WAKEGUARD: DROWSINESS DETECTION & ALARM SYSTEM

By

Miss Archana Chawala

Mr. Gaurav Dev

Miss Shivani Agrawal

National Institute of Technology Arunachal Pradesh

(Established by Ministry of Education, Govt. of India)

Jote, District: Papum Pare, Arunachal Pradesh – 791113

May, 2024

WAKEGUARD: DROWSINESS DETECTION & ALARM SYSTEM

(CS-499)

Thesis

Submitted in partial fulfillment of the requirements
for the award of degree of

Bachelor of Technology

By

Miss Archana Chawala (0000001258/A/2020)
Mr. Gaurav Dev (0000001273/A/2020)
Miss Shivani Agrawal (0000001335/A/2021)

Under the supervision of:

Dr. Rajat Subhra Goswami
Associate Professor
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

National Institute of Technology Arunachal Pradesh
(Established by Ministry of Education, Govt. of India)
Jote, District: Papum Pare, Arunachal Pradesh 791113

May, 2024

राष्ट्रीय प्रौद्योगिकी संस्थान अरुणाचल प्रदेश
NATIONAL INSTITUTE OF TECHNOLOGY ARUNACHAL PRADESH
(शिक्षा मंत्रालय, भारत सरकार के तहत राष्ट्रीय महत्व का संस्थान)
(Institute of National Importance under Ministry of Education, Govt. of India)
जोटे, अरुणाचल प्रदेश -791113, भारत
JOTE, ARUNACHAL PRADESH -791113, INDIA
ई-मेल E-Mail: nitarunachal@nitap.ac.in/registrarcell@nitap.ac.in
वेबसाइट Website: www.nitap.ac.in, फोन Ph: 0360-2954549

CERTIFICATE OF APPROVAL

The dissertation entitled "**WAKEGUARD**: **Drowsiness Detection & Alarm System**" submitted by **ARCHANA CHAWALA** bearing Registration No. **0000001258/A/2020**, **GAURAV DEV** bearing Registration No. **0000001273/A/2020**, **SHIVANI AGRAWAL** bearing Registration No. **0000001335/A/2021** is presented in a satisfactory manner to warrant its acceptance as a prerequisite for the degree of Bachelor of Technology in Computer Science & Engineering of the National Institute of Technology, Arunachal Pradesh. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but only for the purpose for which it has been submitted.

BOARD OF EXAMINERS:

1. **(External Examiner)**

2. **(Internal Examiner)**

राष्ट्रीय प्रौद्योगिकी संस्थान अरुणाचल प्रदेश
NATIONAL INSTITUTE OF TECHNOLOGY ARUNACHAL PRADESH
(शिक्षा मंत्रालय, भारत सरकार के तहत राष्ट्रीय महत्व का संस्थान)
(Institute of National Importance under Ministry of Education, Govt. of India)
जोटे, अरुणाचल प्रदेश -791113, भारत
JOTE, ARUNACHAL PRADESH -791113, INDIA
ई-मेल E-Mail: nitarunachal@nitap.ac.in/registrarcell@nitap.ac.in
वेबसाइट Website: www.nitap.ac.in, फोन Ph: 0360-2954549

CERTIFICATE FROM SUPERVISOR

This is to certify that the dissertation entitled "**WAKEGUARD**: **Drowsiness Detection & Alarm System**" submitted by **ARCHANA CHAWALA** bearing Registration No. **0000001258/A/2020**, **GAURAV DEV** bearing Registration No. **0000001273/A/2020**, **SHIVANI AGRAWAL** bearing Registration No. **0000001335/A/2021** to the Department of Computer Science & Engineering of the National Institute of Technology, Arunachal Pradesh, as a partial fulfillment of their B. Tech Degree in Computer Science & Engineering of the Institute is absolutely based upon his/her own work, carried out during the period from January - May, 2024 under my supervision. Neither this dissertation nor any part of it has been submitted for the award of any other degree of this Institute or any other Institute/University.

(Dr. Rajat Subhra Goswami)

Supervisor

Date: 22.05.2024

राष्ट्रीय प्रौद्योगिकी संस्थान अरुणाचल प्रदेश
NATIONAL INSTITUTE OF TECHNOLOGY ARUNACHAL PRADESH
(शिक्षा मंत्रालय, भारत सरकार के तहत राष्ट्रीय महत्व का संस्थान)
(Institute of National Importance under Ministry of Education, Govt. of India)
जोटे, अरुणाचल प्रदेश -791113, भारत
JOTE, ARUNACHAL PRADESH -791113, INDIA
ई-मेल E-Mail: nitarunachal@nitap.ac.in/registrarcell@nitap.ac.in
वेबसाइट Website: www.nitap.ac.in. फोन Ph: 0360-2954549

PLAGIARISM UNDERTAKING CERTIFICATE

This is to certify that the dissertation entitled "**WAKEGUARD**: **DROWSINESS DETECTION & ALARM SYSTEM**" submitted by **ARCHANA CHAWALA** bearing Registration No. **0000001258/A/2020**, **GAURAV DEV** bearing Registration No. **0000001273/A/2020**, **SHIVANI AGRAWAL** bearing Registration No. **0000001335/A/2021** to the Department of Computer Science & Engineering of the National Institute of Technology, Arunachal Pradesh, as a partial fulfillment of their B.Tech Degree in Computer Science & Engineering of the Institute is free from any plagiarism articles (with similarity index of 11%). Each chapter is an outcome of independent and original work; thus, duly acknowledge all sources from which the ideas and extracts have been taken in adherence to the law of anti-plagiarism.

(Archana Chawala)	(Gaurav Dev)	(Shivani Agrawal)
B. Tech	B. Tech	B. Tech
Date: 22.05.2024	Date: 22.05.2024	Date: 22.05.2024

(Dr. Rajat Subhra Goswami)
Supervisor
Date: 22.05.2024

ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and sincere thanks to my project supervisor **Dr. Rajat Subhra Goswami**, Associate Professor & HOD, Department of Computer Science & Engineering, National Institute of Technology, Arunachal Pradesh whose unwavering guidance, encouragement, and profound expertise were pivotal in the successful completion of this project.

We would also like to extend our heartfelt thanks to **Dr. Swarnendu Kumar Chakraborty**, **Dr. Koj Sambyo**, **Dr. Subhasish Banerjee**, **Dr. Achyuth Sarkar** and **Dr. Biri Arun** for their valuable advice and assistance whenever required. Additionally, we are immensely grateful to **Dr. Prases Kumar Mohanty** for his help in arranging various hardware components needed for the project, and to Rahul Prajapati, Ph.D Scholar for his assistance in setting up the environment. Their support was crucial in enabling us to overcome technical challenges and successfully complete our project.

Moreover, our appreciation goes out to all the non-teaching staff members and to our friends for their unwavering encouragement and support throughout the course of this project.

Lastly, we extend our profound gratitude to our families for their unwavering love, understanding, and motivation, enabling us to navigate through the complexities of academic pursuits and successfully complete this thesis.

Archana Chawala	Gaurav Dev	Shivani Agrawal
0000001258/A/2020	0000001273/A/2020	0000001335/A/2021
Place: Jote, Papum Pare	Place: Jote, Papum Pare	Place: Jote, Papum Pare
Dated 22.05.2024	Dated 22.05.2024	Dated 22.05.2024

ABSTRACT

In today's time, India experiences 1200 road accidents every day on average. Of these, 400 result in the instant death of a person, and the remaining instances have serious consequences. The main factor contributing to these occurrences is fatigue brought on by the consumption of alcohol and lack of sleep. Therefore, to overcome this issue a drowsiness detection system named "WakeGuard" is designed to enhance safety measures.

Using advanced facial landmark mapping and deep learning techniques, the system monitors key facial features, especially the eyes, and detects early signs of drowsiness. It classifies whether a driver is alert or drowsy by analysing eye states and aspect ratios. Extensive testing has shown that the system achieves high accuracy in detecting drowsiness, making it a reliable tool for preventing accidents.

The developed system has the potential to significantly reduce drowsy driving incidents by promptly alerting drivers when drowsiness is detected. This proactive approach not only aims to save lives and prevent injuries but also contributes to overall public safety and well-being. By reducing the number of accidents, the system can lower healthcare costs, decrease traffic congestion, and improve the quality of life for drivers, thereby creating a safer and more secure environment for everyone on the road.

Key Words: Drowsiness detection; Raspberry Pi; Dlib; EAR; Deep Learning; CNN

LIST OF CONTENTS

CHAPTER 4

SOFTWARE REQUIREMENTS SPECIFICATION

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

EEG	Electroencephalography
ECG	Electrocardiography
EMG	Electromyography
Dlib	Digital Library
Open CV	Open Source Computer Vision
fNIRS	Functional Near-Infrared Spectroscopy
CNN	Convolutional Neural Network
ROI	Region of Interest
LAN	Local Area Network
SD	Secure Digital
ResNet	Residual Network
VGG	Visual Geometry Group
USB	Universal Serial Bus
VCC	Voltage Common Collector
GND	Ground
ROC Curve	Receiver Operating Characteristic Curve
AUC	Area Under the Curve
PR Curve	Precision-Recall Curve
GUI	Graphical User Interface
ID	Identification
IDE	Integrated Development Environment
GPI	Genuine Progress Indicator
GPIO	General Purpose Input Output
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
PFCN	Product Fuzzy Convolutional Network
MTCNN	Multi-Task Cascaded Convolutional Neural Networks
SRS	Software Requirement Specification

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Drowsiness while driving causes a substantial risk to road safety, leading to several fatal accidents each year. Real-time detection of driver drowsiness has become a critical focus of research and development, with the goal of reducing the risks associated with driver fatigue.

In the past two decades, extensive research has explored various methodologies to detect and address inattentiveness and drowsiness among drivers. Traditional techniques, as detailed in prior studies, often rely on physiological measurements such as brain waves, heart rate, pulse rate, and respiration. While effective, these methods may involve complex setups and sensors.

