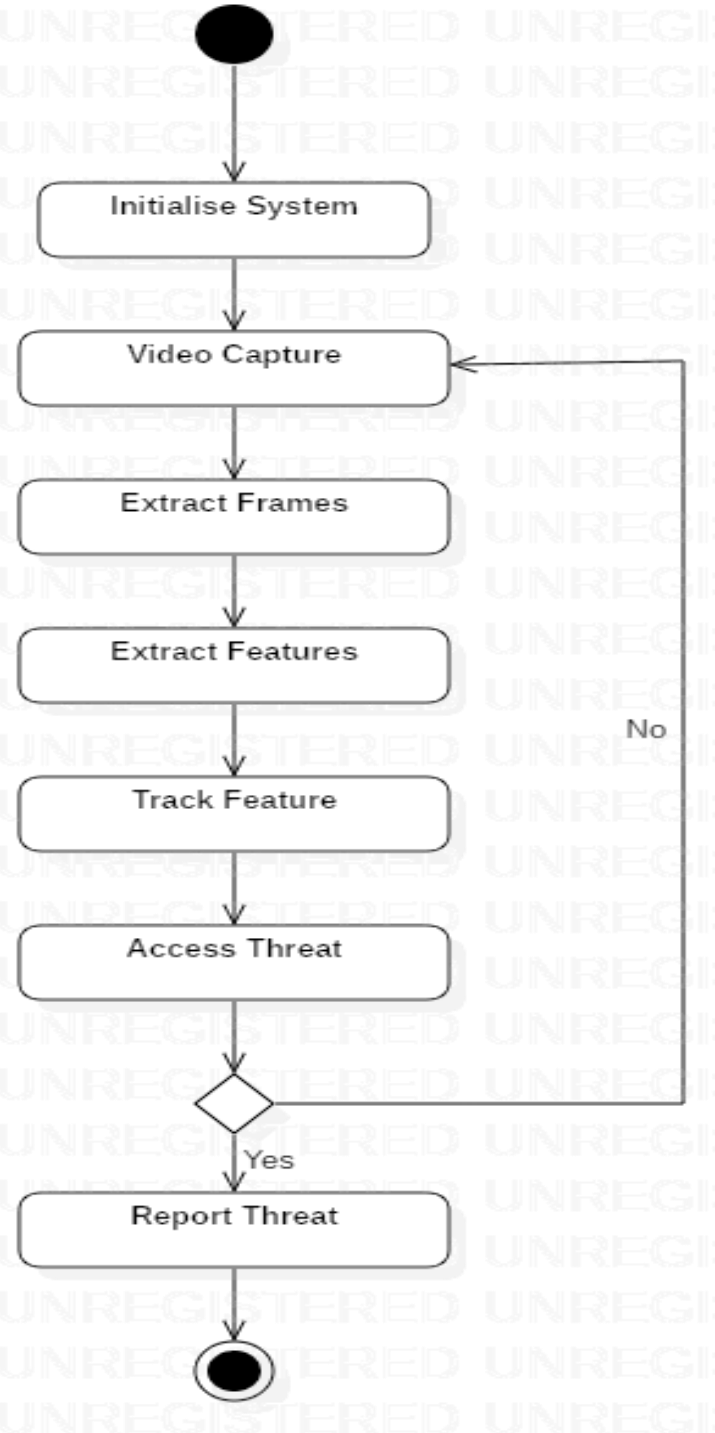


Figure 4.1: Activity Diagram

4.2 Class Diagram:

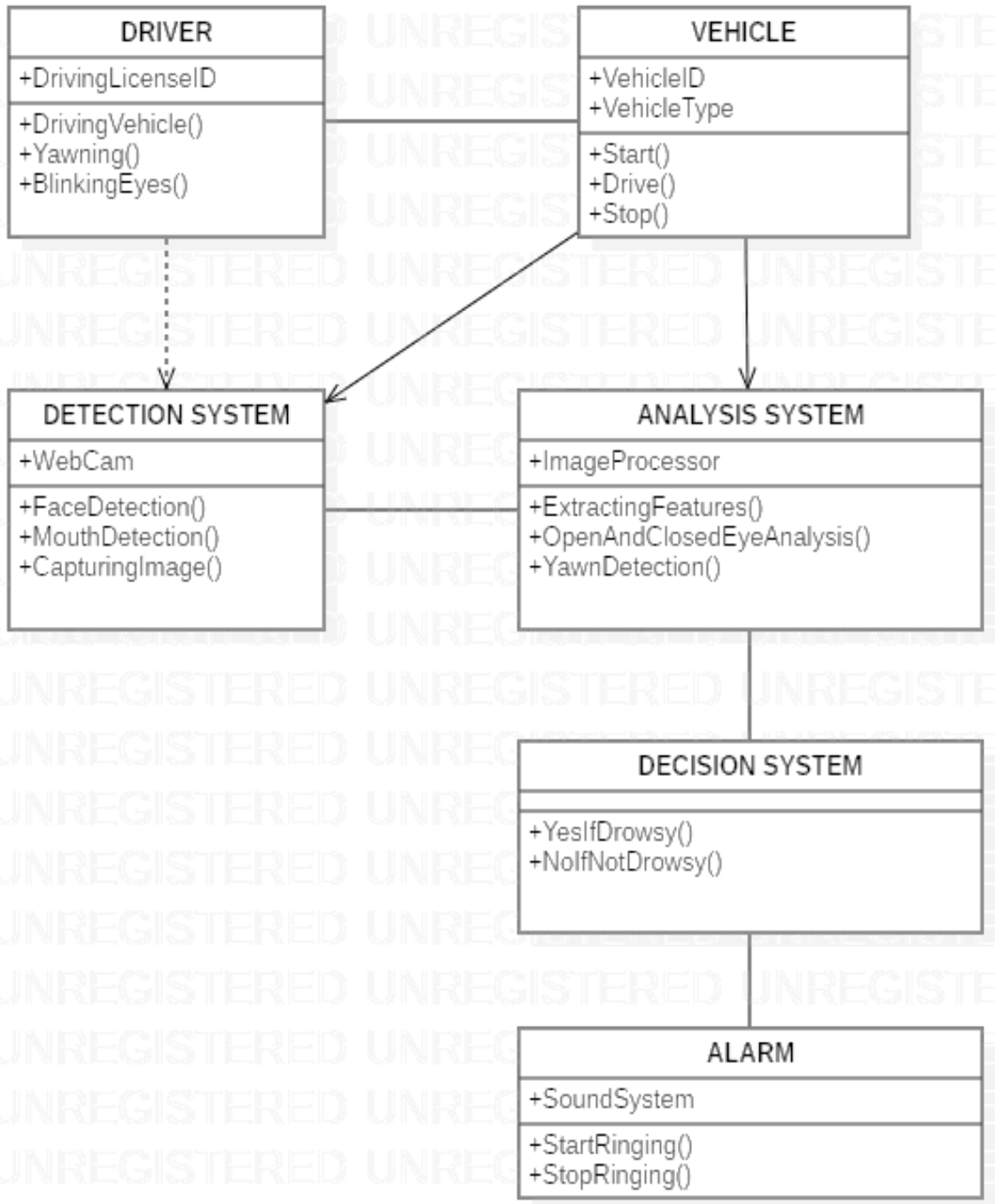


Figure 4.2: Class Diagram

4.3 Sequence Diagram:

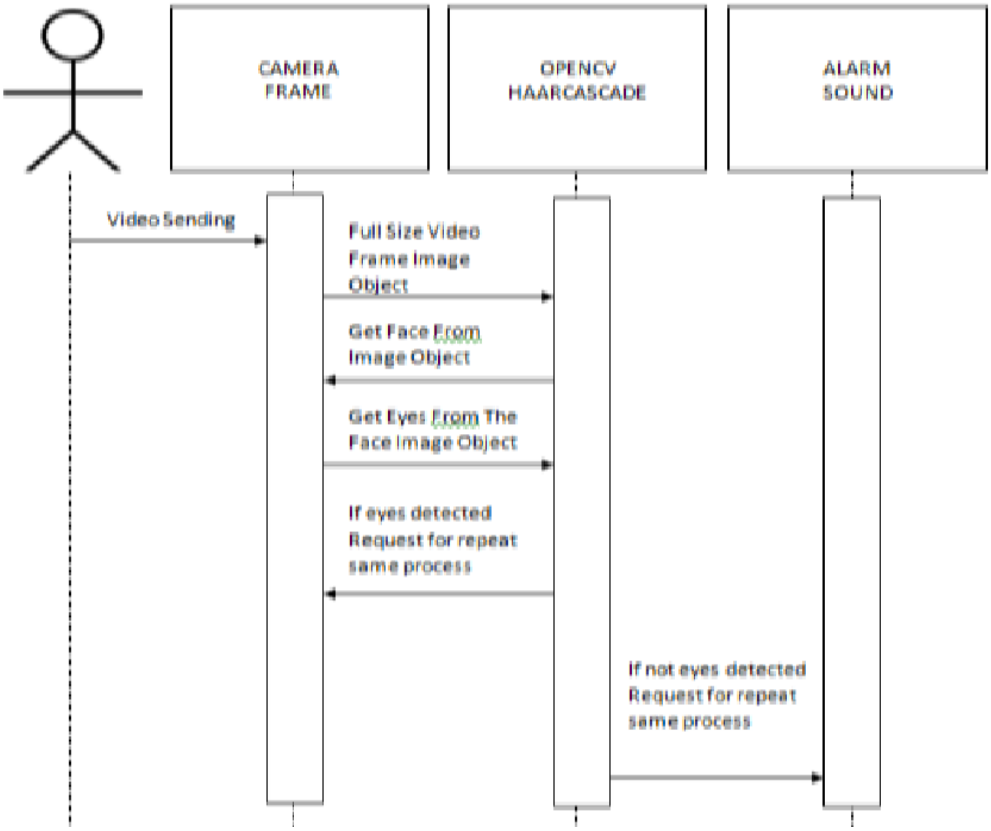


Figure 4.3: Sequence Diagram

4.4 Functional Modelling

4.4.1 DFD 0

4.4.2 DFD 1

4.4.3 DFD 2

4.5 Timeline Chart:

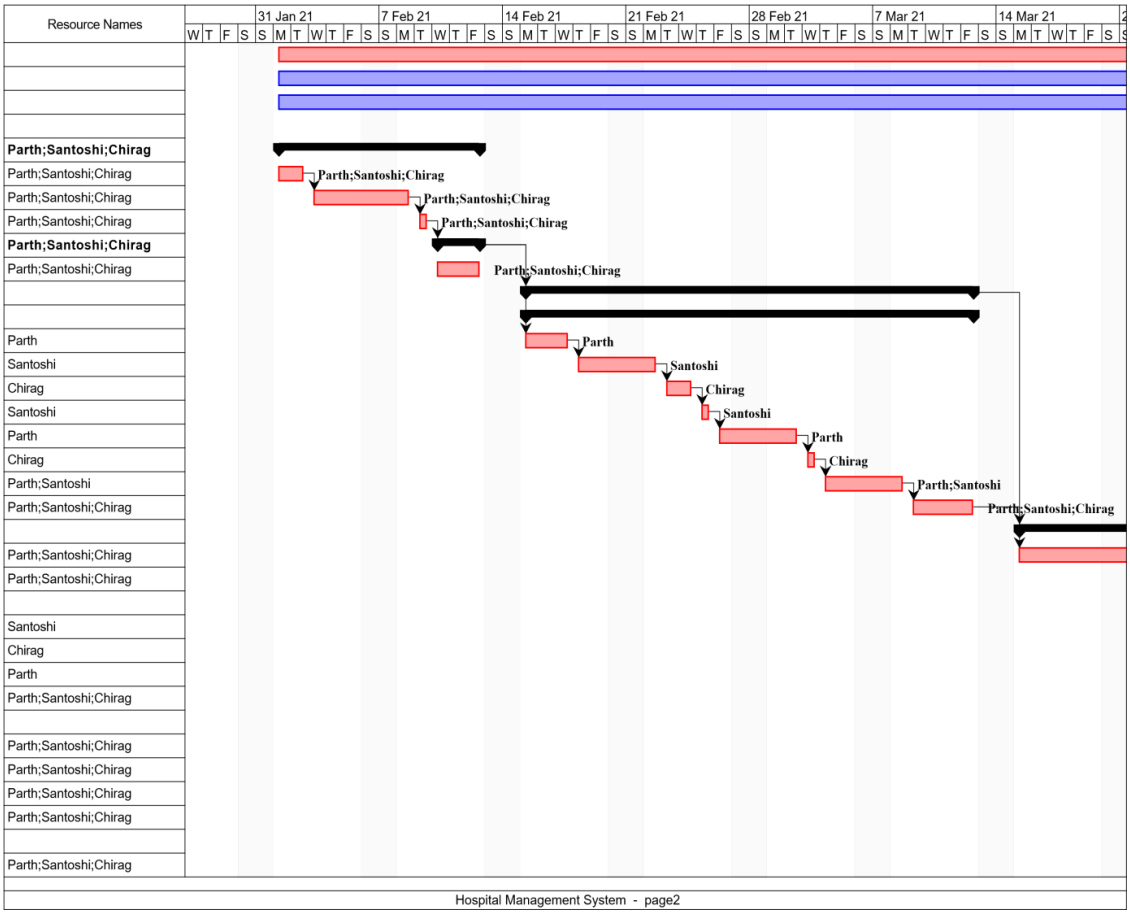
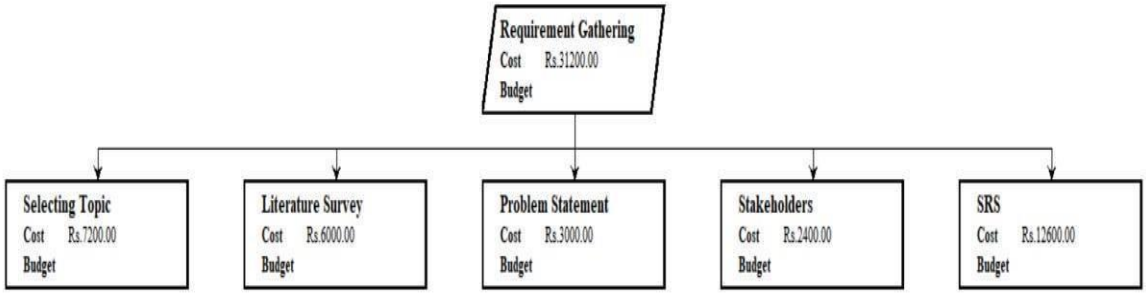
