DRIVER DROWSINESS DETECTION

A MINI PROJECT REPORT

Submitted by

PRANEET KUMAR B (312419104095)

PRASATH KUMAR R (312419104096)

in partial fulfilment for the requirement of award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

St. JOSEPH'S INSTITUTE OF TECHNOLOGY

CHENNAI - 119

ANNA UNIVERSITY: CHENNAI 600 025

JUNE-2022

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this mini project report "**DRIVER DROWSINESS DETECTION"** is the bonafide work of "**PRANEET KUMAR B (312419104095)** and **PRASATH KUMAR R (312419104096)"** who carried out the project under my supervision.

SIGNATURE

Dr. J. DAFNI ROSE M.E, Ph.D.,
Professor,
Head of the Department,
Computer Science and Engineering,
St. Joseph's Institute of Technology,
Old Mamallapuram road,
Chennai – 600119.

SIGNATURE

Ms. A. ANGELINE VALENTINA SWEETY, M.E.,
Assistant Professor,
Computer Science of Engineering.
St. Joseph's Institute of Technology,
Old Mamallapuram road,
Chennai – 600119.

ACKNOWLEDGEMENT

We also take this opportunity to thank our honourable Ch airman **Dr. B. Babu Manoharan, M.A., M.B.A., Ph.D.** for the guidance he offered during our tenure in this institution.

We extend our heartfelt gratitude to our honourable Director **Mrs. S. Jessie Priya, M.com.,** for providing us with the required resources to carry out this project.

We express our deep gratitude to our honourable CEO **Mr. B. Sashi Sekar, M.Sc (INTL.Business)** for the constant guidance and support for our project.

We are indebted to our Principal **Dr. P. Ravichandran, M.Tech., Ph.D.** for granting us permission to undertake this project.

Our earnest gratitude to our Head of the Department **Dr. J. Dafni Rose, M.E., Ph.D.,** for her commendable support and encouragement for the completion of the project with perfection.

We express our profound gratitude to our guide **Ms. A.Angeline Valentina Sweety, M.E.,** for her guidance, constant encouragement, immense help and valuable advice for the completion of this project.

We wish to convey our sincere thanks to all the teaching and non-teaching staff of the department of **COMPUTER SCIENCE AND ENGINEERING** without whose co-operation this venture would not have been a success.

CERTIFICATE OF EVALUATION

College Name : St. JOSEPH'S INSTITUTE OF TECHNOLOGY

Branch : COMPUTER SCIENCE AND ENGINEERING

Semester :VI

Sl. No	Name of the Students	Title of the Project	Name of the Supervisor with Designation
1 2	**PRANEET KUMAR B (312419104095)** **PRASATH KUMAR R (312419104096)**	**DRIVER DROWSINESS DETECTION**	**Ms. A ANGELINE VALENTINA SWEETY M.E.,** **Assistant Professor,** **St. Joseph's Institute Of Technology**

The report of the project work submitted by the above students in partial fulfilment for the award of Bachelor of Engineering Degree in **Computer Science and Engineering** of Anna University were evaluated and confirmed to be report of the work done by above students.

(INTERNAL EXAMINER) **(EXTERNAL EXAMINER)**

ABSTRACT

Many of the accidents occur due to drowsiness of drivers. It is one of the critical causes of roadways accidents now-a-days. Latest statistics say that many of the accidents were caused because of drowsiness of drivers. Vehicle accidents due to drowsiness in drivers are causing death to thousands of lives. More than 30% accidents occur due to drowsiness. For the prevention of this, a system is required which detects the drowsiness and alerts the driver which saves the life. In this project, we present a scheme for driver drowsiness detection. In this, the driver is continuously monitored through webcam. Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving. The objective of this intermediate Python project is to build a drowsiness detection system that will detect that a person’s eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected. 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.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NUMBER
	ABSTRACT	**v**
	LIST OF FIGURES	**viii**
1.	**INTRODUCTION**	**1**
	1.1 OVERVIEW	1
	1.2 PROBLEM STATEMENT	2
	1.3 EXISTING SYSTEM	2
	1.4 PROPOSED SYSTEM	2
2.	**LITERATURE REVIEW**	**5**
3.	**SYSTEM DESIGN**	**12**
	3.1 UML FLOW DIAGRAMS	12
	3.1.1 USE CASE DIAGRAM	12
	3.1.2 CLASS DIAGRAM	14
	3.1.3 SEQUENCE DIAGRAM	15
	3.1.4 ACTIVITY DIAGRAM	16
	3.1.5 STATE CHART DIAGRAM	17
	3.1.6 COMPONENT DIAGRAM	18
	3.1.6 DEPLOYMENT DIAGRAM	19
4.	**SYSTEM ARCHITECTURE**	**20**
	4.1 ARCHITECTURAL DESIGN	20
	4.2 ARCHITECTURE DESCRIPTION	
	4.3 MODULES	21
	4.3.1 FACE DETECTION MODULE	21
	4.3.2 EYE DETECTION MODULE	22
	4.3.3 FACE TRACKING MODULE	22
	4.3.4 EYE TRACKING MODULE	22
	4.3.5 DETECTION MODULE	22

5. SYSTEM IMPLEMENTATION 23

5.1 IMPLEMENTATION OF DRIVER DROWSINESS DETECTION **23**

6. CODING AND SCREENSHOTS 26

6.1 CODE FOR DROWSINESS DETECTION 26

6.2 OUTPUT AND COMPARISON 31

7. CONCLUSION AND FUTURE WORK 34

7.1 CONCLUSION 34

7.2 FUTURE WORK 34

8. REFERENCES 35

LIST OF FIGURES

LIST OF FIGURES	NAME OF THE FIGURE	PAGE NO
3.1.1	Use Case Diagram of Driver Drowsiness Detection	12
3.1.2	Class Diagram of Driver Drowsiness Detection	14
3.1.3	Sequence Diagram of Driver Drowsiness Detection	15
3.1.4	Activity Diagram of Driver Drowsiness Detection	16
3.1.5	State Chart Diagram of Driver Drowsiness Detection	17
3.1.6	Deployment Diagram of Driver Drowsiness Detection	19
4	System Architecture	20
4.1	Architecture Design	20
6.1	Sample Code	26
6.2	Sample Output	31

CHAPTER 1

INTRODUCTION

Drowsiness of the drivers is one of the key issues for majority of road accidents. Drowsiness threatens the road safety and causes severe injuries sometimes, resulting in fatality of the victim and economical losses. Drowsiness implies feeling lethargic, lack of concentration, tired eyes of the drivers while driving vehicles. Most of the accidents happen in India due to the lack of concentration of the driver. Performance of the driver gradually deteriorates owing to drowsiness. To avoid this anomaly, we developed a system that is able to detect the drowsiness nature of the driver and alert him immediately. This system captures images as a video stream through a camera, detects the face and localizes the eyes. The eyes are then analysed for drowsiness detection using CNN algorithm. Based on the result, the driver is alerted for drowsiness through an alarm system.

1.1 OVERVIEW

Now-a-days, there is huge increase in private transportation day by day in this modernize world. It will be tedious and bored for driving when it is for long time distance. One of the main causes behind the driver’s lack of alertness is due to long time travelling without sleep and rest. Tired driver can get drowsy while driving. Every fraction of seconds drowsiness can turn into dangerous and life-threatening accidents may lead to death also. To prevent this type of incidents, it is required to monitor driver’s alertness continuously and when it detects drowsiness, the driver should be alerted. Through this we can reduce significant number of
accidents and can save lives of people.

1.2 PROBLEM STATEMENT

Many of the road accidents will occur due to drowsiness of the driver. Drowsiness can be detected by monitoring the driver through continuous video stream with a mobile or camera. The general objective is to create a model that will indicate whether a person is feeling drowsy or not. The model takes image for every second and check for eye blinking and calculate the time for eye closed by CNN algorithm. If the blinking is high and eye is closed for certain amount of time then it will indicate driver through a sound.

1.3 EXISTING SYSTEM

The current drowsiness detection systems include the usage of the devices that detect the respiration rate, heart rate, blood pressure, etc. These devices can cause the driver to be uncomfortable for driving. Cannot be assured that the drivers wear these devices all the time while driving. May get lost or improper functioning which may lead to low accuracy in the result. The existing system does not produce good results in low light conditions. If the light conditions are dark or too low it is unable to detect the face and eyes of the driver which results in lower accuracy.