A novel approach proposed in recent studies involves leveraging facial and eye features captured by cameras to detect signs of drowsiness. By monitoring specific facial indicators such as eye blinking patterns, researchers have developed algorithms to assess drowsiness levels in real-time. This method offers a practical and non-intrusive means of detecting driver fatigue.

As a part of this project, we aim to design and implement a drowsiness detection system that focuses specifically on analysing eye movements (including blinking frequency). By processing live video feed from an onboard camera, the system will extract relevant facial features and use algorithms to assess the driver's alertness level. This project seeks to contribute to the progress of driver safety technology by developing an effective and dependable drowsiness detection system based on facial analysis, with a particular emphasis on eye behaviour.

1.2 PROBLEM STATEMENT

Development of an effective and efficient drowsiness detection system that relies on non-intrusive image processing techniques to analyse facial features, particularly focusing on eye behaviour (blinking patterns). The goal is to precisely evaluate the driver's alertness level in real-time using onboard camera data, facilitating prompt alerts or interventions to avert accidents due to drowsy driving.

1.3 MOTIVATION

Driver drowsiness is a major contributor to road accidents, leading to considerable loss of life, injuries, and economic burdens. Conventional methods, such as public awareness campaigns and scheduled rest breaks, have not been fully effective in mitigating this risk. The emergence of advanced technologies offers the potential to develop more robust solutions capable of real-time drowsiness detection and intervention. Utilizing artificial intelligence, machine learning, and multimodal sensing technologies, we can create systems that greatly enhance road safety. This thesis is motivated by the pressing need to reduce accidents related to driver fatigue and to improve overall traffic safety through innovative and dependable detection systems.

1.4 OBJECTIVES

The proposed project possesses the following objectives:

- To develop a reliable drowsiness detection system that actively monitors driver alertness, provides timely alerts, and reduces the incidence of road accidents caused by driver fatigue, thereby improving overall road safety and promoting the well-being of drivers.
- To develop and optimize image preprocessing techniques to accurately isolate and enhance relevant facial features under varying lighting and environmental conditions, ensuring robust performance of the drowsiness detection system.
- To design a prototype for drowsiness detection using raspberry pi and dlib implementation in it.

1.5 ORGANIZATION OF THE REPORT

This thesis is structured into several chapters, each focusing on a specific aspect of the research. The following is an overview of the organization of the report:

Chapter 1 provides an overview of the research topic, detailing the background, problem statement, motivation, and objectives of the study. It sets the foundation for the thesis by explaining the significance of addressing driver drowsiness and the goals of developing an effective detection system.

Chapter 2 reviews existing literature related to drowsiness detection systems and driver fatigue. It discusses previous research, current technologies, and methodologies in the

field. The review highlights the gaps in existing knowledge and underscores how this thesis aims to contribute to the field.

Chapter 3 outlines the research design and methodologies used in the study, focusing on the two different approaches namely deep learning based and dlib based drowsiness detection.

Chapter 4 specifies the hardware and software requirements necessary for the system. It includes detailed descriptions of the models used, hardware setup, and software setup, providing a comprehensive overview of the components and configurations required to build the drowsiness detection system.

Chapter 5 presents the outcomes of the experiments and evaluations conducted on the drowsiness detection system and also discusses constraints and limitations of this project.

Chapter 6 provides a detailed performance analysis of the drowsiness detection system and evaluates the system's effectiveness, accuracy, and reliability through various tests and metrics.

The final chapter i.e. **Chapter 7** summarizes the key findings of the research, presenting the overall conclusions drawn from the study, discusses the contributions of the thesis to the field of drowsiness detection and outlines the future scope of the research, suggesting potential directions for further investigation and development.

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND

Several research studies have investigated methods for detecting driver drowsiness. These studies applied various mechanisms to understand the state of mind of the driver. These techniques are broadly categorized into two main groups: methods involving direct contact (Non-Visual Features) and methods without direct contact (Visual features) [1-5]. In contact methods, a range of non-visual indicators or physiological responses [6] are used for detection including brain activity measured by electroencephalography (EEG), heart rate monitored through electrocardiography (ECG), and muscle activity assessed using electromyography (EMG). Although these techniques produce highly accurate results, they face practical limitations that have hindered wider acceptance. On the other hand, non-contact methods are generally either vehicle-based or behavioural based. Vehicle-based approaches rely on road conditions and the driver's skill level, making them less reliable. Behaviour-based measures are minimally invasive and are considered highly feasible and reliable. They detect drowsiness by monitoring shifts in the driver's facial features, using cameras [7].

Various researches have been conducted to understand the driver's state of mind and detect drowsiness in them. Mittal et al. [8] conducted a comparative analysis of various techniques for drowsiness detection and used head movement for their system. Lienhart et al. [9] introduced a technique based on a set of rotated Haar-like features, which served as a face detector, later implementing post-optimization procedures. Vitabile et al. [10] used an infrared camera interfaced with a Celoxica RC203E FPGA-based board for real-time capturing. They used the bright pupil phenomenon to detect driver drowsiness. Bhowmick et al. [11] classified the eye states (open/closed) using Otsu threshold and used this to extract the facial region. By identifying key facial landmarks like eyebrows and potential facial center, they localized the eyes.

Danisman et al. [12] presented an automatic drowsy driver monitoring based on variations in eye blink duration. They utilized Rowley's eye detection algorithm from the STASM library. The issue they encountered was due to the presence of glasses, which significantly affected their performance. Li et al. [13] emphasized yawning as a

key indicator of driver fatigue. They detected yawning by calculating the ratio of mouth height to width using two different cameras. Deng et al. [1] proposed a model called "DriCare" which detected the driver's fatigue using video images capturing yawning, blinking, and eye closure duration, all without employing any devices on the driver's body.

Tanveer et al. [14] proposed a deep-learning-based system that used functional near-infrared spectroscopy (fNIRS) to detect drowsiness. Chaudhari et al. [16] have used the Viola-Jones detection technique for calculating the facial and ocular regions. They subsequently implemented a deep convolutional neural network to classify the sleepy state of the driver. Yogarajan et al. [17] have proposed a system using EEG and ECG signals and used it as an input to 2D CNN and extracted features for drowsiness detection. Guanglong et al. [18] also implemented the system using EEG and ECG signals. They proposed a model named product fuzzy convolutional network (PFCN) and captured the temporal variation of high-dimensional EEG signals and hence, reduced time-space complexity.

Bajaj et al. [19] developed a model that combines the two different types of methods for driver drowsiness. They developed a hybrid model based upon an AI-based Multi-Task Cascaded Convolutional Neural Network (MTCNN) for non-intrusive features and the Galvanic Skin Response (GSR) sensor to collect the physiological features. They obtained an overall accuracy of 91% in all circumstances. Siddiqui [20] classified the states of drowsiness using respiration rate. They implemented a non-invasive, non-touch and impulsive-radio ultra-wideband radar. They obtained an accuracy of 87%.

Suhaiman et al. [21] have developed a real-time based system for drowsiness detection in which they captured the driver's eye state and analysed it using Python, dlib, and OpenCV. They measured the eye region of the driver and used it for detection. Jahan et al. [22] proposed a CNN based 4D model for drowsiness detection. They trained the model using MRL Dataset and found that the 4D model performed better than the existing pre-trained models such as VGG16, and VGG19 with an accuracy of 97.53%.

CHAPTER 3

METHODOLOGY

3.1 METHODOLOGY

The methods used to implement the proposed model are discussed below:

3.1.1 Foreground

The proposed “Wake Guard” model performs the following steps upon activation:

Step 1 - Upon initializing the Raspberry Pi, the model activates and initiates the camera.

Step 2 - The camera activates and starts capturing images continuously.

Step 3 - The system then processes the recorded photos.

Step 4 - If the system detects signs of drowsiness persistently across 20 consecutive frames, it triggers a buzzer alarm and displays an alert message on the screen. This will wake up the driver.

Step 5 - If not, the device keeps taking pictures without sounding an alarm until it senses drowsiness.

3.1.2 Background

The following two methodologies are implemented in the project:

3.1.2.1 Deep Learning based Implementation

The following strategy is used to implement the different deep learning models which is also shown in fig. 3.1:

1. **Dataset Preparation:** The suggested model is trained using the MRL Dataset, which consists of human eye pictures in open and closed states. The dataset is pre-processed using a variety of methods before training, including scaling, normalization, and grayscale conversion (BGR2GRAY). This preprocessing improves the dataset's quality and guarantees homogeneity.
2. **Model Training:** The model is then trained using the prepared dataset. For training, many convolutional neural networks (CNN) architectures are used,

including InceptionV3, MobileNet, ResNet, and VGG19 as shown in fig.5.3 and fig. 5.4.

3. **Testing:** The model is then tested using a separate test dataset to evaluate its generalizability and performance after training. Unused samples from the training phase are included in the test dataset. The outcome of this stage makes the model capable of accurately categorizing eye states as shown in fig. 5.5 and fig. 5.6.
4. **Detection of Unknown Images:** Once the model training is complete and the model is deemed sufficiently accurate, it is utilized for detecting eye states in unknown images. The model uses the Haar cascade frontal detector to identify the facial region in an unknown image. The identified facial region is then cropped in order to extract images of the eyes, which are the region of interest (ROI) as displayed in fig. 5.7.
5. **Eye State Classification:** The system uses the trained model to identify whether the eyes are awake or sleepy based on patterns and attributes that have been learned. If the drowsy state continues for particular time frames, then the alarm is activated.

This methodology is used to train, test, and implement the suggested model for eye state recognition, facilitating the identification of drowsiness in real-world scenarios as shown in fig. 5.8.

3.1.2.2 Dlib based Implementation

For dlib-based implementation, the following approach is used as shown in fig. 3.2:

1. **Camera Interface Setup:** The proposed model initiates with a camera interface generated by OpenCV, activating the camera and capturing images.
2. **Image processing:** From the collected photos, 68 landmark points on the face are identified using the Dlib shape predictor. The Eye Aspect Ratio (EAR) value can be determined and eye regions can be identified with the help of landmark points.
3. **Detection of Drowsiness:** A predetermined threshold is compared to the EAR value. The system initiates an alarm mechanism if the EAR value remains below the threshold for a certain number of consecutive frames (e.g., 20 frames). To

notify the user of detected drowsiness, the alert mechanism starts playing a buzzer and displaying an alert message on the screen as shown in fig. 5.9.