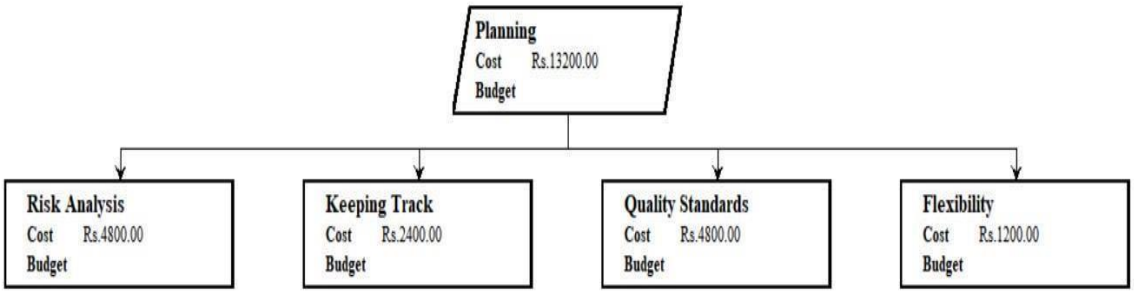
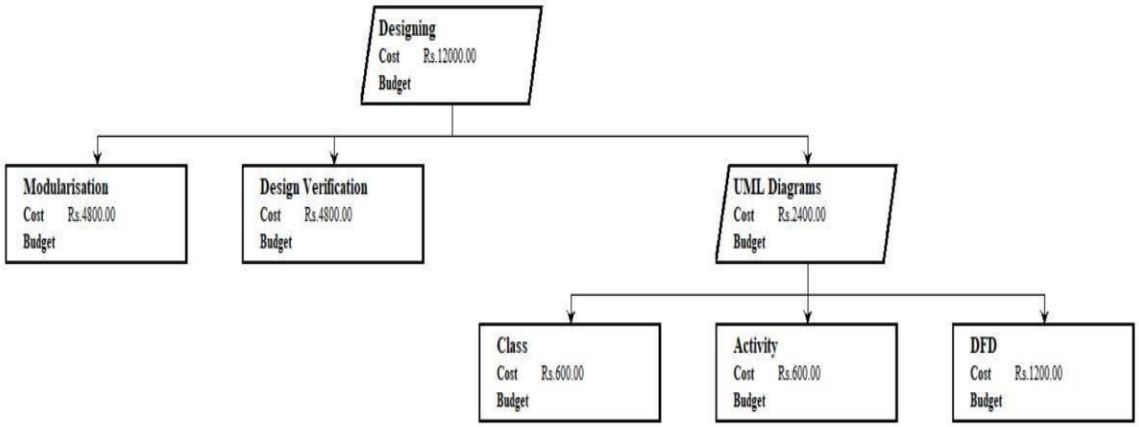


Fig 4.5 Timeline chart

4.6 Work Breakdown Structure (WBS) Chart:


Requirement Gathering
Cost Rs.31200.00
Budget
Selecting Topic
Cost Rs.7200.00
Budget
Literature Survey
Cost Rs.6000.00
Budget
Problem Statement
Cost Rs.3000.00
Budget
Stakeholders
Cost Rs.2400.00
Budget
SRS
Cost Rs.12600.00
Budget


Planning
Cost Rs.13200.00
Budget
Risk Analysis
Cost Rs.4800.00
Budget
Keeping Track
Cost Rs.2400.00
Budget
Quality Standards
Cost Rs.4800.00
Budget
Flexibility
Cost Rs.1200.00
Budget


Designing
Cost Rs.12000.00
Budget
Modularisation
Cost Rs.4800.00
Budget
Design Verification
Cost Rs.4800.00
Budget
UML Diagrams
Cost Rs.2400.00
Budget
Class
Cost Rs.600.00
Budget
Activity
Cost Rs.600.00
Budget
DFD
Cost Rs.1200.00
Budget