1.4 PROPOSED SYSTEM

First of all, the system captures images through the webcam and after capturing it detects the face through Haar cascade algorithm. It uses Haar features which can detect the face. If the system founds it as face the it will proceed for next phase i.e., eye detection. The eye is also detected using Haar cascade features and it is used for blink frequency. The state of eye will be detected using CNN algorithm. Through this algorithm we can find the percentage of time the eye lids remain closed. If it found eyes in closed state then it detects driver in drowsy state and alerts him by an alarm. In some cases, distraction can be measured by continuous gazing. The driver’s face is analyzed continuously to detect any distraction. If found then alarm is activated by the system. The biggest advantage of the proposed model is the model is compressed small enough that it can be deployed on an embedded board while preserving reasonable accuracy.

1.4.1 ALGORITHM

Face detector (Haar classifier): It is an object detection application. It is based on a smart use of boosting. A trained frontal face detector is available with the OpenCV distribution. This works remarkably well. We can train the algorithm for other objects by using the software provided. This works wonderfully for rigid objects with characteristic views.

This algorithm comprises of four stages:

1. **Haar Feature Selection**

   The first step is to collect the Haar features. A **Haar feature** is essentially calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums.

2. **Creating Integral Images**

   integral images essentially speed up the calculation of these Haar features. Instead of computing at every pixel, it instead creates sub-rectangles and creates array references for each of those sub-rectangles. These are then used to compute the Haar features.

3. **Adaboost Training**

   Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination of **“weak classifiers”** to create a **“strong classifier”** that the algorithm can use to detect objects.

4. **Cascading Classifiers**

   The cascade classifier is made up of a series of stages, where each stage is a collection of weak learners. Weak learners are trained using boosting, which allows for a highly accurate classifier from the mean prediction of all weak learners. Though Haar Cascade is used for detecting almost all objects, it is popular for detecting faces in images.

Convolution Neural Network : A convolutional neural network (CNN or ConvNet), is a network architecture for deep learning which learns directly from data, eliminating the need for manual feature extraction.
CNNs are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They can also be quite effective for classifying non-image data such as audio, time series, and signal data.

Applications that call for object recognition and computer_vision — such as self-driving vehicles and face-recognition applications — rely heavily on CNNs

Uses

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features”

Some Haar cascade benefits are that they're very fast at computing Haar-like features due to the use of integral images (also called summed area tables). They are also very efficient for feature selection through the use of the AdaBoost algorithm.

CHAPTER 2
LITERATURE REVIEW

Hong Su et. al. [1] described ‘A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness’. They proposed a new technique of modelling driver drowsiness with multiple eyelid movement features based on an information fusion technique—partial least squares regression (PLSR), with which to cope with the problem of strong collinear relations among eyelid movement features and, thus, predicting the tendency of the drowsiness. The predictive precision and robustness of the model thus established are validated, which show that it provides a novel way of fusing multi-features together for enhancing our capability of detecting and predicting the state of drowsiness.

Bin Yang et. al. [2] described ‘Camera based Drowsiness Reference for Driver State Classification under Real Driving Conditions’. They proposed that measures of the driver’s eyes are capable to detect drowsiness under simulator or experiment conditions. The performance of the latest eye tracking based in-vehicle fatigue prediction measures are evaluated. These measures are assessed statistically and by a classification method based on a large dataset of 90 hours of real road drives. The results show that eye-tracking drowsiness detection works well for some drivers as long as the blinks detection works properly. Even with some proposed improvements, however, there are still problems with bad light conditions and for persons wearing glasses. As a summary, the camera-based sleepiness measures provide a valuable contribution for a drowsiness reference, but are not reliable enough to be the only reference.

M.J. Flores et. al. [3] described ‘Driver drowsiness detection system under infrared illumination for an intelligent vehicle’. They proposed that to reduce the amount of such fatalities, a module for an advanced driver assistance system, which caters for automatic driver drowsiness detection and also driver distraction, is presented.

Artificial intelligence algorithms are used to process the visual information in order to locate, track and analyse both the driver's face and eyes to compute the drowsiness and distraction indexes. This real-time system works during nocturnal conditions as a result of a near-infrared lighting system. Finally, examples of different driver images taken in a real vehicle at night-time are shown to validate the proposed algorithms.

A. Cheng et. al. [4] described 'Driver Drowsiness Recognition Based on Computer Vision Technology'. They presented a nonintrusive drowsiness recognition method using eye-tracking and image processing. A robust eye detection algorithm is introduced to address the problems caused by changes in illumination and driver posture. Six measures are calculated with percentage of eyelid closure, maximum closure duration, blink frequency, average opening level of the eyes, opening velocity of the eyes, and closing velocity of the eyes. These measures are combined using Fisher's linear discriminated functions using a stepwise method to reduce the correlations and extract an independent index. Results with six participants in driving simulator experiments demonstrate the feasibility of this video-based drowsiness recognition method that provided 86% accuracy.

G. Kong et. al. [5] described 'Visual Analysis of Eye State and Head Pose for Driver Alertness Monitoring'. They presented visual analysis of eye state and head pose (HP) for continuous monitoring of alertness of a vehicle driver. Most existing approaches to visual detection of non-alert driving patterns rely either on eye closure or head nodding angles to determine the driver drowsiness or distraction level. The proposed scheme uses visual features such as eye index (EI), pupil activity (PA), and HP to extract critical information on non-alertness of a vehicle driver. A support vector machine (SVM) classifies a sequence of video segments into alert or non-alert driving events. Experimental results show that the proposed scheme offers high classification accuracy with acceptably low errors and false alarms for people of various ethnicity and gender in real road driving conditions.

In June, Eyosiyas et. al. [6] described ‘Driver Drowsiness Detection through HMM based Dynamic Modelling’. They proposed a new method of analysing the facial expression of the driver through Hidden Markov Model (HMM) based dynamic modelling to detect drowsiness. They have implemented the algorithm using a simulated driving setup. Experimental results verified the effectiveness of the proposed method.

García et. al. [7] described ‘Driver Monitoring Based on Low-Cost 3-D Sensors’. They proposed a solution for driver monitoring and event detection based on 3-D information from a range camera is presented. The system combines 2-D and 3-D techniques to provide head pose estimation and regions-of-interest identification. Based on the captured cloud of 3-D points from the sensor and analysing the 2-D projection, the points corresponding to the head are determined and extracted for further analysis. Later, head pose estimation with three degrees of freedom (Euler angles) is estimated based on the iterative closest points algorithm. Finally, relevant regions of the face are identified and used for further analysis, e.g., event detection and behaviour analysis. The resulting application is a 3-D driver monitoring system based on low-cost sensors. It represents an interesting tool for human factor research studies, allowing automatic study of specific factors and the detection of special event related to the driver, e.g., driver drowsiness, inattention, or head pose.

Knapik and Cyganek et. al. [8] presented a novel approach for driver fatigue detection, based on yawning detection, using long-range infrared thermal imaging. A special dataset was created for this research. The system works as follows. First, images are acquired from a thermal video. Then, three cascaded detection modules are applied for the face area, eye corners, and yawn. Since the mouth area is sometimes hard to detect in thermal images, due to the temperature difference in that area, information about other face regions' relative temperatures is used to detect the yawn reflex. Thus, the authors used the eye corners as an indicator for yawning. Cold and hot thermal voxel sum methods were used to detect yawning. Finally, based on the proposed algorithm's results and assumed constraints, an alarm is initiated when fatigue is detected. The system showed accuracies of 71% for cold voxels detection and 87% for hot voxels detection.

Kiashari et al. [9] introduced a non-intrusive system that detects drowsiness using facial thermal imaging to analyse the driver's respiration signal. Thirty subjects participated in their study, which was conducted in a car simulator. A thermal camera was used to capture the driver's thermal images. From the obtained thermal signals, the standard deviation and mean of both the respiration rate and inspiration-to-expiration time ratio were calculated and used as input features, in order to train two machine learning classifiers, namely, support vector machine (SVM) and k-nearest neighbour (KNN). Both classifiers were able to detect drowsiness. However, SVM outperformed the KNN, with 90% accuracy, 85% specificity, 92% sensitivity, and 91% precision.