4. **System Reaction:** The system will continue taking pictures discreetly while keeping an eye out for any indications of sleepiness if it does not discover any.

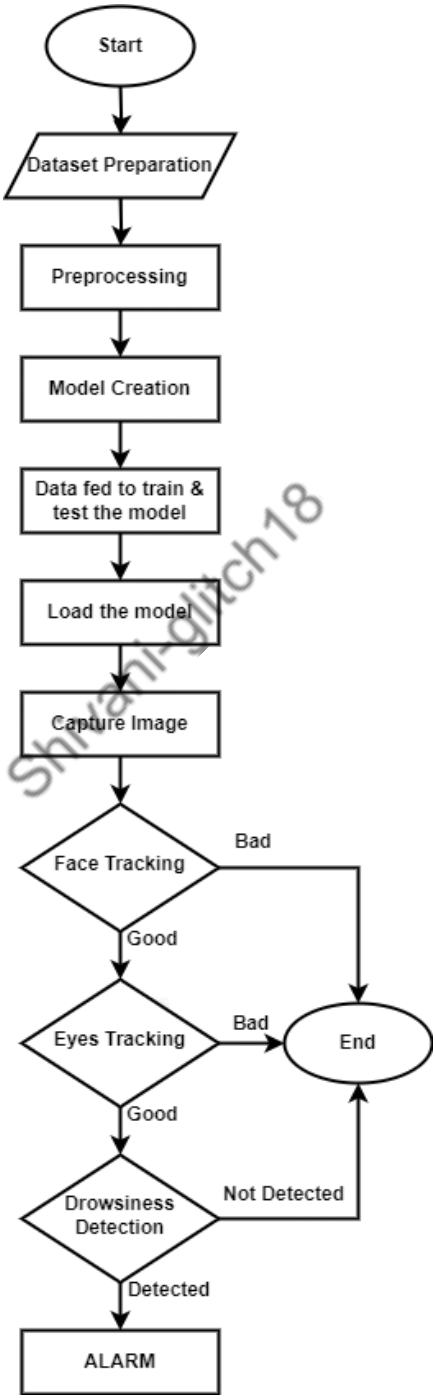


Fig. 3.1: Flowchart illustrating deep learning based implementation in the project

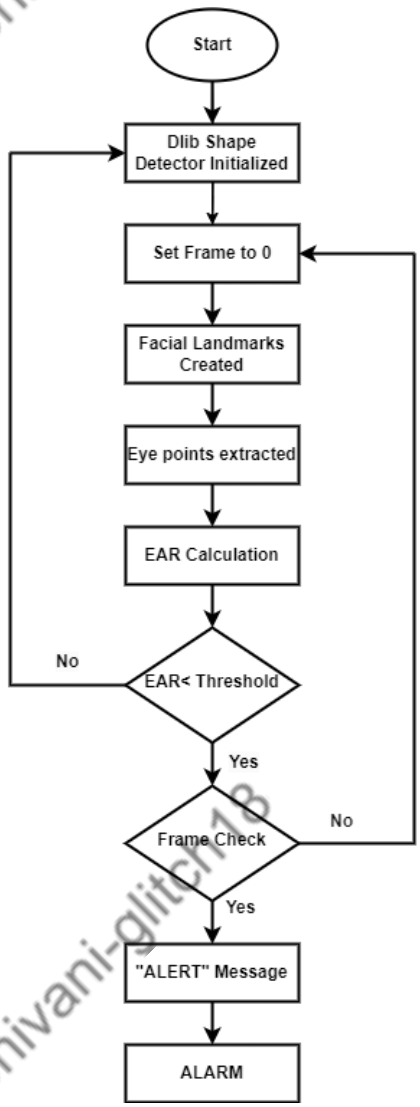


Fig. 3.2: Flowchart illustrating dlib based implementation in the project

3.2 REGION OF INTEREST

The Region of Interest (ROI) is a foundational concept in image processing and computer vision, representing a specific area within an image or video frame that is designated for focused analysis, processing, or feature extraction. The selection of an ROI is driven by the objective of a particular task or application, aiming to isolate and prioritize relevant information while minimizing computational overhead on less critical regions. By strategically defining ROIs, we can optimize the efficiency and effectiveness of image processing and computer vision algorithms across a wide range of applications, ultimately contributing to advancements in visual data analysis and interpretation.

3.3 MATHEMATICAL COMPUTATION

The mathematical calculations required for the system are given below:

3.3.1 Face Detection

Facial detection is a technology used to identify or authenticate individuals based on facial characteristics. It plays a crucial role in applications like security systems, photo organization, and video analysis. Facial detection systems employ computer vision algorithms to locate faces within images or video frames. Facial detection is implemented using OpenCV and Python, widely used for processing images and computer vision applications. The Dlib library enhances this process by precisely detecting and localizing facial landmarks. Dlib's facial landmark detector which are pre-trained identifies key features on the face, including the eyes, nose, and mouth, which are critical for subsequent analysis. The use of Dlib's models, with 68 and 5 landmark points respectively, allows for detailed facial feature extraction as shown in fig.3.3. This level of precision is essential for tasks ranging from emotion recognition to facial expression analysis. The integration of OpenCV, Python, and Dlib showcases the intersection of software development and computer vision, facilitating innovative solutions for facial analysis and recognition.

3.3.2 EAR

The Eye Aspect Ratio (EAR) is a fundamental metric employed in computer vision and facial recognition systems to monitor and track facial features, particularly focusing on eye movements and the state of eyelid closure. The EAR computation involves measuring specific distances between key facial landmarks around the eyes, such as the distance between the upper and lower eyelids relative to the distance between the inner and outer corners of the eye. The EAR serves as a valuable indicator in various applications like detecting drowsiness in drivers or monitoring attentiveness. By continuously calculating and analysing the EAR in real-time, these systems can issue alerts or take preventive measures when signs of fatigue or distraction are detected. In addition to its role in fatigue detection and attention monitoring, the EAR can also be instrumental in assessing facial expressions and emotions. Changes in the EAR can reveal subtle cues about a person's emotional state, enabling more subtle interaction in human-computer interfaces and virtual environments. The versatility of the EAR

extends beyond specific applications, providing a holistic approach to facial analysis. Its integration with computer vision and machine learning enables innovative solutions in healthcare, automotive safety, education, and beyond, highlighting the profound impact of facial recognition technologies on modern life.

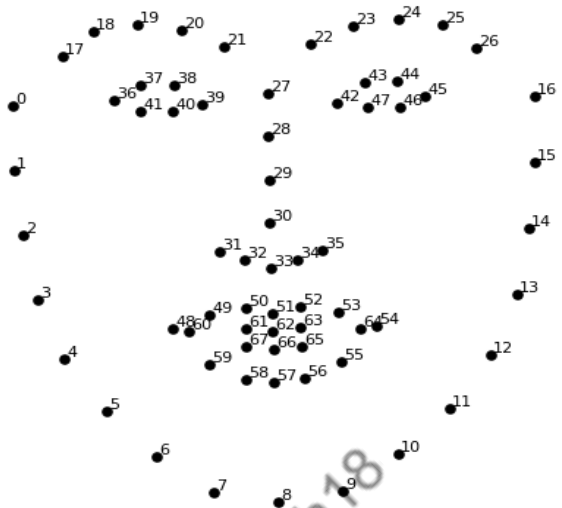


Fig. 3.3: Sequence in which 68 facial landmarks are mapped on a detected face

3.3.3 EAR Calculation

For calculating the EAR, the process begins with identifying specific eye landmarks on the face. These landmarks include the corners of the eyes and key points along the eyebrows and eyelids. The interaction starts by recognizing explicit eye landmarks, such as the corners of the eyes and the midpoint of the upper and lower eyelids.

To calculate the EAR, the vertical and horizontal distances between key facial landmarks are utilized:

$$EAR = \left(|P2 - P6| + |P3 - P5| \right) / \left(2 * |P1 - P4| \right)$$

Here's a breakdown of the landmarks involved which is also shown in fig. 3.4:

- $P1$: The leftmost point of the eye (typically the left corner).
- $P2$ & $P3$: The topmost point of the eye.
- $P4$: The rightmost point of the eye (typically the right corner).

- $P5$ & $P6$: The bottom point of the eye.

The EAR serves as a crucial metric in applications such as driver drowsiness detection and assessing attentiveness. Its calculation leverages the relationship between specific facial landmarks, enabling real-time analysis of eye behavior and aiding in various computer vision tasks.

