DRIVER DROWSINESS DETECTION SYSTEM

IT5613 SOCIALLY RELEVANT PROJECT LABORATORY

A PROJECT REPORT

Submitted by

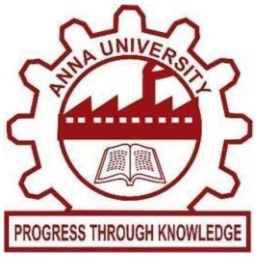
Gayathri K (2022506032)

Nishanth D (2022506048)

Amrith Eshwar T T (2022506058)

Amsa Priya K (2022506061)

Sohit Calistus A (2021506102)



DEPARTMENT OF INFORMATION TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY, CHENNAI-600 044.

FEBRUARY 2025 - JUNE 2025

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report **"DRIVER DROWSINESS DETECTION SYSTEM"** is the bonafide work of **Gayathri K(2022506032), Nishanth D (2022506048)**, **Amrith Eshwar T T(2022506058)**, **Amsa Priya K (2022506061)** and **Sohit Calistus A (2021506102)** who carried out the project work under my supervision.

SIGNATURE

Dr. M R SUMALATHA

HEAD OF DEPARTMENT

Professor

Department of Information Technology

MIT Campus, Anna University

Chennai – 600044

SIGNATURE

Dr. E PUGHAZENDHI

SUPERVISOR

Professor

Department of Information Technology

MIT Campus, Anna University

Chennai - 600044

ACKNOWLEDGEMENT

It is essential to mention the names of the people, whose guidance and encouragement made us accomplish this project.

We express our thankfulness to our project supervisor **Dr. E PUGHAZENDHI**, Professor, MIT Campus, for providing project guidance and encouragement to do the project.

We express our gratitude and sincere thanks to our respected Dean of MIT Campus, **Dr. K Ravichandran**, for providing computing facilities.

Gayathri K	**2022506032**
Nishanth D	**2022506048**
Amrith Eshwar T T	**2022506058**
Amsa Priya K	**2022506061**
Sohit Calistus A	**2021506102**

ABSTRACT

Drowsiness among drivers is a critical factor contributing to road accidents globally. The primary objective of this project is to develop a real-time, non-intrusive, and accurate drowsiness detection system utilizing computer vision and deep learning. Drowsy driving accounts for a significant percentage of fatal and non-fatal road accidents each year, and existing preventive systems are either intrusive, costly, or not reliable enough in dynamic, real-world conditions.

The proposed system aims to bridge this gap by offering a solution that leverages a standard webcam to monitor the driver's eye state using Convolutional Neural Networks (CNN). By processing live video streams using OpenCV, the system isolates the facial region and extracts the eye portion. These extracted regions are then passed through a trained CNN model to classify whether the eyes are open or closed. A scoring algorithm accumulates a drowsiness score based on how long the eyes remain closed. When the score exceeds a predefined threshold, it is inferred that the driver is drowsy, and an audible alarm is triggered using the Pygame library.

This approach is lightweight, scalable, and practical for deployment in everyday vehicles. With high training and validation accuracies, the CNN model demonstrates strong generalization, even under challenging conditions such as variations in lighting, facial orientation, and the presence of eyewear.

The implementation focuses on achieving real-time performance without requiring high-end hardware, making it suitable for integration into consumer-grade laptops, onboard vehicle systems, or even future mobile applications. The system not only alerts the driver but also serves as a proof-of-concept for embedding AI-driven safety features into conventional transport technologies. Ultimately, this project contributes to the broader goal of reducing road accidents by addressing one of the most common but preventable causes—driver drowsiness.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
ROI	Region of Interest
OpenCV	Open Source Computer Vision Library
EEG	Electroencephalogram
ECG	Electrocardiogram
HRV	Heart Rate Variability
GPU	Graphics Processing Unit
RAM	Random Access Memory
IDE	Integrated Development Environment

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Drowsy driving remains a significant contributor to traffic accidents, causing thousands of fatalities and injuries each year. As awareness grows about the dangers of fatigue behind the wheel, developing proactive technologies becomes essential. Traditional detection methods, such as electroencephalogram (EEG) sensors, while effective, are intrusive and impractical for everyday drivers. This project proposes a novel approach to detecting drowsiness using only a standard webcam and machine learning techniques. With advancements in computer vision and deep learning, particularly in real-time image processing, our system enables continuous monitoring of the driver's eye state, offering a simple yet effective way to alert drivers before accidents occur.