Khan et al. [10] proposed a real-time DDD system based on eyelid closure. The system was implemented on hardware that used surveillance videos to detect whether the drivers' eyes were open or closed. The system started by detecting the face of the driver. Then, using an extended Sobel operator, the eyes were localized and filtered to detect the eyelids' curvature. After that, the curvature's concavity was measured. Based on the measured concavity value, the eyelid was classified as open (concave up) or closed (concave down). If the eyes were deemed closed for a certain period, a sound alarm is initiated. The system used three datasets. The authors generated two of them, and the third was acquired from. The first dataset, which contained simple images, with a homogenous background, showed an accuracy of 95%. The second set, which included a complex benchmark image dataset, achieved an accuracy of 70%; the third one, which used two real-time surveillance videos, showed an accuracy that exceeded 95%.

Ouabida et al. [11] proposed a fast method for DDD that depends on an optical correlator to detect the eye and then estimates its state using optical correlation with a deformed filter. This method was the first to use a numerical simulation of the optical Vander Lugt correlator to detect the eye centre automatically. The proposed DDD method precisely estimates the eye's location and state (open or closed), using a specific filter in the Fourier plane of the optical Vander Lugt correlator. In this method, the eyes are initially detected in non-zoomed facial images. Using the simulated optical correlator, the eye state is estimated under different lighting, head orientations, and with or without eyeglasses. The researchers evaluated the proposed method on five international databases: FEI, ICPR, BioID, GI4E, and the second Strategic Highway Research Program results (SHRP2). Additionally, a group of correlation filters was proposed and designed to recognize eyes' states in noisy and cluttering environments. The proposed optical correlation, with a deformed eye filter, showed the best performance.

In this work, Maior et al. [12] developed a drowsiness detection method based on eye patterns monitored by video streams using a simple web camera. The method tracks the blinking duration using the EAR metric. The proportion between the eye's height and width is calculated to evaluate the EAR value. A high EAR value indicates that the eye is open, while a low value indicates that it is closed. The proposed method consists of three main parts: eye detection, EAR calculation and blink classification, and real-time drowsiness detection. An experiment was conducted to generate a training database. After obtaining the images from the web camera, the EAR values were calculated and stored for each frame. Then, a specific number of consecutive values were used as input for the machine learning algorithms. Drowsiness is detected if the blink duration is longer, compared to a standard blink. Three classification methods were employed: multilayer perceptron, random forest (RF), and SVM. Overall, SVM showed the best performance, with an average test accuracy of 94.9%.

In , Bamidele et al. [13] presented a nonintrusive DDD system, based on face and eye state tracking. The research utilized the NTHUDDD Computer Vision Lab's video dataset. The proposed system starts by acquiring and pre-processing the required data. Then, it extracts the targeted features, including the PERCLOS, maximum closure duration of the eyes, and blink frequency. The extracted features are then fed to various classifiers to decide whether they belong to a drowsy or awake person. These classifiers include KNN, SVM, logistic regression, and artificial neural networks (ANN). The final results revealed that the best models were the KNN and ANN, with accuracies of 72.25% and 71.61%, respectively.

Hashemi et al. [14] proposed a real-time DDD system based on the area of eye closure and use of the convolutional neural network (CNN). Three networks were introduced for eye closure classification: fully designed neural network (FD-NN), transfer learning in VGG16 (TL-VGG16), and transfer learning in VGG19 (TL-VGG19), with extra designed layers. The authors used the ZJU gallery dataset, in addition to 4157 new images. The experiment resulted in the following network accuracies: 98.15%, 95.45%, and 95%, respectively.

Zandi et al. [15] proposed a non-intrusive drowsiness detection ML system based on eye-tracking data. The experiments were conducted in a simulated driving environment, with 53 participants. The authors collected data for eye-tracking signals and multichannel electroencephalography signals. The electroencephalography signal was only used as a reliable baseline for comparison and to label the eye-tracking signals epochs as drowsy or alert. The proposed ML system extracted 34 eye-tracking signals' features, obtained from overlapping eye signals' epochs with different lengths. The system performance, subject to various combinations of different features and epoch lengths, was also studied. Two binary classifiers were used: the RF classifier with 200 trees and non-linear SVM with a Gaussian kernel classifier. The experiment results revealed that the RF classifiers resulted in an accuracy range of 88.37% to 91.18% across all epochs, as well as a sensitivity–specificity of 88.1% to 88.8% for a 10-s epoch. In contrast, the non-linear SVM classifier showed an accuracy range of 77.12% to 82.62%. Additionally, it resulted in a sensitivity–specificity of 79.1% to 80.8% for a 10-s epoch. Using eye-tracking data and a proper classification framework, such results confirmed that drowsiness could be reliably detected with high accuracy, specificity, and sensitivity

Phanikrishna et al. [16] designed an automatic classification model for detecting the drowsiness of the driver using wavelet packet transform. The wavelet packet transform was extracted from the single channel Electro-Encephalogram (EEG) signals from the driver. The proposed model yields 94.45% of accuracy in performing the real-time sleep analysis. Taherisadr et al. (2018) designed a model for identifying the attention of the driver using Mel-Frequency Cepstrum in the two-dimensional transform and Convolution Neural Network (CNN). The designed model extracts the two-dimensional Mel-Frequency Cepstrum representation of the Electrocardiogram (ECG) sensed from the driver. The analytical results yield that the designed model is more efficient than the existing methodologies of drowsiness detection during driving. Lee et al. (2017) have designed a system that performs correlation analysis of Electrocardiography (ECG) and Photoplethysmogram (PPT) data for detecting the drowsiness of the driver. This model is a noise replacement model and the experimental analysis proves that the Noise replacement model is better efficient than the PPT method of detecting the driver's drowsiness.

CHAPTER 3

SYSTEM DESIGN

In this chapter, the various UML diagrams for the DRIVER DROWSINESS DETECTION is represented and the various functionalities are explained.

3.1 UNIFIED MODELLING LANGUAGE

Unified Modelling language (UML) is a standardized modelling language enabling developers to specify, visualize, construct and document artifacts of a software system. Thus, UML makes these artifacts scalable, secure and robust in execution. It uses graphic notation to create visual models of software systems. UML is designed to enable users to develop an expressive, ready to use visual modelling language. In addition, it supports high-level development concepts such as frameworks, patterns and collaborations. Some of the UML diagrams are discussed.

3.1.1 Use Case Diagram of Driver Detection System

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analysed, the functionalities are captured in use cases. So, it can be said that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system. The actors can be human user, some internal applications or may be some external applications. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements.