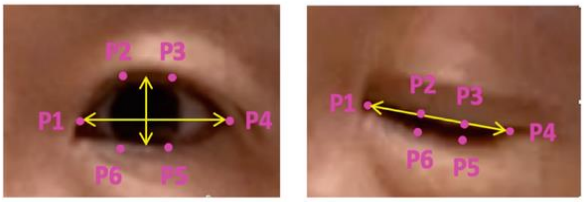


Fig. 3.4: Eye landmarks for the EAR Calculation

3.4 DATA FLOW DIAGRAM

Data Flow Diagram of the proposed project is shown in fig. 3.5 and fig. 3.6:

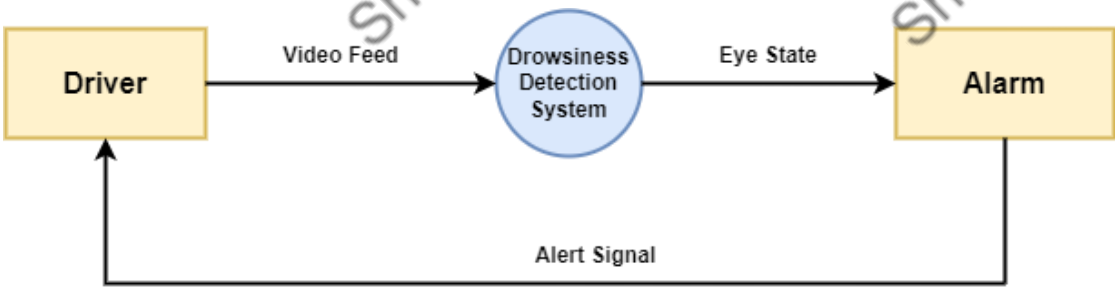


Fig. 3.5: Data Flow Diagram: Level 0

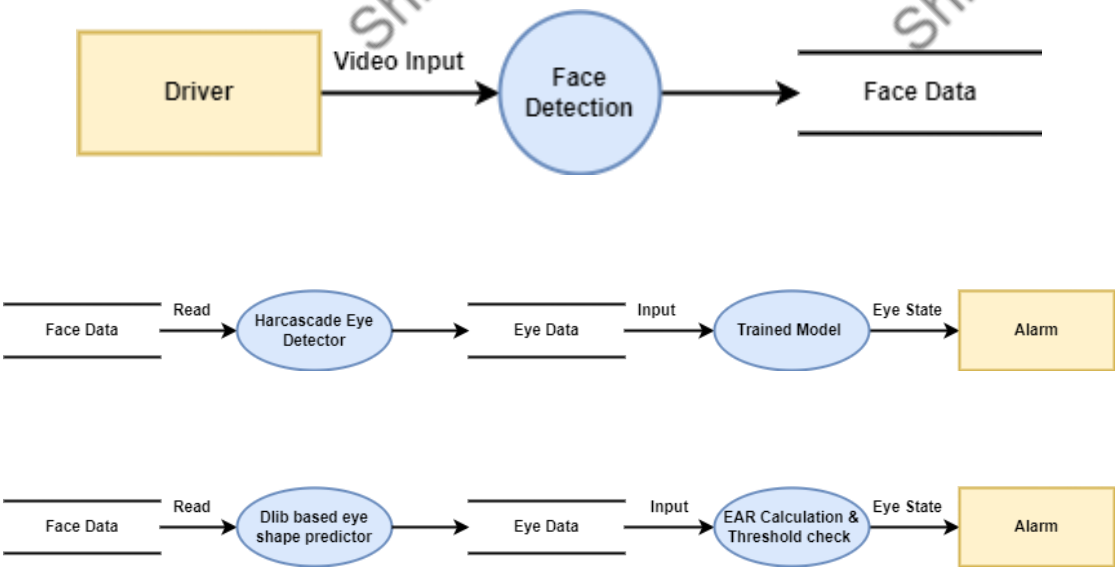


Fig. 3.6: Data Flow Diagram: Level1

CHAPTER 4

SOFTWARE REQUIREMENTS SPECIFICATION

4.1 SOFTWARE REQUIREMENTS SPECIFICATION

This section contains a list of software, hardware and models required for the project development.

4.1.1 Hardware Requirements

The following hardware equipments were used for implementing the proposed project:

4.1.1.1RaspberryPi

The Raspberry Pi 3 is a widely recognized single-board computer renowned for its versatility and affordability. This compact device is capable of interfacing with various sensors and external devices, enabling the collection of data for analysis and identification of patterns or anomalies. This capability proves invaluable for applications such as environmental monitoring or automation in industrial settings. Moreover, the Raspberry Pi 3 features sufficient processing power to execute machine learning algorithms and AI models, making it well-suited for projects involving image recognition, natural language processing, or predictive analytics. This empowers users to explore advanced technologies on a budget-friendly platform. In addition, the Raspberry Pi 3 can be used as a low-cost security testing tool, ideal for analysing network vulnerabilities, assessing security risks, or monitoring network traffic. Its versatility extends further into robotics applications, where it can serve as the brain of a robot. By controlling movements, sensors, and actuators, users can delve into topics such as autonomous navigation, object detection, or human-robot interaction.

4.1.1.2 Camera

A camera is an essential device used to capture detailed facial images for various applications, particularly in facial recognition systems where it serves as the primary input device for acquiring visual data necessary for analysis and comparison. Furthermore, the camera's role extends beyond simply capturing images, involving the capture of detailed facial features such as the structure of the face, eyes, nose, mouth, and distinct identifying characteristics. These captured facial images form the basis for

the operations of facial recognition systems. In practical terms, the camera's integration into facial recognition systems enables the extraction of unique facial patterns and features through complex algorithms. This process generates a digital representation of each individual's face, known as a face template, which serves as the basis for subsequent identification and verification tasks within the system. Its ability to capture and convey detailed facial information enables sophisticated analysis and interpretation, contributing to advancements in identification technology and digital security measures.

4.1.1.3 Buzzer

A buzzer is a type of electronic device that emits a continuous or intermittent buzzing or beeping sound. It includes a small electromechanical component that vibrates swiftly in response to an electric current, generating sound waves. Buzzer devices are commonly used in various applications for alerting, signalling, or indicating events. They can be found in alarm systems, electronic gadgets, appliances, and industrial equipment. Buzzer sounds can vary in tone and intensity depending on the design and purpose, ranging from gentle beeps for notifications to loud, attention-grabbing alarms for critical alerts. Buzzer components are compact, cost-effective, and easy to integrate into electronic circuits, making them versatile and widely used across different industries and consumer electronics.

4.1.1.4 Cooling-Fan

Cooling fans are essential components used to manage the temperature of electronic devices, including single-board computers like the Raspberry Pi. These fans are designed to dissipate heat generated by the device's components, such as the processor and other integrated circuits, to prevent overheating and maintain optimal performance and reliability. In Raspberry Pi, cooling fans are commonly integrated into custom enclosures or attached directly to the board to improve airflow and heat dissipation. By efficiently circulating air and removing excess heat, cooling fans help prevent thermal issues that can affect the Raspberry Pi's stability and longevity.

4.1.1.5 Mouse

A mouse is a common input device used with computers and other digital devices. It enables users to control the cursor on the screen by moving the mouse across a flat

surface. Typically, a mouse includes buttons that can be clicked to perform actions such as selecting, dragging, and interacting with graphical user interfaces.

4.1.1.6 Keyboard

A keyboard serves as a fundamental input device employed for typing text and inputting commands into computers and comparable devices. It consists of a set of keys laid out in a specific arrangement, including letters, numbers, punctuation marks, and special function keys. Keyboards may be connected to devices via USB or wireless technology. They are essential for text input, shortcut commands, and navigation within software applications.

4.1.1.7 Ethernet LAN

An Ethernet Local Area Network (LAN) is a network that connects computers and devices within a confined geographical area, such as a residence, workplace, or educational campus, using Ethernet cables. Ethernet LANs provide high-speed data transmission and reliable connectivity, enabling devices to communicate and share resources like files and internet access.

4.1.1.8 Jumper Wire

A jumper wire is a brief insulated wire featuring connectors at both ends, designed for establishing electrical connections between components on a breadboard or circuit board. These wires are frequently employed in electronics prototyping and experimentation to create temporary or semi-permanent connections between different points within a circuit. Jumper wires are available in assorted lengths and colors, facilitating straightforward organization and identification of connections in complex circuits.

4.1.1.9 SD Card

A Secure Digital (SD) card is a removable flash memory card utilized for storing and transferring digital data. These cards find extensive application in devices such as digital cameras, smartphones, tablets, and single-board computers like the Raspberry Pi. They offer portable and expandable storage capacity, allowing users to store photos, videos, music, documents, and software applications.

4.1.2 Software Requirements

The following software are used for implementing the proposed project:

4.1.2.1. Operating System: Raspbian

Raspbian is the official operating system derived from Debian and designed specifically for the Raspberry Pi single-board computer. It features a user-friendly interface and comes with a selection of pre-installed software optimized for Raspberry Pi hardware, enabling users to easily begin developing and running projects on the Raspberry Pi platform.

4.1.2.2 Programming Language: Python

Python is a flexible programming language recognized for its clear syntax and ease of use. It is extensively applied across diverse domains such as software development, machine learning, web development, data analysis, and scientific computing. Python's comprehensive standard library and extensive community-driven collection of third-party packages make it a preferred option for rapidly prototyping and efficiently building applications.

4.1.2.3 Open CV

OpenCV is a freely available library of programming functions primarily designed for real-time computer vision applications. It offers a range of tools and algorithms for processing images and videos, such as feature detection, object recognition, and vision applications based on machine learning.

4.1.2.4 Dlib

Dlib is a contemporary C++ toolkit that incorporates machine learning algorithms and tools aimed at developing sophisticated software solutions in C++ to address practical challenges. It is particularly known for its effectiveness in facial recognition and facial landmark detection tasks, making it valuable for applications in biometrics, image analysis, and object detection.

4.1.2.5 TensorFlow

TensorFlow is a machine learning framework that has been open-sourced by Google. This framework empowers developers to construct and train machine learning models, particularly neural networks, for diverse applications such as natural language processing, image recognition, and predictive analytics. TensorFlow is valued for its flexibility, scalability, and user-friendly design, making it widely adopted by researchers and practitioners in the artificial intelligence field.

4.1.2.6 Imutils

Imutils is a library of convenience functions for basic image processing tasks in Python. It simplifies common operations such as resizing, rotating, and displaying images, making it useful for rapid prototyping and development of computer vision applications using Python and OpenCV.

4.1.2.7 IDE Anaconda

Anaconda is a distribution that integrates Python and R programming languages for scientific computing, offering a comprehensive platform for data science and machine learning tasks. It includes conda, a package and environment manager that simplifies the installation and management of packages. Anaconda provides the Anaconda Navigator which is a graphical user interface (GUI) designed to facilitate the management of packages, environments, and IDEs like Jupyter Notebook and Spyder. These features have made Anaconda a favoured option among data scientists and researchers who work extensively with Python-based tools and libraries.

4.1.3 Models

This section contains various types of models used in the project:

4.1.3.1 CNN

A Convolutional Neural Network (CNN) is a specialized architecture within deep neural networks tailored for handling structured grid-like data, such as images or sequential data like time-series signals. CNNs excel in tasks related to visual recognition and classification by autonomously learning hierarchical feature representations from raw

data. This network model consists of different types of layers, including convolutional layers, pooling layers, and fully connected layers as displayed in fig. 4.1.