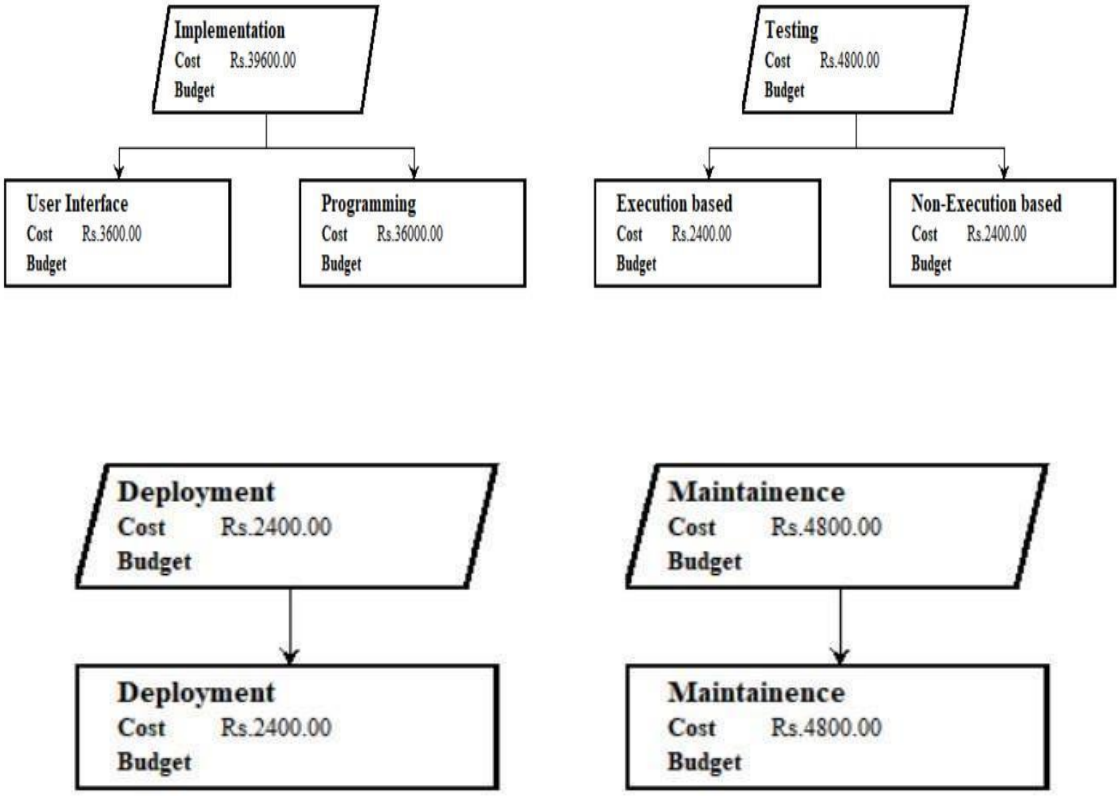


Fig 4.6: WBS Chart

Chapter 5

Report on Present Investigation

5.1 Proposed System

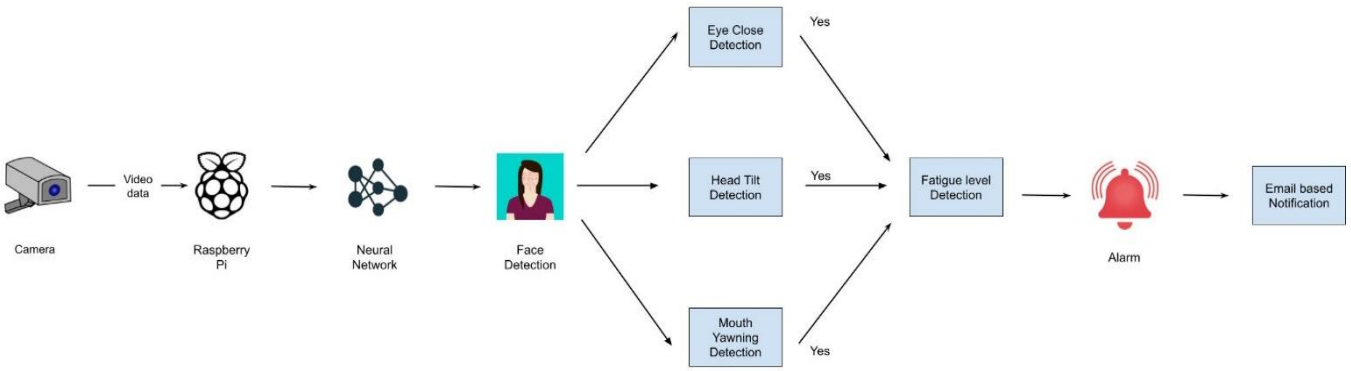


Fig 5.1 Block Diagram of Proposed System

The block diagram of proposed system for detecting fatigue and sleepiness utilizing Deep learning and IoT technologies. In Fig 5.1 a driver can activate the system simply by turning on the power. When the system is turned on, the camera's input is transmitted to the CNN algorithm in Python file. Where the person's face is processed by the corresponding libraries, and every facial feature in the diagram is compared to its threshold value and a prediction of tiredness is made using a series of if else statements and the counter variable count. Based on the counter value, an email is sent to the concerned person or organisation who is linked to the driver for alertness. The key components of the proposed system are as follows:

Raspberry Pi 4 Model B: A Raspberry Pi is single-board based computer that functions as the micro-computer which has Raspbian operating system built in.

Speaker: To produce sound to alarm the driver.

Web Camera: A webcam is a little digital recording device videos with static photos at 720p 30fps. that connects to the Raspberry Pi. It captures high-definition.

GPS Module: The GPS (Global Positioning System) is satellite-based system that uses satellites and ground control station to accurately measure and track its geographical position on earth surface.

The driver's involvement with the system is enabled by the camera, which sends the visual stream to the Raspberry Pi microcontroller. Raspberry Pi is a small microprocessor with its own operating system that performs the same functions as a computer.

Proposed system is basically divided into four major parts:

- **Detection of yawning**: The trained model recognizes the driver's mouth, marks points while the mouth is closed, and calculates the person's tiredness based on the points and mouth open ratio. The approach helps in calculating clear prediction as the value obtained could be directly compared against the threshold value.

$$\text{YAWN Ratio} = \frac{\text{Upper-Lower Lip distance}}{\text{Distance between mouth corners}}$$

- **Detection of head position**: The Trained Model recognizes the drivers head based position through length of nose and compares with average length of nose for detection.
- **Alarming System:** Dependent on the driver's level of exhaustion, the sound system uses a microcontroller to alert the driver to take specific actions based on the amount of fatigue. In the event of an accident, an e-mail is sent to the appropriate employees as part of the alarming procedure.

Chapter 6
Technologies Used

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is a subset of machine learning (ML), which is itself a subset of artificial intelligence (AI). Deep learning is an important element of data science, which includes statistics and predictive modelling. It is extremely beneficial to data scientists who are tasked with collecting, analysing and interpreting large amounts of data; deep learning makes this process faster and easier. At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.) It has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. It can be used on a server to create web applications. It can connect to database systems. It can also read and modify files. Python can be used for rapid prototyping, or for production-ready software development. It runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-oriented way or a functional way.

Machine learning (ML) is a type of artificial intelligence that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine Learning (ML) manages to influence user's machines to gain from the external environment. This external environment can be sensors, electronic segments, external storage gadgets, and numerous other devices. Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is

so pervasive today that you probably use it dozens of times a day without knowing it.

Dlib is an open source C++ library implementing a variety of algorithms, including classification, regression, clustering, data transformation, and structured prediction. *Dlib* is a toolkit containing *machine learning* algorithms and tools for creating complex software in C++ to solve real world problems.

Convolutional Neural Network- In Deep Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image. In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyse visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution.

Deep Neural Networks (DNN): It consists of multilayer perceptron with many hidden layers. The weights are completely linked and after using supervised or unsupervised pretraining method, the weights are initialized.