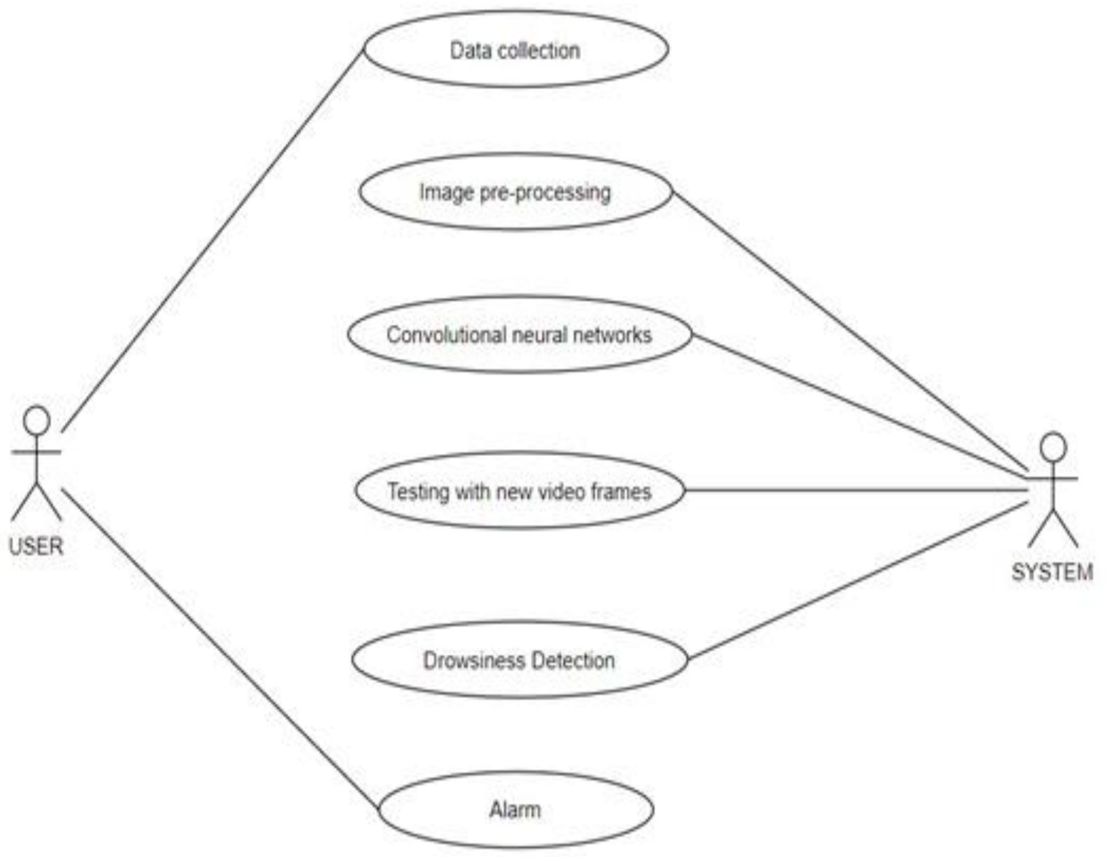


Figure 3.1 Use case diagram of DRIVER DROWSINESS DETECTION

Functionalities are to be represented as a use case in the representation. Each and every use case is a function in which the user or the server can have the access on it. The names of the use cases are given in such a way that the functionalities are preformed, because the main purpose of the functionalities is to identify the requirements .To add some extra notes that should be clarified to the user, the notes kind of structure is added to the use case diagram. Only the main relationships between the actors and the functionalities are shown because all the representation may collapse the diagram. The use case diagram as shown in Figure 3.1.

3.1.2 Class Diagram of Driver Detection System

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified. All of these specifications for the system is displayed as a class diagram in Figure 3.2

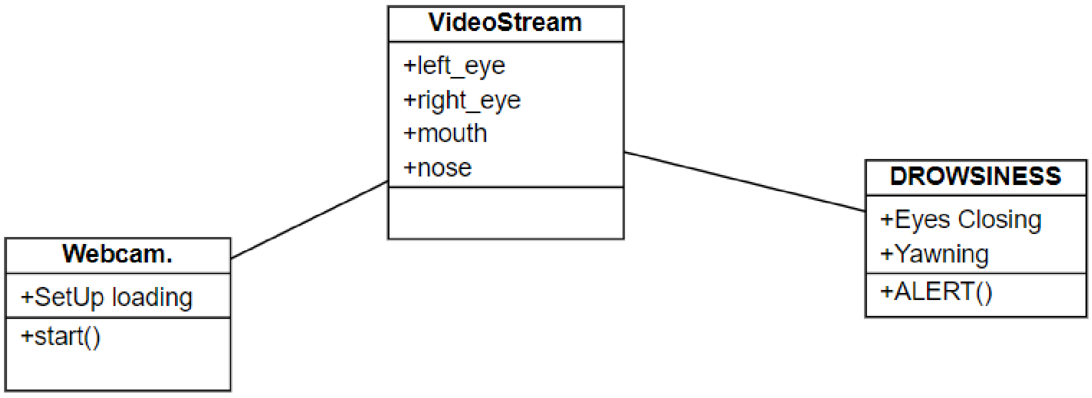


Figure 3.2 Class diagram of DRIVER DROWSINESS DETECTION

3.1.3 Sequence Diagram of Driver Detection System

UML sequence diagrams model the flow of logic within the system in a visual manner, enabling to both document and validate the logic, and are commonly used for both analysis and design purposes.

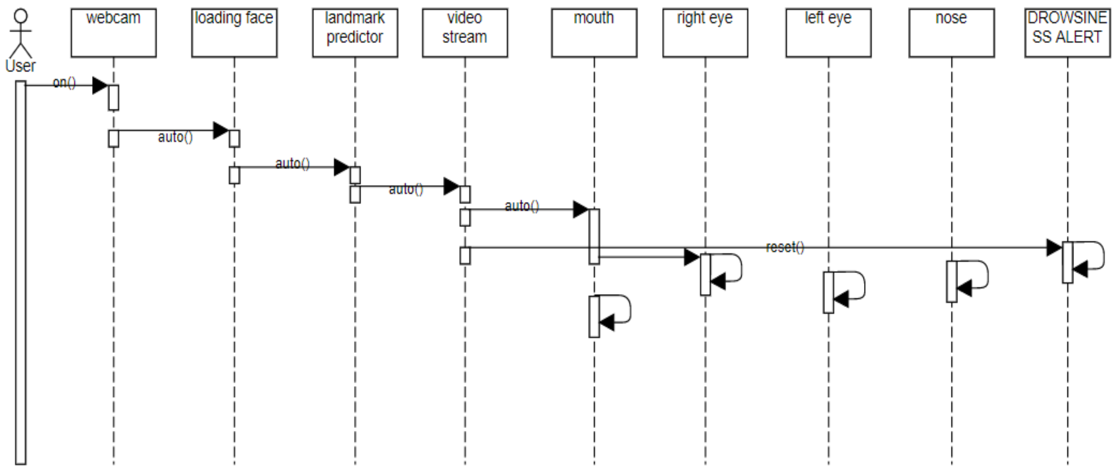


Figure 3.3 Sequence diagram of DRIVER DETECTION SYSTEM

The various actions that take place in the application in the correct sequence are shown in Figure 3.3 Sequence diagrams are the most popular UML for dynamic modelling.

3.1.4 Activity Diagram of Driver Detection System

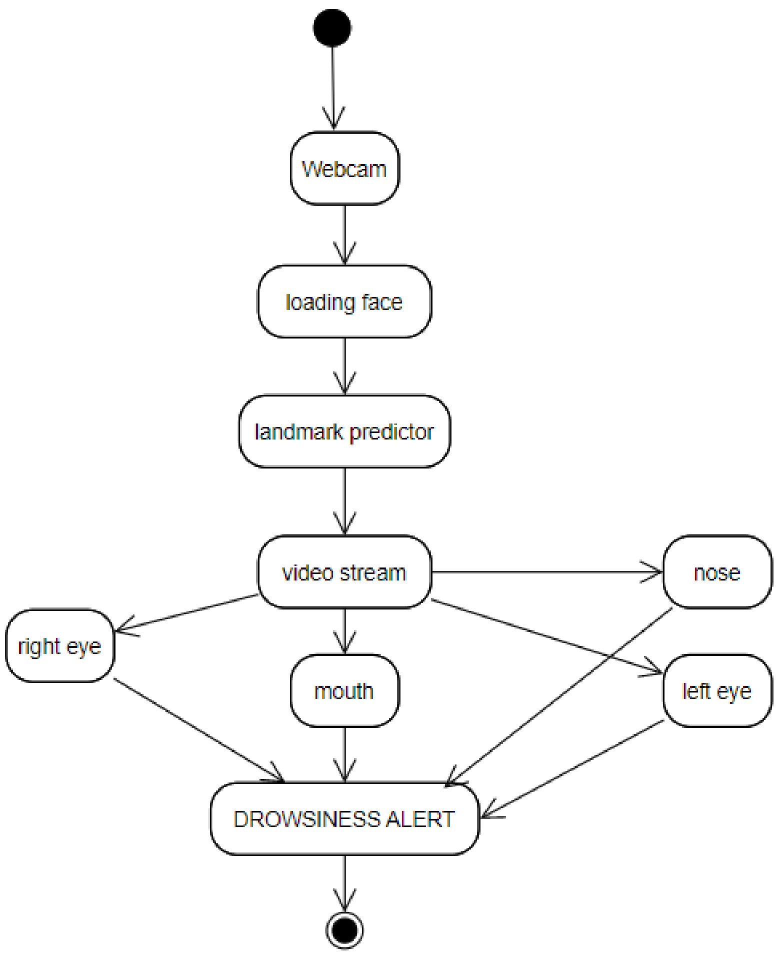


Figure 3.4 Activity diagram of DRIVER DETECTION SYSTEM

Activity is a particular operation of the system. Activity diagram is suitable for modelling the activity flow of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. An application can have multiple systems. Activity diagram also captures these systems and describes the flow from one system to another. This specific usage is not available in other diagrams. These systems can be database, external queues, or any other system. Activity diagram is suitable for modelling the activity flow of the system. It does not show any message flow from one activity to another. Activity diagram is sometime considered as the flow chart. Although the diagrams look like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single. The Figure 3.4 shows the activity diagram of the developed application.

3.1.5 State Chart Diagram of Driver Detection System

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagram is to model lifetime of an object from creation to termination. State chart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system. A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioural diagram and it represents the behaviour using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behaviour of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams. We prefer to model the states with three or more states.