- **Convolutional Layers:** Convolutional layers are essential components of CNNs. They perform convolution operations on input data using trainable filters or kernels. As these filters slide across the input data, they extract local features, generating feature maps that encode spatial hierarchies of patterns. Convolutional layers are adept at learning translation-invariant features and are frequently employed in tasks such as image recognition and object detection.
- **Pooling Layers:** Pooling layers are utilized to decrease the spatial dimensions (width and height) of feature maps produced by convolutional layers while retaining significant information. Typical types of pooling include max pooling and average pooling, where a window moves across the feature map and combines values (e.g., selecting the maximum value) to generate a downsampled output. Pooling layers assist in managing overfitting, lowering computational complexity, and enhancing translation invariance within CNNs.
- **Fully Connected Layers (Dense Layers):** Fully connected layers, often referred to as dense layers, are conventional neural network layers where each neuron is linked to every neuron in the preceding layer. These layers are commonly positioned at the conclusion of a CNN architecture and function as classifiers or regressors. Fully connected layers aggregate high-level features learned from previous layers to make predictions based on the extracted features. They are crucial for mapping extracted features to specific output categories in tasks like image classification and object recognition.

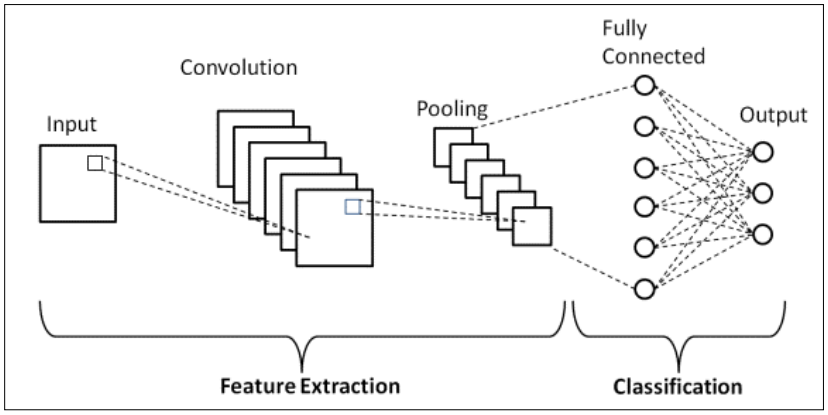


Fig. 4.1 CNN Architecture

The following types of CNN models were trained and tested for the project:

4.1.3.1.1 Inception V3

Inception V3 is a type deep convolutional neural network architecture which is built by Google. It is renowned for being effective and having a high degree of accuracy in image classification tasks. It possesses a special module called the "Inception module," which is made up of several parallel convolutional operations with varying kernel sizes. This enables the model to capture features at various scales and the level of complexities efficiently. Inception V3 is commonly used in applications where both accuracy and computational efficiency are important, such as image recognition and transfer learning.

4.1.3.1.2 ResNet

A ground-breaking deep neural network architecture called Residual Network (ResNet) is renowned for introducing residual connections, which helps to solve the problem of vanishing gradients while training very deep networks. The main novelty of ResNet is the use of skip connections, which omit one or more layers, and provide easier gradient flow during backpropagation. This makes it possible to train incredibly deep networks (e.g., ResNet-50, ResNet-101) with enhanced performance and convergence. It has achieved state-of-the-art results in multiple computer vision tasks, such as object detection, image segmentation, and image classification.

4.1.3.1.3 VGG19

VGG19 is a type of deep convolutional neural network architecture which was created by the Visual Geometry Group at the University of Oxford. It possesses several features such as simplicity in nature and uniform architecture, comprising of 19 layers (16 convolutional and 3 fully connected layers) along with max pooling layers and small receptive fields of 3x3 filters. Because of its simple design and good performance in image classification tasks, VGG19 has been set as a benchmark model in computer vision research. However, VGG19 is computationally expensive compared to more modern architectures like ResNet and Inception.

4.1.3.1.4 MobileNet

A type of convolutional neural network architecture which is lightweight and is created for mobile and embedded devices with constrained computational power is called as

MobileNet. It is based upon depthwise separable convolutions in order to reduce the number of parameters and to cut down the computational cost while maintaining its efficiency. They perform well in reference to memory usage and inference speed, enabling them to be suitable for on-device applications such as object detection, image classification, and semantic segmentation on smartphones, drones, and other edge devices. MobileNet has various variants (e.g., MobileNetV1, MobileNetV2, MobileNetV3) optimized for different trade-offs between speed, size, and accuracy, catering to diverse deployment scenarios.

4.1.4 Hardware Set-Up

Interface the Raspberry Pi to a power source utilizing a miniature Universal Serial Bus (USB) power connector. Plug in a USB console into one of the USB ports on the Raspberry Pi. Associate an Ethernet LAN link to the Ethernet port on the Raspberry Pi and interface the opposite finish to our switch. Buzzer requires power, so associate its Voltage Common Collector (VCC) (+) pin to the +5V pin to the Raspberry Pi (or the +3.3V pin), and interface its Ground (GND) (-) pin to GND pin to the Raspberry Pi. Guarantee all associations are secure and accurately made. We connected VCC pin to 36th pin and GND pin to 9th pin. The hardware setup done for the project is shown in fig. 4.2.

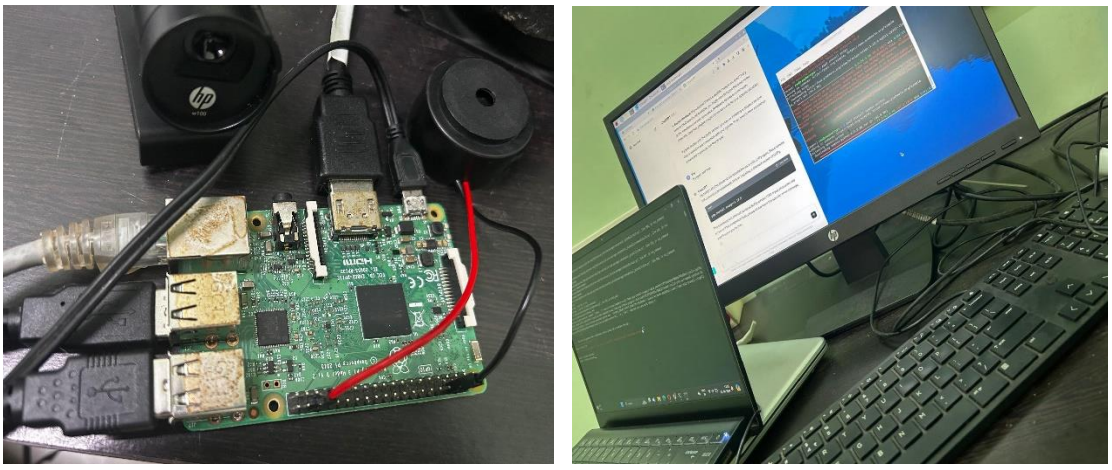


Fig. 4.2 Hardware Setup

4.1.5 Software Set-Up

First, download the latest version of the Raspberry Pi operating system (previously known as Raspbian) from the official Raspberry Pi website. Next, use a tool like Etcher, available for Windows, macOS, and Linux, to flash the Raspberry Pi operating system image onto a microSD card. Insert the microSD card (64 GB) into the Raspberry Pi's microSD card slot, and then use a HDMI cable to connect the Raspberry Pi to a display or monitor. Connect the miniature USB power connector to power on the Raspberry Pi. The Raspberry Pi will boot up and the operating system will be launched. Open a terminal window to enable access to the GPIO (General Purpose Input/Output) pins that will be utilized to operate the buzzer. Test the GPIO pins by writing a simple Python script to toggle the GPIO pin connected to the buzzer on and off. Once everything is set up and tested successfully, we can begin using the Raspberry Pi with the connected peripherals for our project.

CHAPTER 5

RESULTS & LIMITATIONS

5.1 RESULTS & DISCUSSION

Based on the provided methodology, the model is trained and tested and the model is evaluated with the accuracy obtained, confusion matrix, and classification report.

5.1.1 Dataset Description

Fundamentally, the cornerstone of training any machine learning or deep learning model lies in the availability of suitable data. To address this requirement, we have assembled a dataset sourced from the MRL Eye Dataset [15], comprising a diverse collection of images depicting both open and closed eyes as shown in fig. 5.2. This dataset was meticulously curated by compiling eye images from a cohort of 37 individuals, consisting of 33 men and 4 women.

This dataset encompasses a comprehensive collection of infrared images, encompassing both low and high resolutions. These images were captured under a diverse range of lighting conditions using various infrared imaging devices. The inclusion of images captured under different lighting conditions and with different devices ensures a robust evaluation of algorithm performance across varying real-world scenarios. The images have been carefully categorized into several distinct classes or categories based on specific criteria relevant to infrared image analysis.

The images in the dataset are stored in a well-defined format and are accompanied by detailed annotations that specify key characteristics of each image. For example, each annotated image is named according to a specific convention, such as 's0012_03054_0_1_0_2_1_01' as shown in fig.5.1. This naming convention serves to encode important properties of the image, which are specified in the following manner:

- **Subject ID:** the first term (s0012) signifies the subject id of the image.
- **Image ID:** the second term signifies the image ID. Overall, this dataset consists of 84,898 images.
- **Gender [0 - man, 1- woman]:** the dataset includes demographic annotations for each image, specifying the gender of individuals depicted (male or female) in the image.

- **Glasses [0- No, 1- Yes]:** each eye image in the dataset is annotated to indicate whether the subject is wearing glasses or not.
- **Eye State [0-Closed, 1-Open]:** this property contains information regarding the two different eye states: open and closed.
- **Reflections [0-none, 1-small, 2-big]:** Reflections is classified into three separate states: no reflection, little reflection, and large reflection, based on their size.
- **Lighting Conditions [0-bad, 1-good]:** each image in the dataset is categorized into two states—'good lighting condition' and 'bad lighting condition'—based on the ambient lighting conditions present during image capture,
- **Sensor ID [01 - RealSense, 02 - IDS, 03 - Aptina]:** the dataset includes photos taken with three different sensors: the 640 x 480-pixel Intel RealSense RS 300 sensor; the 1280 x 1024-pixel IDS Imaging sensor; and the 752 x 480-pixel Aptina sensor in which images are captured.