Chapter 7

Implementation

7.1 Algorithms used

HAAR CASCADE ALGORITHM

HAAR cascade are very effective for object detection which is based on Viola-Jones Algorithm. This approach involves a lot of positive and negative images that are used to train the classifier. There are many features on the face, sum of pixels under white and black rectangles, so we can detect the feature even in low image quality. For training the images the threshold is calculated using the contrast of lighter to darker region. We select certain features for minimum error rate i.e. choosing such facial features which could help easily classify face and non-face image. As most of the part of image will be non-face a different classifier stages are used to identify based on which the face of the person is recognized if it passes through all the stages of the classifiers. In figure 1. A Haar kernel is a rectangle with all of the light pixels below and all of the dark pixels on the upper side of the rectangle. The difference between the average of the pixel values in the darker zone and the average of the pixel values in the lighter region is used to calculate the Haar. If the difference is close to one, then there is a edge detected by Haar.

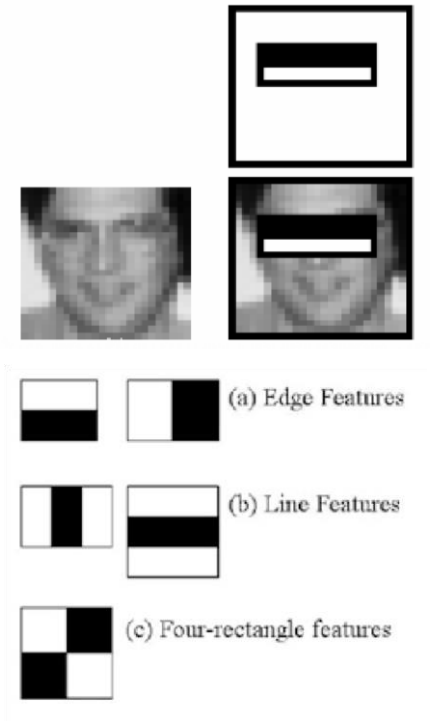


Fig. 7.1: HAAR cascade Algorithm

Every feature of a single value is calculated by taking subtraction of total pixels beneath white rectangle from total number of pixels beneath black rectangle.

Dlib library based on CNN : Deep Learning techniques is subfield of Machine Learning that is driven by the human brain. Dlib is a C++ deep learning library which uses CNN as its base algorithm. CNNs, also termed as ConvNets, are multilayered neural networks that are primarily use in image processing, video analysis as well as object detection. To execute the convolution process, CNN has convolution layer with several filters. The CNN has responsibility of taking a picture whether the eye is opened or closed.

Facial Detection & Recognition:

Implementation of parameters for facial landmark detection is done in the planned work to identify the driver's state. 68 predetermined landmarks for predicting shape and identifying facial parts such as the eye, mouth, head and so on. Many differences in the dimensions of the distinguishing points reflect the person's diverse expressions.

Proposed system is basically divided into four major parts:

Fig. 3: Facial landmark points

Drowsiness and tiredness are detected using pre-existing features for face landmark recognition. We utilized the open source Python library dlib to recognize 68 facial landmarks. These predetermined annotations assist in form of prediction based recognition of numerous face region parts such as eyes,

nose, eyebrows, mouth and others shown in above Fig.3. The person's numerous expressions are represented by variations in the set parameters of these separate points.

These are the following landmark coordinates represented by the above coordinates referred from

Fig .3.: Table 1: Various facial region coordinates

Sr. no.	Facial Points	Landmark Coordinates
1)	Left Eye Region	[37, 42]
2)	Right Eye Region	[43, 48]
3)	Nose Region	[28, 36]
4)	Mouth Region	[49, 68]

The procedure for recognizing a facial landmark is as follows:

High-resolution cameras are used to monitor and capture images in process of extracting frames and then generate alerts. Each captured frame is evaluated to examine the pattern of features of the face, and EAR (Eye Aspect Ratio) and MAR (Mouth Aspect Ratio) at each frame is calculated using Haar Cascade Classifier. A blink and a yawn are considered when the Eye ratio and Mouth ratio values reach at their specific threshold levels. If eye gaze and yawns are detected for particular frames in a sequence, then system informs the driver via playing an alarm. The alarm is set off to get attention of driver and will continue to ring until driver gets wakes up.

A video is captured with a webcam and the frames are retrieved on a laptop. Following that, we have used image processing techniques such as open CV and dlib to extract the image. The video is recorded as soon as the driver sits in front of the web cam. Then eye closure, blinking, head tilt, and yawning are detected.

Eye Detection:

6 coordinates are marked on the eyes using the dlib landmarks predictor function, starting from the left corner and moving clockwise. The formula to obtain EAR (Eye Aspect Ratio), which is the height-to-width ratio of the eye.

The EAR value is calculated using,

$$EAR = \frac{||d1 - d3|| + ||d2 - d4||}{2\,||d0 - d5||}$$

Where, d1, d2, d3, d4, d5, d6 are 6 facial landmark points to be extracted for eyes and $||di - dj||$ is Euclidean distance between points i and j. We determine whether the driver seems

drowsy or not by monitoring eye closure and eye opening ratios using the above formula. EAR is often calculated independently for left and right eyes.

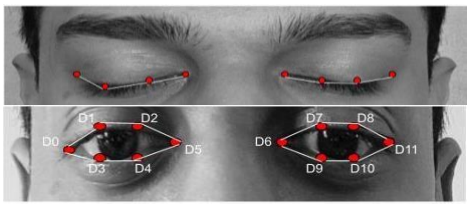


Fig. 4: Eye open and close coordinates

Mouth Detection:

The MAR (Mouth Aspect Ratio) is metric that determines the of wide your mouth. The mouth is represented by an 8coordinate pair. The face landmarks are designated in clockwise manner, beginning at the left corner of your mouth. The MAR is obtained by taking the vertical distance between lower and upper lips by the horizontal distance between the lip corners.

$$MAR = \frac{||d2 - d8|| + ||d3 - d7|| + ||d4 - d6||}{2||d1 - d5||}$$

Whenever driver yawns, the gap between your lower and upper lips widens. The yawn count is increased when the MAR value surpasses a specified level.