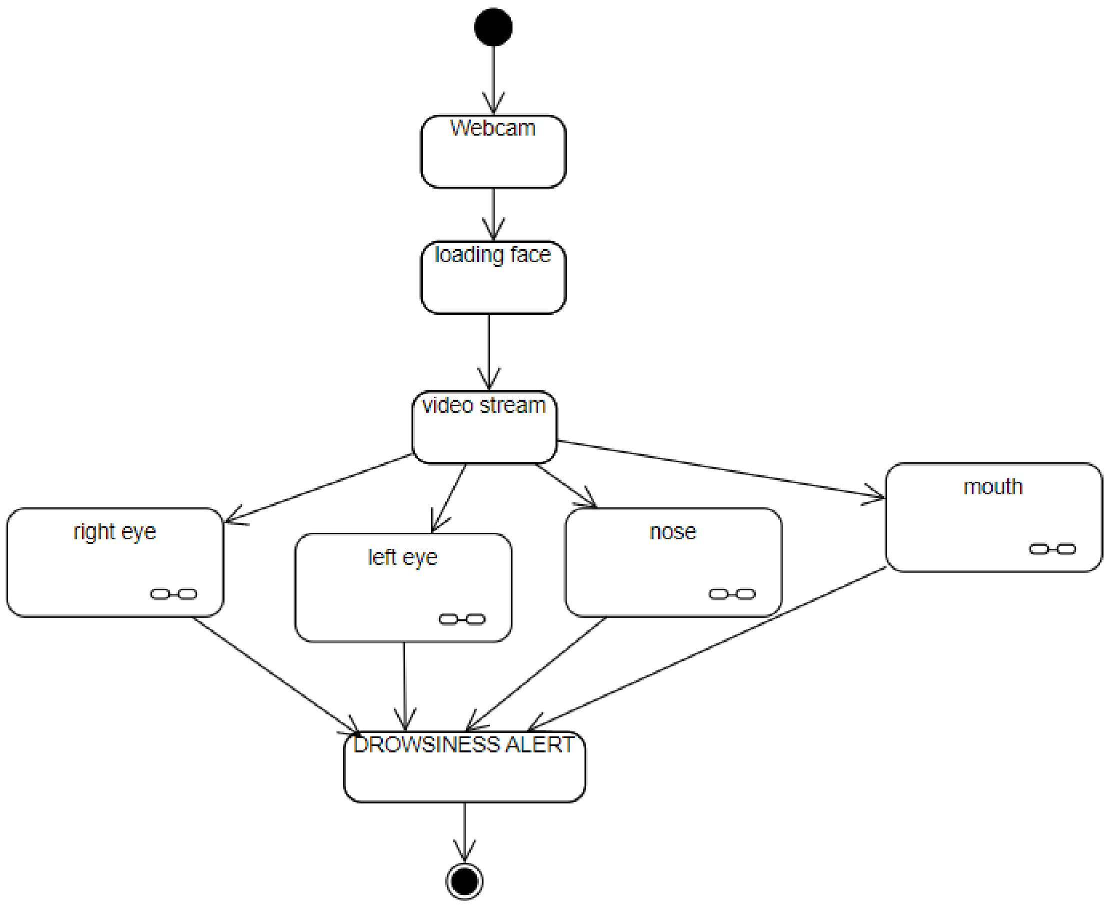


Figure 3.5 State Chart diagram of DRIVER DETECTION SYSTEM

3.1.6 Component Diagram of Driver Detection System

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Thus, from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment. A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole. The component diagram is used to explain working and behaviour of various components of a system and is static diagrams of UML.

They are also used for subsystem modelling. The main purpose of component diagram is simply to show relationship among various components of a system.

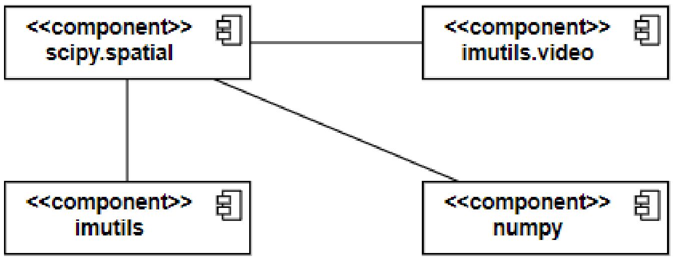


Figure 3.6 Component diagram of DRIVER DETECTION SYSTEM

3.1.7 Deployment Diagram of Driver Detection System

Deployment Diagrams are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. We illustrate system architecture as distribution of software artifacts over distributed targets. An artifact is the information that is generated by system software. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.

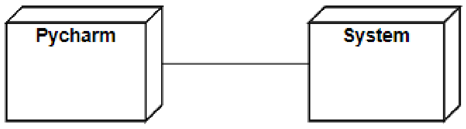


Figure 3.7 Deployment diagram of DRIVER DETECTION SYSTEM

CHAPTER 4

SYSTEM ARCHITECTURE

In this chapter, the System Architecture for the Driver Drowsiness Detection is represented and the modules are explained.

4.1 ARCHITECTURE DESIGN

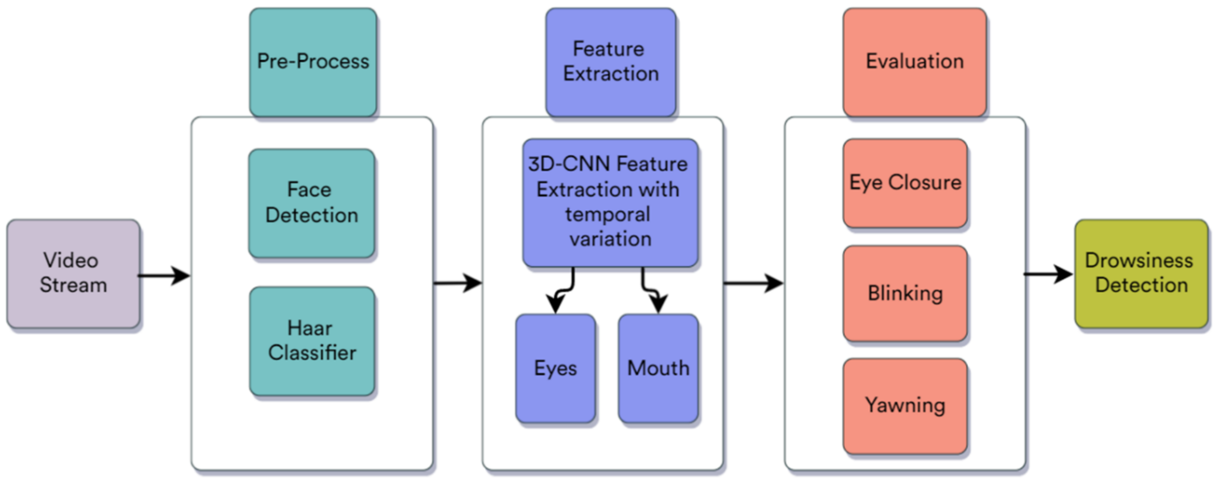


Figure 4.1 SYSTEM ARCHITECTURE DIAGRAM

4.2 ARCHITECTURE DESCRIPTION

- **Pre-Processing**: The pre-processing phase is a necessary step that is performed prior to the core process of applying the data to the convolution neural networks. The measured raw data when applied directly to the convolution neural networks creates error in the output data and hence the preprocessing phase is considered to be a more vital process for processing the raw data into the acceptable format by the convolution neural networks.

- **Feature Extraction**: The motive for employing the Convolution Neural Networks (CNN) over the Neural Networks in the driver drowsiness monitoring system is that, the Neural Networks involve complex procedures to train the datasets and it requires all the datasets must be trained which is a time consuming and complex process.

- **Evaluation**: The input image is considered as the test image and is compared with the trained image to classify the emotion of the driver under multiple levels eye closure, blinking and yawning.

4.3 MODULES

The entire architecture is divided into 6 modules.

1. Face Detection
2. Eye Detection
3. Face Tracking
4. Eye Tracking
5. Drowsiness Detection
6. Distraction Detection

4.3.1 Face Detection:

This module takes input from the camera and tries to detect a face in the video input. The detection of the face is achieved through the Haar classifiers mainly, the Frontal face cascade classifier. The face is detected in a rectangle format and converted to grayscale image and stored in the memory which can be used for training the model.

4.3.2 Eye Detection:

Since the model works on building a detection system for drowsiness, we need to focus on the eyes to detect drowsiness. The eyes are detected through the video input by implementing a haar classifier namely Haar Cascade Eye Classifier. The eyes are detected in rectangular formats.