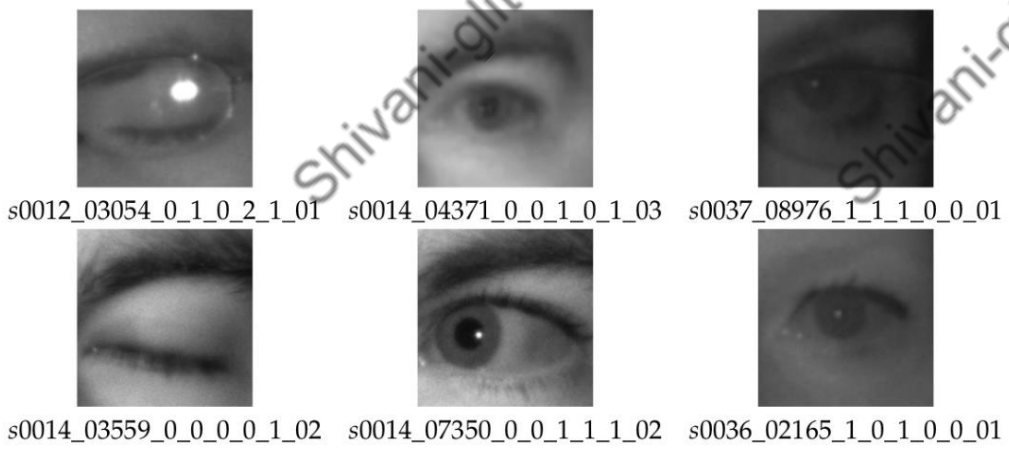


Fig. 5.1 An example of image annotations from the given dataset

Fig. 5.2 Examples of open and closed eyes extracted from the dataset

5.1.2 Screenshot of the project

This section contains a snapshot of the proposed system:

5.1.2.1 Deep Learning based Implementation

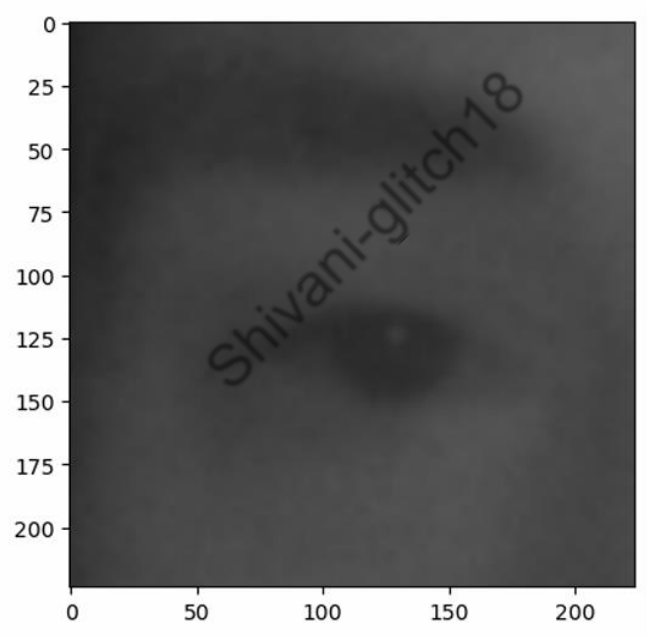
In [14]:
```
import tensorflow as tf
from tensorflow.keras import layers

# Load the MobileNet model
model = tf.keras.applications.MobileNet()
model.summary()

# Get input and output of the base MobileNet model
base_input = model.input
base_output = model.get_layer('conv_pw_13_relu').output  # Example layer name fr

# Flatten the output
flat_layer = layers.Flatten()(base_output)

# Add new layers
dense_layer = layers.Dense(1)(flat_layer)
output_layer = layers.Activation('sigmoid')(dense_layer)

# Create a new model
new_model = tf.keras.Model(inputs=base_input, outputs=output_layer)
new_model.summary()
```

```
Model: "mobilenet_1.00_224"
```

Fig. 5.3 Model Creation Code

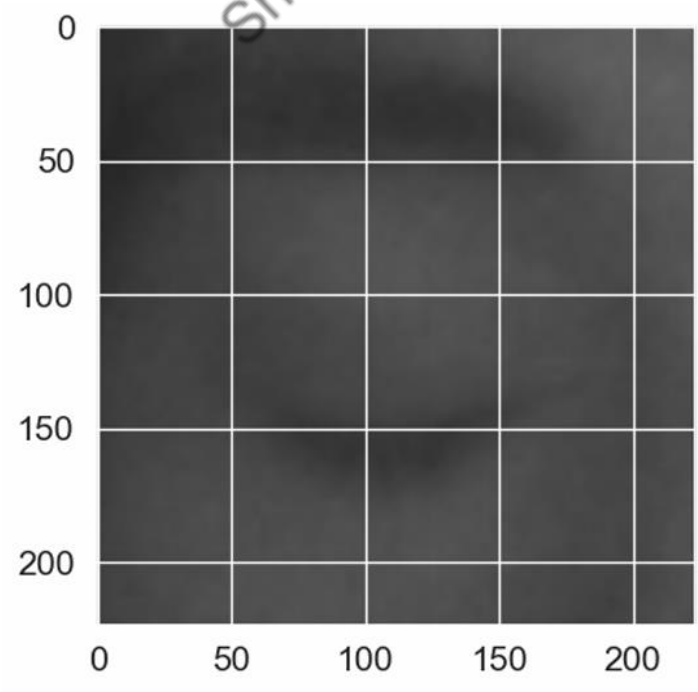
```
Epoch 21/25
1911/1911 ━━━━━━━━━━━━━━━━━━━━ 2793s 1s/step - accuracy: 0.9984 - loss: 0.0042 -
val_accuracy: 0.9900 - val_loss: 0.0404
Epoch 22/25
1911/1911 ━━━━━━━━━━━━━━━━━━━━ 2793s 1s/step - accuracy: 0.9975 - loss: 0.0094 -
val_accuracy: 0.9906 - val_loss: 0.0362
Epoch 23/25
1911/1911 ━━━━━━━━━━━━━━━━━━━━ 2791s 1s/step - accuracy: 0.9987 - loss: 0.0035 -
val_accuracy: 0.9913 - val_loss: 0.0453
Epoch 24/25
1911/1911 ━━━━━━━━━━━━━━━━━━━━ 2794s 1s/step - accuracy: 0.9986 - loss: 0.0039 -
val_accuracy: 0.9898 - val_loss: 0.0516
Epoch 25/25
1911/1911 ━━━━━━━━━━━━━━━━━━━━ 2794s 1s/step - accuracy: 0.9983 - loss: 0.0049 -
val_accuracy: 0.9894 - val_loss: 0.0422
```

Fig. 5.4 Model Training




```
In [30]: # Predict using the loaded model
         prediction = new_model.predict(X_input)
         pred=(prediction > 0.5).astype(int)
         print(pred)

1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 610ms/step
[[1]]
```

Fig. 5.5 Model eye state prediction on an open eye image




```
In [97]: # Predict using the loaded model
         prediction = new_model.predict(X_inputs)
         pred=(prediction > 0.5).astype(int)
         print(pred)

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 222ms/step
[[0]]
```