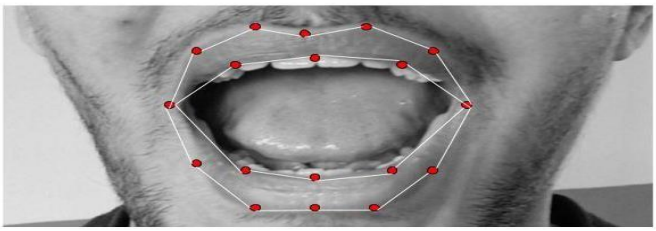


Fig. 5: Open mouth coordinates

Head Tilt Detection:

Drowsiness causes the driver's head to tilt forward, sideways or backward in relation to vertical axis. As a result, driver drowsiness can also be estimated based on the head bending angle. Since the simulated length of the nose on camera focal plane is proportional to the tilting, it could be used for determining head bending. Under normal circumstances, nose forms acute angle with the camera's focal plane. As the head travels upward, the angle increases, and as it travels downwards, it decreases. As a result, the ratio of the nose length to average of the nose length whereas awakened is measured for head bending, and if value

is larger or less than specific range, it indicates both bending of head and drowsiness. The length of nose is calculated by using facial landmarks and defined as

$$NLR = \frac{\text{nose length } (d31 - d28)}{\text{average nose length}}$$

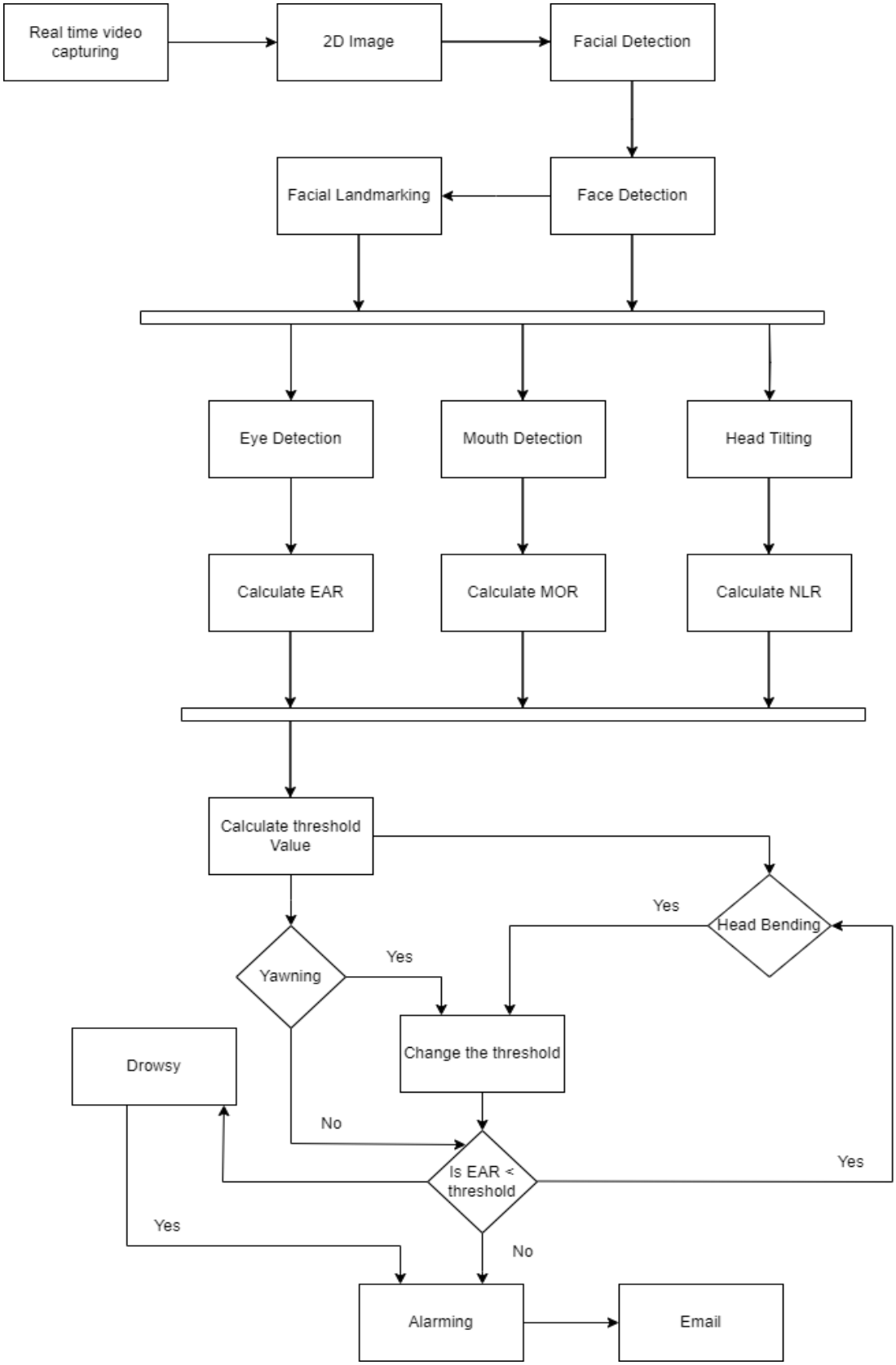


Fig.7 Block Diagram

Chapter 8

Code

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import os
from pygame import mixer
mixer.init()
sound = mixer.Sound('alarm.wav')


def alarm(msg):
    global alarm_status
    global alarm_status2
    global saying
    while alarm_status:
        print('call')
        s = 'espeak "()"'.format(msg)
```

```
            os.system(s)
        if alarm_status2:
            print('call')
            saying = True
            s = 'espeak "' + msg + '"'
            os.system(s)
            saying = False
    def eye_aspect_ratio(eye):
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        C = dist.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear
    def final_ear(shape):
        (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
        (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        return (ear, leftEye, rightEye)
    def lip_distance(shape):
        top_lip = shape[50:53]
```

```
    top_lip = np.concatenate((top_lip, shape[61:64]))
    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))
    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
    distance = abs(top_mean[1] - low_mean[1])
    return distance
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0,help="index of webcam on system")
args = vars(ap.parse_args())
EYE_AR_THRESH = 0.25
EYE_AR_CONSEC_FRAMES = 30
YAWN_THRESH = 30
alarm_status = False
alarm_status2 = False
saying = False
COUNTER = 0
print("-> Loading the predictor and detector...")
#detector = dlib.get_frontal_face_detector()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")    #Faster but less accurate
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
print("-> Starting Video Stream")
vs = VideoStream(src=args["webcam"]).start()
#vs= VideoStream(usePiCamera=True).start()       //For Raspberry Pi
```

```
time.sleep(1.0)


while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    #rects = detector(gray, 0)
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
                    minNeighbors=5, minSize=(30, 30),
                    flags=cv2.CASCADE_SCALE_IMAGE)
    #for rect in rects:
    for (x, y, w, h) in rects:
        rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye [1]
        rightEye = eye[2]
        distance = lip_distance(shape)
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

```
lip = shape[48:60]
cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)
if ear < EYE_AR_THRESH:
    COUNTER += 1
    if COUNTER >= EYE_AR_CONSEC_FRAMES:
        if alarm_status == False:
            alarm_status = True
            sound.play()
            t = Thread(target=alarm, args=('Wake Up Sir',))
            t.deamon = True
            t.start()
        cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
else:
    COUNTER = 0
    alarm_status = False
if (distance > YAWN_THRESH):
    sound.play()
    cv2.putText(frame, "Yawn Alert", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    if alarm_status2 == False and saying == False:
            alarm_status2 = True
            t = Thread(target=alarm, args=('Take Some Fresh Air sir',))
```

```
                t.deamon = True
                t.start()
        else:
            alarm_status2 = False
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
cv2.destroyAllWindows()
vs.stop()
```

Dataset:

The dataset contains 80,000 images divided into train and test data where images indicate different facial features of eye open and close pictures, mouth opening and closing and lastly head tilt feature. The images are converted to grayscale for the algorithm to easily identify the landmarks.