4.3.3 Face Tracking:

Facial detection identifies and localizes human faces and ignores any background objects. OpenCV uses Harr cascade of classifiers where each frame of the video is passed through stages of classifiers and if the frame passes through all the classifiers, the face is present else the frame is discarded from the classifier i.e., the face is not detected. Due to the real-time nature of the project, we need to track the faces continuously for any form of distraction. Hence the faces are continuously detected during the entire time.

4.3.4 Eye Tracking:

The input to this module is taken from the previous module. The eyes state is determined through CNN algorithm.

4.3.5 Drowsiness detection:

In the previous module the frequency is calculated and if it remains 0 for a longer period then the driver is alerted for the drowsiness through an alert from the system.

CHAPTER 5

SYSTEM IMPLEMENTATION

In this chapter, the System Implementation for the Driver Drowsiness Detection is explained in detail.

5.1 IMPLEMENTATION OF DRIVER DROWSINESS DETECTION

The system designs are implemented in a PyCharm Text Editor. Here, the various functionalities required for the application are implemented by coding them in Python.

Driver drowsiness detection system is divided into three main blocks, i.e., camera, speaker, and applications for detecting driver drowsiness. Camera block record the driver's facial image and submit it to the drowsiness detection applications as processing input. Processing will take place at any time in real-time, when it was discovered that the driver was in a state of drowsiness then the application will send a sound through the speakers to wake the driver. Drowsiness detection system blocks are made is shown by the following TABLE I.

TABLE I

DROWSINESS DETECTION SYSTEM SPESIFICATION

Block	Interface	Block relation	Function
Camera	Web camera 1 pc, in front of driver	Connect to application through USB	Input to get images of driver
Speaker	Active speaker, 1 pc	Connect to application through audio port	Output to sound warning to the driver
Application	1 PC	Connect to camera and speaker to receive input and send from analysed image	Detect drowsiness from image from camera and send warning to driver

TABLE II
FUNCTIONAL SPECIFICATION

Function	**Interaction**	**Description**
Detect face of driver	Webcam – face of driver	Webcam records the face of driver and then searched the area of the face, if the face is found then eye detection process is done, but when the face is not found then the eye detection process will not be done again
Detect the eyes of driver	Webcam – eyes of driver	After successfully detected the driver's face, the next process is to find the eyes of driver obtained by face.
Detect the drowsiness of driver	Webcam – eyes of driver.	Once the eyes are found, later determined the driver's eyes closed for a while or not. Closed boundaries here is 3 seconds, eyes closed if for 3 seconds then the driver is considered sleepy. If only closed 1 second then considered blink.
Warn the driver	Computer - speaker	If the driver is considered sleepy then computer will play audio file through speaker.

Drowsiness detection application is made using PyCharm which is running on Microsoft Windows 10/11 operating system. Application is made using procedural programming using function which is provide by compiler. From implementation we get executable file helloEyes.exe, to run detection process.

Stages of face and eye detection is done as follows.

1. Getting Haar training values

Training value is obtained from the classifier that contains the training data has been done before. The training process obtains a sample of the images from the cameras and compared with values in OpenCV. Frame rate will be used to detect whether the object image is a face or not.

2. Determining ROI (Region of Interest) on the images

Region of Interest is a rectangular area that is used to determine the position of an object or area required for further processing in the system. By getting ROI, detection process becomes faster because the scope of searching becomes smaller so there is no need to compare all of the values to determine which area is eyes.

3. Projection Integral

After face is detected, next process is calculating integral projection to determine area of the eyes. Here, area of the eyes is assumed 1/3 from top of face. Object searching starts from sub window with 24x24 size of scale on whole face

4. Eyes detection

After area of the eyes is determined then we search the eyes. Eyes searching in sub windows based on XML classifier (haarcascade_eye.xml). To create XML classifier, method gentle ad boost is used, by looking features which have high degree of differentiation. This is done by evaluating every feature to training data using value of each feature. Feature which has the greatest range value between eyes and not eyes is considered as the best feature.