Fig. 5.6 Model eye state prediction on a close eye image

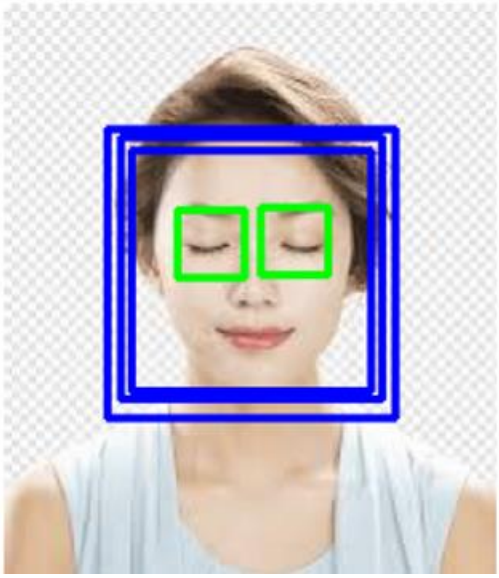


Fig 5.7 Extracting ROI in an unknown image for classification

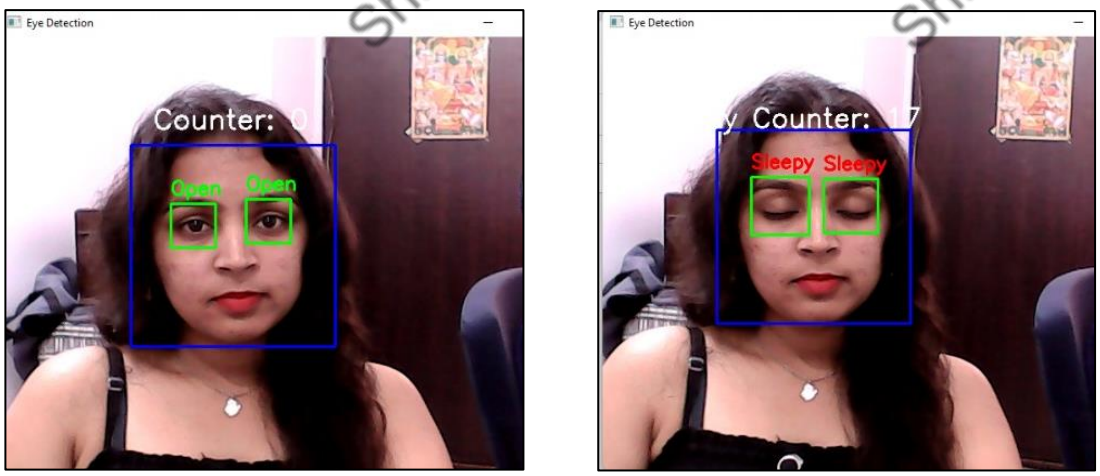


Fig.5.8 Drowsiness detection by the model in real-time

5.1.2.2 Dlib based Implementation

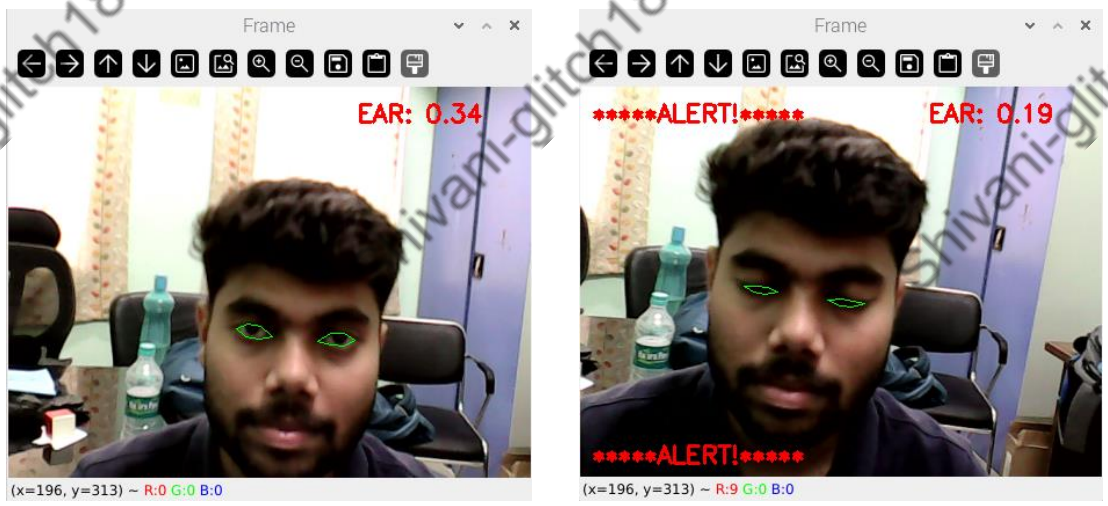


Fig.5.9 EAR Calculation for drowsiness and alarm activation

5.1.3 Limitations

Some critical limitations that impact the performance, reliability, and practicality of the developed drowsiness detection system are given below:

- The system's effectiveness may be compromised in dimly lit environments, affecting its ability to accurately detect drowsiness during nighttime or low-light driving scenarios.

- The drowsiness detection model's performance may vary due to diverse facial features, including the presence of glasses.
- Dependency on technology introduces the risk of system malfunctions or failures, emphasizing the need for driver awareness and proactive response to drowsiness cues independent of the system's operation.

CHAPTER 6

PERFORMANCE ANALYSIS

6.1 PERFORMANCE ANALYSIS

In this section these four elements helped in performing analysis for the model:

1. **True Positive (TP):**
   TP is the number of positive occurrences (e.g., drowsy state) that are accurately predicted by the model as positive.
2. **True Negative (TN):**
   TN is the number of negative occurrences (e.g., awake state) that the model accurately predicts to be negative.
3. **False Positive (FP):**
   FP is the number of negative occurrences that are incorrectly predicted by the model as positive.
4. **False Negative (FN):**
   FN is the number of positive occurrences that the model predicted incorrectly as negative.

The various parameters used for performance analysis are discussed below:

Accuracy

- Accuracy measures Accuracy quantifies the percentage of true positives and true negatives out of the total instances predicted by the model.
- Accuracy = (TP + TN) / (TP+ TN + FP + FN)
- A higher accuracy value indicates that the model is making correct predictions overall.

Precision

- Precision quantifies the percentage of actual positive predictions among all the positive predictions predicted by the model.
- Precision = (TP) / (TP + FP)
- Precision reflects the measure of the model's accuracy for positive prediction.

Recall

- Recall also known as sensitivity quantifies the percentage of true positive events that were accurately predicted by the model among all actual positive instances.
- Recall = (TP) / (TP + FN)
- Recall shows how well the model can recognize positive instances.

F1 Score

- The F1 score is defined as the harmonic mean of recall and precision.
- F1 Score = 2 * (Recall * Precision) / (Recall + Precision)
- A model that performs well in terms of recall and precision has a higher F1 score.

Confusion Matrix

A confusion matrix is a table that summarizes the numbers of true positive, true negative, false positive, and false negative predictions for each class, and is used to evaluate how well a classification model performs as shown in fig. 6.1. It enables a thorough analysis of the model's capacity to categorize instances both accurately and inaccurately. The obtained confusion matrix is displayed in fig. 6.2.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Fig. 6.1: Confusion Matrix Layout

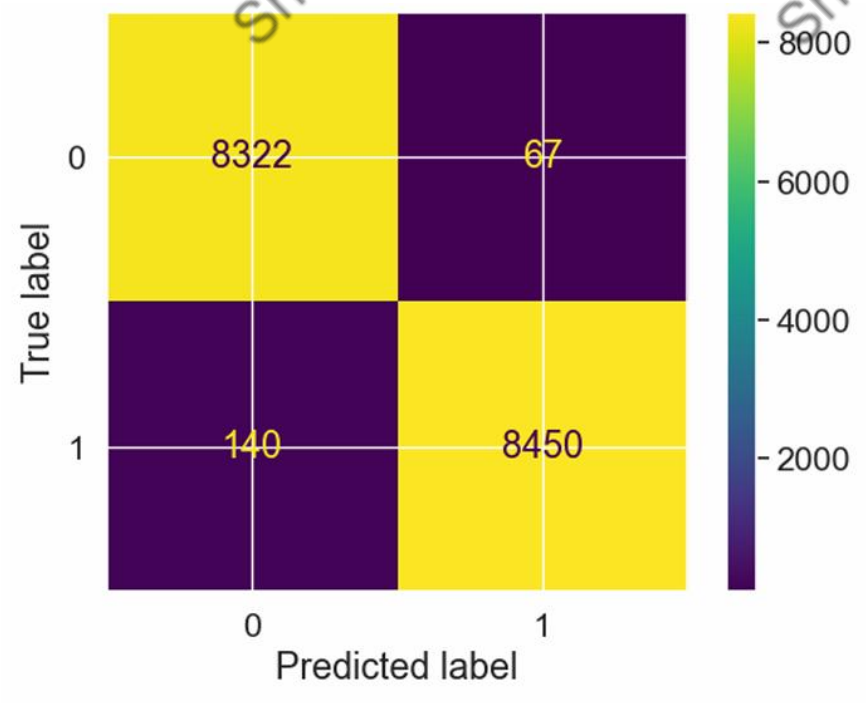


Fig. 6.2: Performance of model by confusion matrix

6.2 PERFORMANCE OF MODEL

As a part of this project, we have tried out different deep learning models such as ResNet, VGG19, MobileNet, and InceptionV3. Out of these models, MobileNet gave the best results with a test accuracy score equal to 98.78%, precision score equal to 99%, recall score equal to 99%, and F1 Score equal to 99%. The performance of the model and the obtained accuracy are shown in Table 1 and the classification report in fig. 6.3.

TABLE 1: Accuracy obtained from various models

MODELS	ACCURACY	
	TRAIN ACCURACY	TEST ACCURACY
ResNet	50.59%	50.59%
VGG19	89.79%	87.30%
Inception V3	98.29%	97.72%
MobileNet	99.89%	98.78%

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	8389
1	0.99	0.98	0.99	8590
accuracy			0.99	16979
macro avg	0.99	0.99	0.99	16979
weighted avg	0.99	0.99	0.99	16979

Fig. 6.3: Classification report of the model

6.3 RECEIVER OPERATING CHARACTERISTIC CURVE

A Receiver Operating Characteristic Curve (ROC Curve) is an important tool used for assessing the performance of classifiers. It shows how sensitivity and specificity relate to one another at various classification thresholds. The area under the curve (AUC) and the shape of a classifier are two important indicators of a classifier's discriminating power. An AUC that is closer to 1 represents better performance, whereas prediction at the chance level is indicated by an AUC of 0.5. The obtained ROC Curve is shown in fig. 6.4.

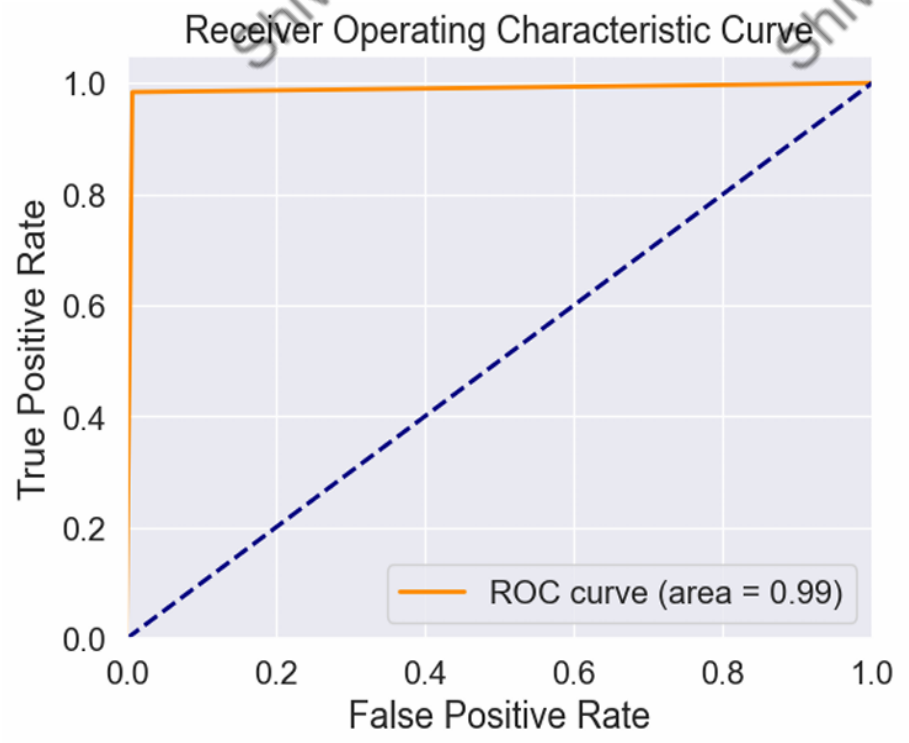


Fig. 6.4 ROC Curve obtained by the model

6.4 PRECISION-RECALL CURVE

Another essential tool for assessing classifier performance is a precision-recall curve (PR Curve), especially in situations where there is an uneven class distribution or a significant difference in the costs of false positives and false negatives. Unlike the ROC curve, which contrasts the true positive rate with the false positive rate, the PR curve emphasizes the trade-off between precision (the accuracy of positive predictions) and recall (the ability to capture positive instances). Good performance of the classifier is shown by a higher curve that is closer to the top-right corner. Overall performance is measured by the area under the PR curve (AUC-PR), where larger values denote greater precision-recall trade-offs. The PR Curve generated by the model is shown in fig. 6.5.