Fig. 8.2: Dataset

The dataset consists of 80000 images which consists of drowsy and non-drowsy images as shown in fig 6 .The image in the dataset is separated using indexing that is 0 for drowsy person and 1 for non-drowsy person. To perform this experimentation total 45600 images are used for training out of which 22800 are drowsy images and 22800 are non-drowsy images .Here 5000 images are used for validation in which 2500 images are drowsy and 2500 images are non-drowsy. Here 11000 images are used for testing the model out of which 5000 images are drowsy and 6000 images are non - drowsy. Currently the proposed model has acquired 96% accuracy after 16 epochs.

Training Images	45600
Validation Images	5000
Testing Images	1100
Training Accuracy (%)	98
Validation Accuracy (%)	97
Testing Accuracy (%)	96

Fig 8.3

8.4 Testing:

Convolutional Neural Network

The first convolutional layer comprises of 32 filters and has the size of 3x3 and accepts the input of image with the size 510x510. The output is then provided to pooling layer which has filters of the size 2x2. The maximum value among the filtered value is chosen as this is the max-pooling operation. The second convolutional layer consists of 32 filters of size 3x3. The output thus obtained is again fed into a pooling layer which consists of input size 126x126 and filters of size 2x2. The third convolutional layer contains image with input size as 124x124 and 32 filters of size 3x3. The output is again fed into pooling layer which consists of input size 62x62 and filter of size 2x2. The ReLU layer is a rectified Layer Unit that performs the rectifying operation to rectify the negative pixel values. The fully connected layer classifies the pixels into one of the 5 different classes.

```
def create_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same', activation='relu', inp
ut_shape=inputShape))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(NUM_CLASSES, activation='softmax'))
    opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
    model.compile(loss="categorical_crossentropy", optimizer=opt,
metrics=["accuracy"])
    return model
```

Fig 8.4.1: Convolutional Neural Network

```
In [38]: 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)
```

```
Epoch 1/5
8032/8032 [==============================] - ETA: 0s - loss: 0.1712 - accuracy: 0.9634
Epoch 00001: val_loss improved from inf to 0.20740, saving model to
C:\Users\ASUS\Desktop\Final\implemented_drowsy\models\model.h5
8032/8032 [==============================] - 405s 50ms/step - loss: 0.1712 - accuracy: 0.9634 - val_loss: 0.2070 -
val_accuracy: 0.9506 - lr: 0.0010
Epoch 2/5
 237/8032 [..............................] - ETA: 5:37 - loss: 0.1519 - accuracy: 0.9688 ETA: 5:35 - loss: 0.1487 -
```

Fig 8.4.2 Model Accuracy

Figure 8.4.1 shows the training model using Convolutional Neural Network that can took input images from dataset to classify it into one of the five classes with multiple convolutional and pooling layers is designed. The model is trained over 16 epochs.

```
compiling model...
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 128, 128, 32)      896
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 126, 126, 32)      9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 63, 63, 64)        18496
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 61, 61, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
dropout_5 (Dropout)          (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 30, 30, 64)        36928
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               6423040
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 2565
=================================================================
Total params: 6,565,029
Trainable params: 6,565,029
Non-trainable params: 0
_________________________________________________________________
```

Figure 8.4.3. Compiling the model

Figure 8.4.3 displays the contents of model after the model is “built”. To display , we call keras summary() method.

The training loss and validation loss is shown in fig 8.4.4The training accuracy and validation accuracy is shown in fig 8.4.5

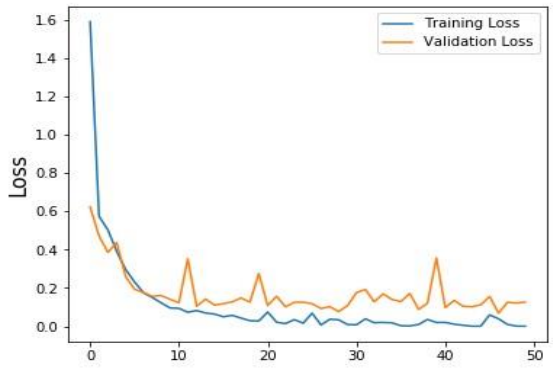


Fig 8.4.4. Validation and training accuracy loss against no of epochs.

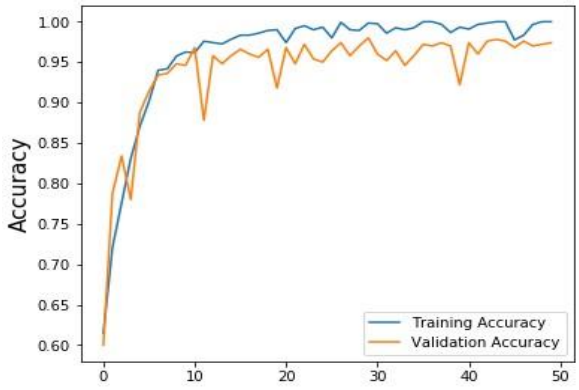


Fig 8.4.5 . Validation and training accuracy against no of epochs.

Chapter 9
RESULT AND DISCUSSION

- We have Built our own dataset which is more realistic in the current scenario.
- The Dataset is divided into training dataset (70%) and testing dataset (30%) and the model is trained on the training dataset.
- The accuracy of the result is high.
- As the result of the execution of the code and the output using Open CV and dlib we have identified the the facial features i.e eyes and mouth in real time and if the eyes is closed and it is alarmed through buzzer.

Result 9.1

A system is built with a single camera and microcontroller and alarm which is compact as some of the methods previously in use were able to cause distraction to the driver, where the drivers were vulnerable to an accident.