CHAPTER 6
CODING AND SCREENSHOTS

6.1 DRIVER DROWSINESS DETECTION

Main.py

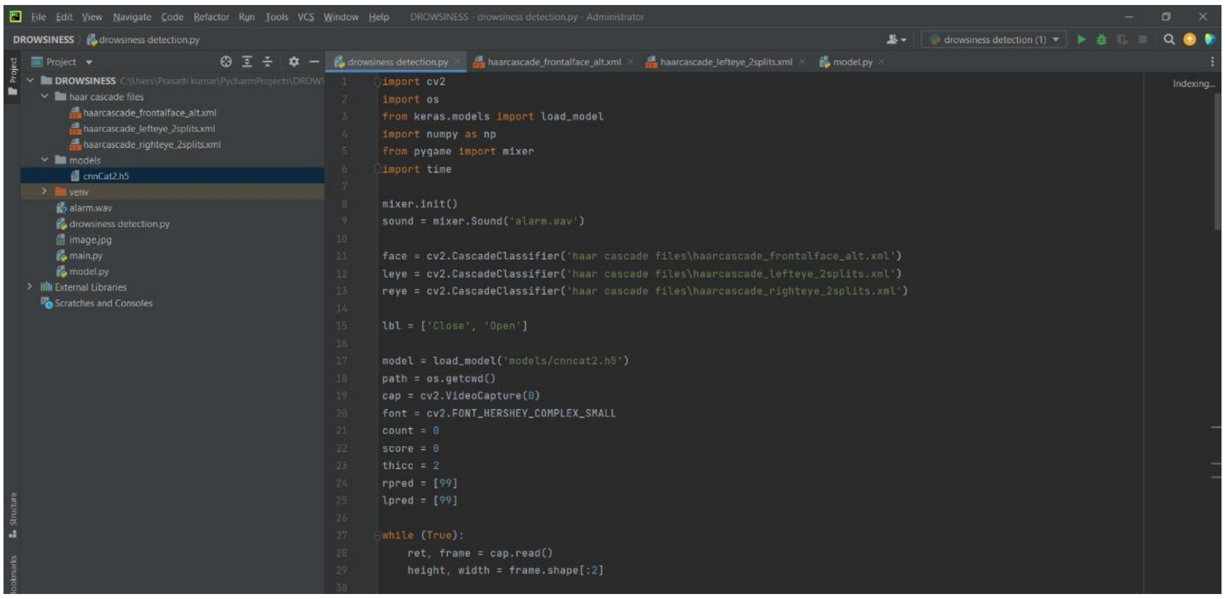
```
import cv2
import os
from keras.models import load_model
import numpy as np
from pygame import mixer
import time

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

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

lbl = ['Close', 'Open']

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

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

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face.detectMultiScale(gray, minNeighbors=5, scaleFactor=1.1,
minSize=(25, 25))
    left_eye = leye.detectMultiScale(gray)
    right_eye = reye.detectMultiScale(gray)

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

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

    for (x, y, w, h) in right_eye:
        r_eye = frame[y:y + h, x:x + w]
        count = count + 1
        r_eye = cv2.cvtColor(r_eye, cv2.COLOR_BGR2GRAY)
        r_eye = cv2.resize(r_eye, (24, 24))
        r_eye = r_eye / 255
        r_eye = r_eye.reshape(24, 24, -1)
        r_eye = np.expand_dims(r_eye, axis=0)
        # rpred = model.predict_classes(r_eye)
        rpred = np.argmax(model.predict(r_eye), axis=-1)
        if (rpred[0] == 1):
            lbl = 'Open'
        if (rpred[0] == 0):
            lbl = 'Closed'
        break

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

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

    if (score < 0):
        score = 0
    cv2.putText(frame, 'Score:' + str(score), (100, height - 20), font, 1, (255, 255, 255),
1, cv2.LINE_AA)
    if (score > 15):
        # person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path, 'image.jpg'), frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass
        if (thicc < 16):
            thicc = thicc + 2
        else:
            thicc = thicc - 2
            if (thicc < 2):
                thicc = 2
        cv2.rectangle(frame, (0, 0), (width, height), (0, 0, 255), thicc)
    cv2.imshow('Drowsiness Detector', 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, 24, 1)),
    MaxPooling2D(pool_size=(1, 1)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1, 1)),
    # 32 convolution filters used each of size 3x3
    # again
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(1, 1)),

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

    # randomly turn neurons on and off to improve convergence
    Dropout(0.25),
    # flatten since too many dimensions, we only want a classification output
    Flatten(),
    # fully connected to get all relevant data
    Dense(128, activation='relu'),
    # one more dropout for convergence' sake :)
    Dropout(0.5),
    # output a softmax to squash the matrix into output probabilities
    Dense(2, activation='softmax')
])

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

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

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

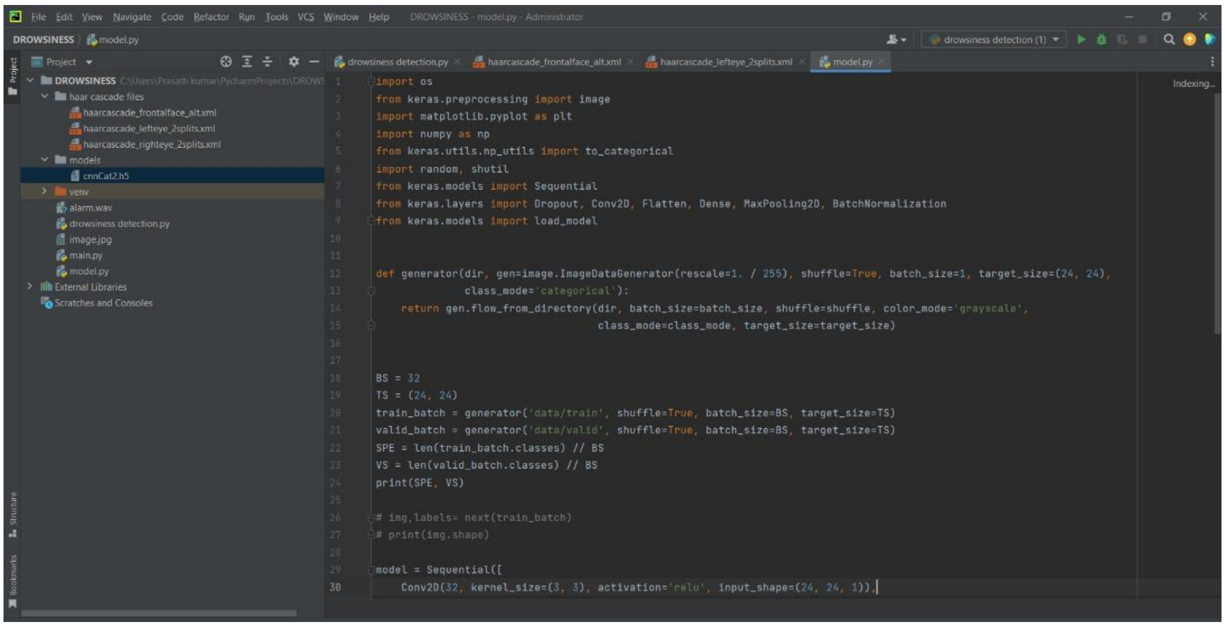
6.2 OUTPUT AND COMPARISON

SAMPLE CODE



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

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

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

lbl = ['Close', 'Open']

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

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

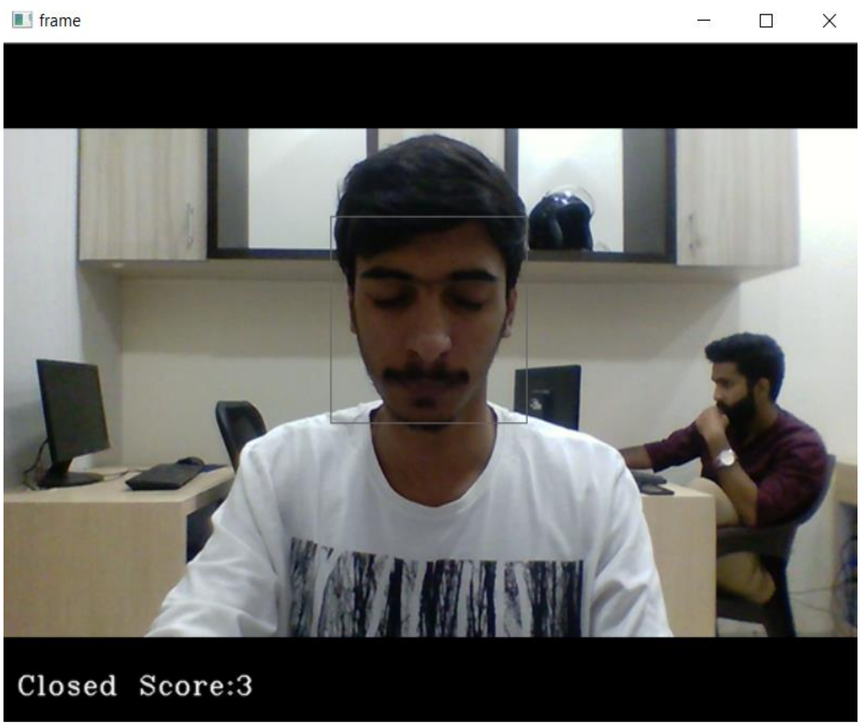
Fig 6.1 Sample Code