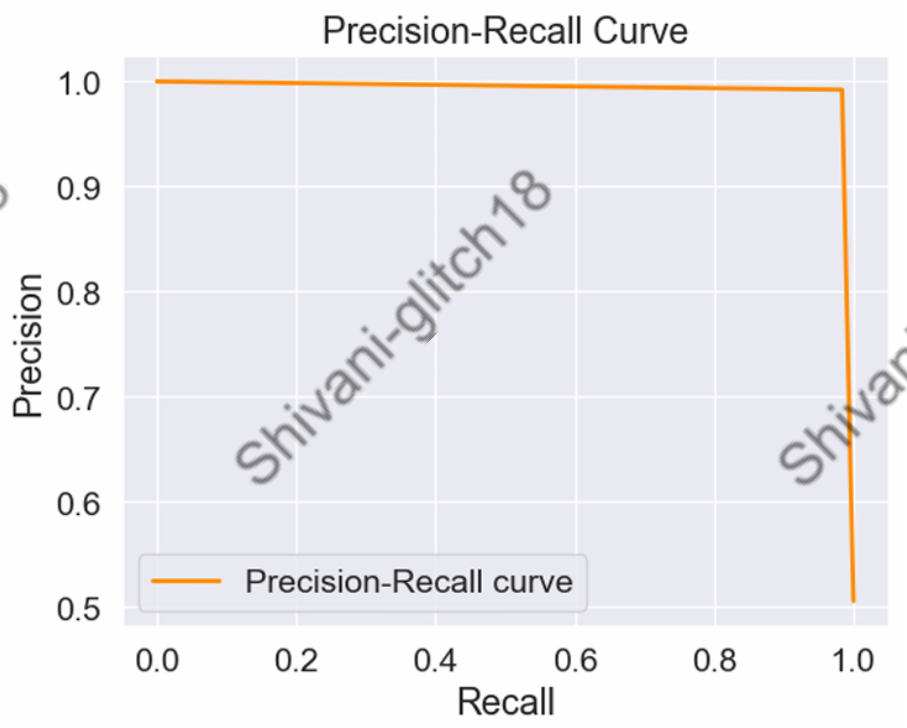


Fig. 6.5 PR Curve produced by the model

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In this project, a robust and non-invasive drowsiness detection system was developed using a combination of dlib for facial landmark detection and deep learning using CNN technology for eye state classification. The system successfully localizes facial landmarks, particularly focusing on the eyes, to monitor changes in aspect ratios indicative of drowsiness. When the system detects prolonged eye closure patterns associated with drowsiness, it issues timely warning signals to alert the driver and prompt necessary interventions. The implementation of image processing algorithms enabled accurate and reliable detection of drowsiness signals without interference or annoyance, offering a non-invasive approach to enhancing driver safety.

The results from the evaluation showcase the effectiveness of the dlib and deep learning-based approach in real-world driving conditions. While the system achieves satisfactory performance with 98.78% accuracy, there are opportunities for improvement, particularly in refining face detection algorithms and optimizing symmetry calculations for enhanced accuracy and robustness.

By further advancing and refining these technologies, our goal is to aid in the development of improved drowsiness detection systems that can notably increase transportation safety and reduce the risks linked to driver fatigue and drowsiness-related accidents.

7.2 FUTURE SCOPE

The development of the drowsiness detection system using dlib and deep learning CNN technology opens up promising avenues for future enhancements and applications which are as follows:

- Environmental Adaptability: Develop systems that perform reliably under poor to mediocre lighting conditions.
- Multimodal Sensing: Integrate diverse sensors (e.g., yawning, heart rate) for comprehensive monitoring.

- Autonomous Vehicle Integration: Enhance safety by combining detection systems with autonomous driving technologies.

REFERENCES

[1] Deng, W., & Wu, R. (2019). Real-Time Driver-Drowsiness Detection System Using Facial Features. IEEE Access, 7, 118727–118738.

[2] Gwak, J., Hirao, A., & Shino, M. (2020, April 22). An Investigation of Early Detection of Driver Drowsiness Using Ensemble Machine Learning Based on Hybrid Sensing. Applied Sciences, 10(8), 2890.

[3] B. Warwick, N. Symons, X. Chen and K. Xiong, "Detecting Driver Drowsiness Using Wireless Wearables," 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, Dallas, TX, USA, 2015, pp. 585-588, doi: 10.1109/MASS.2015.22.

[4] Mehta, S., Dadhich, S., Gumber, S., & Jadhav Bhatt, A. (2019). Real-Time Driver Drowsiness Detection System Using Eye Aspect Ratio and Eye Closure Ratio. SSRN Electronic Journal.

[5] Vesselenyi, T., Moca, S., Rus, A., Mitran, T., & Tătaru, B. (2017, October). Driver drowsiness detection using ANN image processing. IOP Conference Series: Materials Science and Engineering, 252, 012097.

[6] Prashant Dhawde, Pankaj Nagare, Ketan Sadigale, Darshan Sawant, Prof. J. R. Mahajan, 2015, Drowsiness Detection System, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) ICONECT – 2015 (Volume 3 – Issue 06).

[7] Driver Drowsiness Detection System Using Machine Learning. (2023, April 5). International Journal of Food and Nutritional Sciences, 11(12).

[8] A. Mittal, K. Kumar, S. Dhamija and M. Kaur, "Head movement-based driver drowsiness detection: A review of state-of-art techniques," 2016 IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, 2016, pp. 903-908, doi: 10.1109/ICETECH.2016.7569378.

[9] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," Proceedings. International Conference on Image Processing, Rochester, NY, USA, 2002, pp. I-I, doi: 10.1109/ICIP.2002.1038171.

[10] Vitabile, S., De Paola, A., & Sorbello, F. (2011, March 30). A real-time non-intrusive FPGA-based drowsiness detection system. Journal of Ambient Intelligence and Humanized Computing, 2(4), 251–262.

[11] B. Bhowmick and K. S. Chidanand Kumar, "Detection and classification of eye state in IR camera for driver drowsiness identification," 2009 IEEE International Conference on Signal and Image Processing Applications, Kuala Lumpur, 2009, pp. 340-345, doi: 10.1109/ICSIPA.2009.5478674.

[12] Danisman, T., Bilasco, I. M., Martinet, J., & Djeraba, C. (2013, June). Intelligent pixels of interest selection with application to facial expression recognition using multilayer perceptron. Signal Processing, 93(6), 1547–1556.

[13] L. Li, Y. Chen and Z. Li, "Yawning detection for monitoring driver fatigue based on two cameras," 2009 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, 2009, pp. 1-6, doi: 10.1109/ITSC.2009.5309841.

[14] Tanveer, M. A., Khan, M. J., Qureshi, M. J., Naseer, N., & Hong, K. S. (2019). Enhanced Drowsiness Detection Using Deep Learning: An fNIRS Study. IEEE Access, 7, 137920–137929.

[15] R. Fusek, MRL Eye Dataset, Accessed on March 1, 2024. [Online]. Available: http://mrl.cs.vsb.cz/eyedataset

[16] Chaudhari, Monali & Deshmukh, Mrinal & Ramrakhiani, Gayatri & Parvatikar, Rakshita. (2018). Face Detection Using Viola Jones Algorithm and Neural Networks. 1-6. 10.1109/ICCUBEA.2018.8697768.

[17] Yogarajan, G., Singh, R. N., Nandhu, S. A., & Rudhran, R. M. (2023, September 29). Drowsiness detection system using deep learning based data fusion approach. Multimedia Tools and Applications, 83(12), 36081–36095.

[18] Du, G., Long, S., Li, C., Wang, Z., & Liu, P. X. (2023, July). A Product Fuzzy Convolutional Network for Detecting Driving Fatigue. IEEE Transactions on Cybernetics, 53(7), 4175–4188.

[19] Bajaj, J. S., Kumar, N., Kaushal, R. K., Gururaj, H. L., Flammini, F., & Natarajan, R. (2023, January 23). System and Method for Driver Drowsiness Detection Using Behavioral and Sensor-Based Physiological Measures. Sensors, 23(3), 1292.

[20] Siddiqui, H. U. R., Saleem, A. A., Brown, R., Bademci, B., Lee, E., Rustam, F., & Dudley, S. (2021, July 15). Non-Invasive Driver Drowsiness Detection System. Sensors, 21(14), 4833.

[21] Suhaiman, A. A., May, Z., & Rahman, N. A. (2020, September 27). Development of an intelligent drowsiness detection system for drivers using image processing technique. 2020 IEEE Student Conference on Research and Development (SCOReD).

[22] Jahan, I., Uddin, K. M. A., Murad, S. A., Miah, M. S. U., Khan, T. Z., Masud, M., Aljahdali, S., & Bairagi, A. K. (2023, January 3). 4D: A Real-Time Driver Drowsiness Detector Using Deep Learning. Electronics, 12(1), 235.

[23] Raspbian Set Up Library, Accessed on March 10, 2024. [Online]. Available: https://www.raspberrypi.com/software/operating-systems/

[24] Dlib Library Installation, Accessed on March 10, 2024. [Online]. Available: http://dlib.net/

[25] Harcascade Face Detector, Accessed on March 10, 2024. [Online]. Available: https://github.com/opencv/opencv/tree/master/data/haarcascades