Fig. 9.1: Prototype

Result 9.2

Upon activating system, the drivers face is recognized and the points are marked, if the EAR ratio of eye is lower than 0.25 unit which is the threshold value displayed in Fig 9.2 then the driver is alarmed

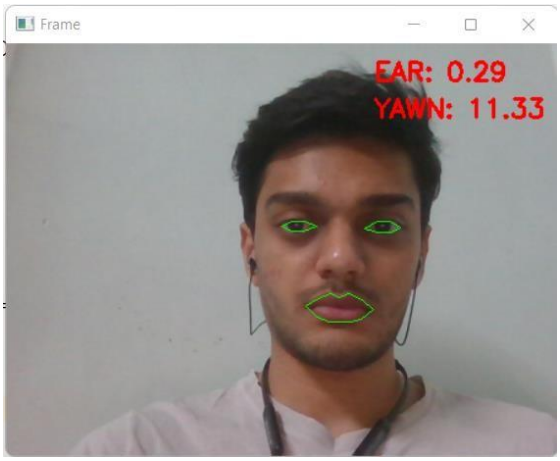


Fig. 9.2: Score when eyes are open and subject is not yawning

Result 9.3

If the Mouth Open Ratio of the of the driver's face id found to be greater than 20 units the driver is alarmed with an alarm and with an appropriate message.

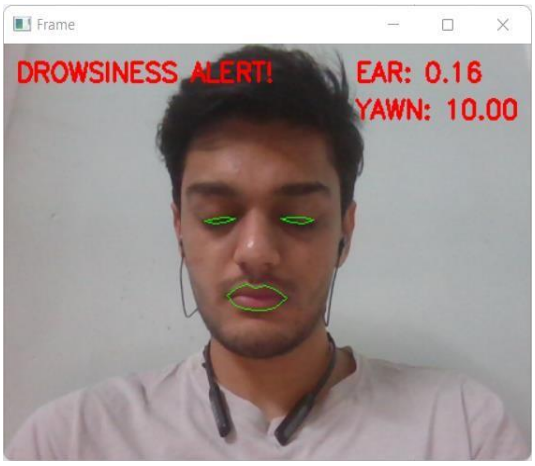


Fig. 9.3: Score when eyes are closed (Alarming Situation)

Result 9.4

With all of the behavioral aspects combined in a single system based on the literature survey performed as all the proposed systems in the paper dealt with one or other signs of drowsiness but the combined result of all the aspects helps in proposing better alarming system which is developed in this paper.

Fig. 9.4: Alert using Email

If the driver offend the drowsiness alert multiple time consecutively then an alert email is delivered to the concerned person alerting him of the drowsy driver and providing with geolocation of the vehicle.

The model trained over the dataset images yields testing accuracy of 96.34% where model is tested on approximate 2000 images.

Chapter 10
CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

The system designed achieves the objective of detecting drowsy driver and notifies the driver with alarming signal and appropriate audio message. With the help of CNN algorithm and Haar cascade files faster detection of face and extraction of facial features according to which the model predicts the mental state of the driver through which his behavioral features, include eye blinking, yawning and head tilt detection. With the help of deep learning technology even if the size is small, we get high accuracy, with the algorithm in place, the accuracy of 96% is achieved. As system is compact, and can be easily integrated in modern cars as it just requires camera and raspberry pi module. In case of consecutive violation of drowsiness threshold an email is sent to the concerned person with a message and current location of the driver is shared to that person. The systems accuracy gets little compromised in highly dark conditions.

10.2 Future Scope

- Additional Features such as adaptive cruise control could be added as a safety feature
- Capture individual driver's steering activity while drowsy.
- Conduct additional simulator experiments to validate the algorithm, test additional road conditions, and test a more diversified group of drivers.

Reference

1) Jongseong Gwak, Motoki Shino and Akinari Hirao, “Early Detection of Driver Drowsiness Utilizing Machine Learning based on Physiological Signals, Behavioral Measures, and Driving Performance,” 21st International Conference on Intelligent Transportation Systems (ITSC) Maui, Hawaii, USA, November 4-7, 2018
2) Janki Chandiwala and Shrushti Agarwal, ”Driver’s real-time Drowsiness Detection using Adaptable Eye Aspect Ratio and Smart Alarm System,”7th International Conference on Advanced Computing & Communication Systems (ICACCS), 2021.
3) Shruti Mohanty, Shruti Hegde, Supriya Prasad, J. Maniknadan, “Design of Real-Time Drowsiness Detection System using Dlib,” 5th IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), 2019.
4) A. f.m saifuddin saif, Zainal rasyid mahayuddin, “Robust Drowsiness Detection for Vehicle Driver using Deep Convolutional Neural Network,” International Journal on Advanced Computer Science and Applications (IJACSA), Vol.11, 2020.
5) Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa AlKhalifa, Moez Krichen, Kamel Barkaoui, “Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application,” IEEE Conference on Informatics, IoT and Enabling Technologies (ICIoT), 2020.
6) Anshul Pinto, Mohit Bhasi, Durvesh Bhalekar, Pradyoth Hegde, Shashidhar G. Koolagudi, “A Deep Learning Approach to Detect Drowsy Drivers in Real Time,” IEEE 16th India Council International Conference (INDICON), 2019.
7) Menchie Miranda, Alonica Villanueva, Mark Jomar Buo, Reynald Merabite, Sergio Paulo Perez, and John Michael Rodriguez, “Portable Prevention and Monitoring of Driver’s Drowsiness Focuses to Eyelid Movement using Internet of Things,” IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management, 2018.
8) Fouzia, R. Roopalakshmi, Jayantkumar A. Rathod, Ashwitha S. Shetty, K. Supriya, “Driver Drowsiness Detection System Based on Visual Features,” Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018.
9) Mariella Dreißig, Mohamed Hedi Baccour, Tim Schäck, Enkelejda
10) Kasneci (2020). “Driver Drowsiness Classification Based on Eye Blink and Head Movement Features Using the k-NN Algorithm,” IEEE Symposium Series on Computational Intelligence (SSCI), 2020
11) Kim, W., Choi, H.-K., & Jang, B.-T, “Study on Training Convolutional Neural Network to Detect Distraction and Drowsiness,” IEEE Region Ten Symposium (Tensymp), 2018.
12) Lin, J. (2020). “Integrated Intelligent Drowsiness Detection System Based on Deep Learning,” IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), 2020.
13) Patnaik, R., Krishna, K. S., Patnaik, S., Singh, P., & Padhy, N., “Drowsiness Alert, Alcohol Detect and Collision Control for Vehicle Acceleration,” International Conference on Computer Science, Engineering and Applications (ICCSEA), 2020.

14) Wanghua Deng, Ruoxue Wu (20, “Real-Time Driver-Drowsiness Detection System Using Facial Features,” IEEE Access, 2019

15) Chaoyun Zhang, Rui Li, Woojin Kim, Daesub Yoon,And Paul Patras, “Driver Behavior Recognition via Interwoven Deep Convolutional Neural Nets With Multi-Stream Inputs,” Electronics and Telecommunications Research Institute (ETRI), Daejeon, South K, 2020.

16) Bhargava Reddy, Ye-Hoon Kim, Sojung Yun, Chanwon Seo, Junik Jang, “Real-Time Driver Drowsiness Detection for Embedded System Using Model Compression of Deep Neural Networks, ” IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017