```python
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, 24, 1)),
```

Fig 6.2 Sample Code

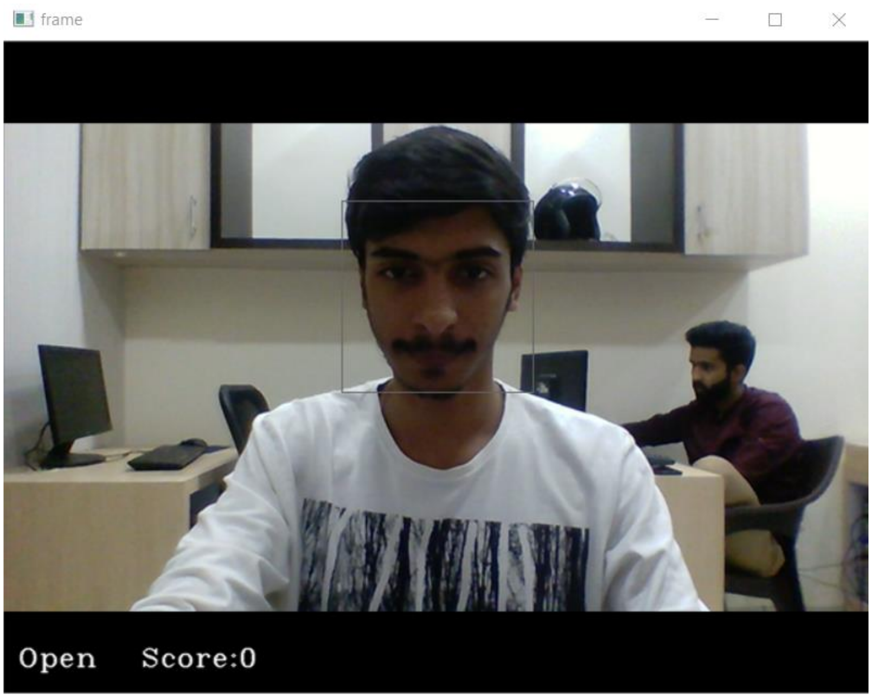
EYES CLOSED



Closed Score:3

Fig 6.3 Closed Eyes

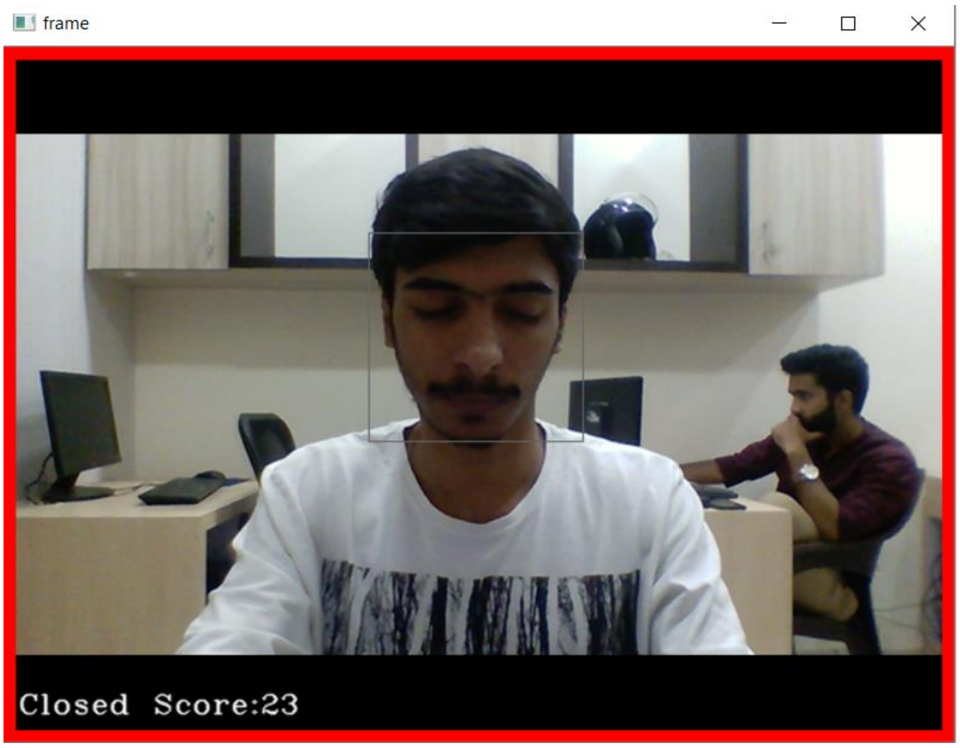
EYES OPENED



Open Score:0

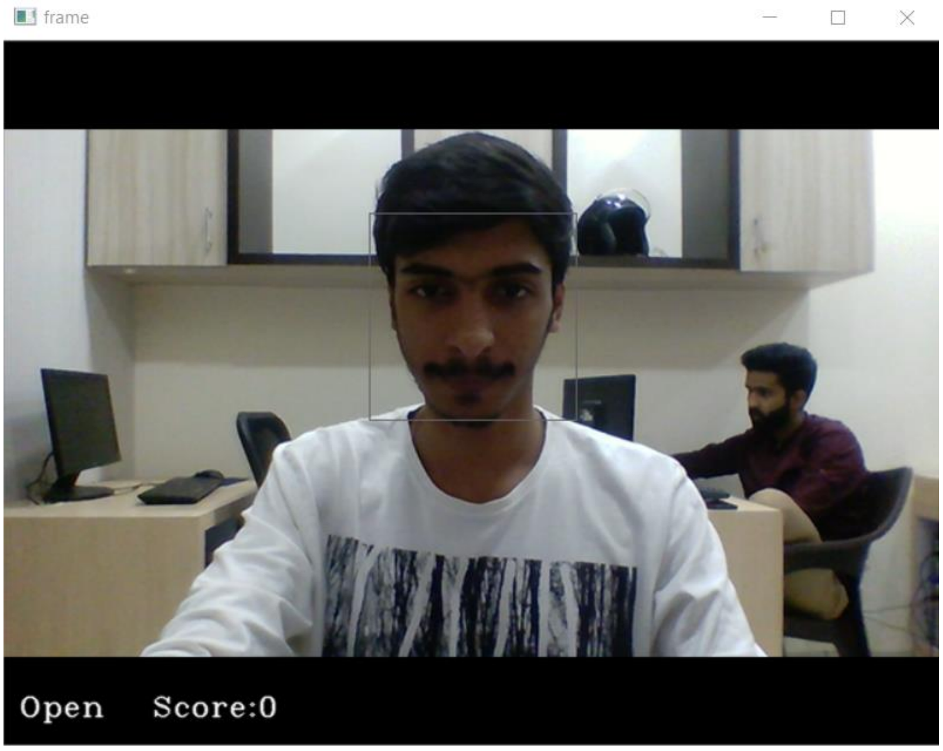
Fig 6.4 Opened Eyes

DETECTION



Closed Score:23

Fig 6.5 Detection of Drowsiness



Open Score:0

Fig 6.6 After Detection of Drowsiness

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

The current study developed an automated system for detecting drowsiness of the driver. The continuous video stream is read from the system and is used for detecting the drowsiness. It is detected by using haar cascade algorithm. The haar cascade algorithm uses haar features to detect face and eyes. Haar features are predefined are used for detecting different things. The haar features are applied on the image and blink frequency is calculated using CNN algorithm. If the value reaches 15, then it detects as sleepy and alerts driver by activating an alarm. If the value remains constant for longer periods, then the driver is said to be distracted then also an alarm is activated.

7.2 FUTURE WORK

The work can be extended by extracting the features of mouth where the driver can be detected as drowsy through yawning. If the driver yawns repeatedly for a greater number of times, then we can say that he is in sleepy mode. If the number exceeds a limit, then we can alert the driver. This work can also be extended by implementing in full night light using IR web cam. It is camera which uses infrared radiations to detect whether the person is drowsy or not. While this is a research project, there is scope when this completely turns out to be developed into an application which can be run by the end users on their own for their own purposes on their own systems.

CHAPTER 8

REFERENCES

[1] Hong Su and Gangtie Zheng, “A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness” IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 38, NO. 5, SEPTEMBER 2008.

[2] Fabian Friedrichs and Bin Yang, “Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions” 2010 IEEE Intelligent Vehicles Symposium University of California, San Diego, CA, USA June 21-24, 2010.

[3] M.J. Flores J. Ma Armingol A. de la Escalera, “Driver drowsiness detection system under infrared illumination for an intelligent vehicle” Published in IET Intelligent Transport Systems Received on 13th October 2009 Revised on 1st April 2011.

[4] Zhang, Wei; Cheng, Bo; Lin, Yingzi,” Driver drowsiness recognition based on computer vision technology.” Published in: Tsinghua Science and Technology (Volume: 17, Issue: 3) Page(s):354 - 362 Date of Publication: June 2012

[5] Ralph Oyini Mbouna, Seong G. Kong, Senior Member, IEEE, and Myung-Geun Chun,” Visual Analysis of Eye State and Head Pose for Driver Alertness Monitoring.” IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 14, NO. 3, SEPTEMBER 2013

[6] Eyosiyas Tadesse, Weihua Sheng, Meiqin Liu,” Driver Drowsiness Detection through HMM based Dynamic Modeling.” 2014 IEEE International Conference on Robotics & Automation (ICRA) Hong Kong Convention and Exhibition Center May 31 - June 7, 2014. Hong Kong, China.

[7] Gustavo A. Peláez C., Fernando García, Arturo de la Escalera, and José María Armingol,” Driver Monitoring Based on Low-Cost 3-D Sensors.” IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 15, NO. 4, Page(s): 1855 - 1860 AUGUST 2014.

[8] Knapik, M.; Cyganek, B. Driver's fatigue recognition based on yawn detection in thermal images. Neurocomputing 2019, 338, 274–292.

[9] Kiashari, S.E.H.; Nahvi, A.; Bakhoda, H.; Homayounfard, A.; Tashakori, M. Evaluation of driver drowsiness using respiration analysis by thermal imaging on a driving simulator. Multimed. Tools Appl. 2020, 79, 17793–17815.

[10] Tayab Khan, M.; Anwar, H.; Ullah, F.; Ur Rehman, A.; Ullah, R.; Iqbal, A.; Lee, B.-H.; Kwak, K.S. Smart real-time video surveillance platform for drowsiness detection based on eyelid closure. Wirel. Commun. Mob. Comput. 2019, 2019, 1–9.

[11] Ouabida, E.; Essadike, A.; Bouzid, A. Optical correlator-based algorithm for driver drowsiness detection. Optik 2020, 204, 164102.

[12] Maior, C.B.S.; das Chagas Moura, M.J.; Santana, J.M.M.; Lins, I.D. Real-time classification for autonomous drowsiness detection using eye aspect ratio. Expert Syst. Appl. 2020, 158, 113505.

[13] Bamidele, A.; Kamardin, K.; Syazarin, N.; Mohd, S.; Shafi, I.; Azizan, A.; Aini, N.; Mad, H. Non-intrusive driver drowsiness detection based on face and eye tracking. Int J. Adv. Comput. Sci. Appl. 2019, 10, 549–569.

[14] Hashemi, M.; Mirrashid, A.; Shirazi, A.B. Driver Safety Development: Real-Time Driver Drowsiness Detection System Based on Convolutional Neural Network. SN Comput. Sci. 2020, 1, 1–10.

[15] Zandi, A.S.; Quddus, A.; Prest, L.; Comeau, F.J. Non-intrusive detection of drowsy driving based on eye tracking data. Transp. Res. Rec. 2019, 2673, 247–57.

[16] Phanikrishna, S. Automatic classification methods for detecting drowsiness using wavelet packet transform extracted time-domain features from single-channel EEG signal. J. Neurosci. Methods 2021, 347, 108927.