1.2 NEED FOR THE SYSTEM

Road safety authorities have identified drowsy driving as a contributing factor in approximately 20% of road accidents. The limitations of existing systems range from high cost, complex hardware, and intrusiveness to unreliability in uncontrolled conditions. Most of these solutions fail to cater to the needs of everyday drivers who require an affordable and easy-to-use system. This project addresses these issues by creating a non-intrusive, cost-effective, and reliable system using standard hardware like webcams and a CNN trained on custom datasets. Its simplicity and efficiency make it highly applicable for integration into consumer vehicles and potentially mobile devices.

1.3 PROBLEM STATEMENT

The objective of the Driver Drowsiness Detection System is to minimize the risk of road accidents caused by fatigue. The system continuously monitors the driver

through a webcam and uses a CNN model to evaluate the state of the driver's eyes. If the eyes are detected as closed for a prolonged duration, the system interprets this as a sign of drowsiness and triggers an alarm. The challenge lies in developing a model that operates accurately under varying real-world conditions such as lighting, eyewear, and facial features. This project aims to create a solution that performs reliably across diverse conditions without causing inconvenience to the user.

1.4 OBJECTIVE

- To capture real-time video from a webcam and process the frames for analysis.
- To detect the driver's face and accurately extract the eye regions.
- To classify the state of the eyes (open or closed) using a trained CNN model.
- To monitor the frequency and duration of closed-eye states using a scoring mechanism.
- To trigger an alarm when a drowsiness threshold is crossed, thereby alerting the driver.
- To ensure the system functions with minimal latency and high accuracy in real-time scenarios.

1.5 SCOPE

The scope of this project encompasses the use of image processing and machine learning for the real-time detection of drowsiness. It is limited to detecting drowsiness through the eye state of the driver but can be expanded to include additional indicators like yawning or head tilting in future enhancements. The system is designed to work on general-purpose hardware with a webcam and does not rely on specialized sensors. It can be deployed in personal vehicles, integrated into fleet monitoring systems, or ported to mobile devices for broader accessibility. The primary focus remains on maximizing detection accuracy while minimizing resource usage and intrusiveness.

CHAPTER 2

LITERATURE SURVEY

The evolution of driver drowsiness detection systems has seen multiple approaches across various domains, including physiological monitoring, vehicular behavior analysis, and image-based detection. A detailed literature review highlights the strengths and weaknesses of each method, guiding the development of our proposed system.

2.1 Physiological Monitoring Methods

Physiological signals such as brainwaves (EEG), heart rate (ECG), and skin conductivity (EDA) offer high accuracy in detecting fatigue. These systems involve the use of sensors directly attached to the driver's body to monitor internal changes associated with drowsiness. Although they provide real-time, reliable data, they are not feasible for practical implementation in regular vehicles due to their intrusiveness, cost, and need for medical-grade hardware.

2.2 Vehicle Behavior-Based Detection

This method relies on monitoring deviations in vehicle dynamics such as lane drifting, steering behavior, and acceleration patterns. These systems utilize built-in vehicle sensors to infer driver alertness. While non-intrusive, these techniques are limited by environmental factors like road type, weather, and traffic conditions. They also struggle with precision when the vehicle is being driven in a steady, straight path for long durations.

2.3 Image-Based Detection Using Classical Techniques

Classical computer vision approaches like the Viola-Jones Haar Cascade algorithm are widely used for face and eye detection in real-time. These methods are lightweight and computationally efficient, making them suitable for embedded applications. However, they lack robustness under varying lighting conditions and are sensitive to occlusions such as spectacles and head movements. Their reliance on hand-crafted features limits their adaptability.

2.4 Deep Learning and CNN-Based Detection

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision. In the context of drowsiness detection, CNNs can learn complex patterns and features directly from raw eye images, improving detection accuracy. Studies show that CNN-based systems outperform traditional methods in classifying eye states under diverse conditions, including low light and different ethnicities. Their flexibility allows them to be trained on custom datasets, enabling tailored solutions for specific environments.

2.5 Hybrid and Multi-Modal Approaches

Recent advancements incorporate hybrid models that combine facial recognition, head pose estimation, and even voice analysis for a more comprehensive understanding of driver fatigue. Although promising, these systems often demand higher computational resources and more complex integration, making them less feasible for low-cost deployment.

2.6 Research Gaps and Motivation

Despite significant progress, challenges persist. Many existing systems suffer from high false alarm rates, low generalizability, and hardware dependence. There is a lack of lightweight, real-time systems that can perform effectively in everyday vehicles. This project aims to address these gaps by using a lightweight CNN model optimized for real-time eye state detection, implemented with widely accessible tools such as OpenCV and Python.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 SYSTEM OVERVIEW

The system follows a client-side architecture utilizing standard hardware components—a webcam and a general-purpose computing device (e.g., a laptop or desktop). It leverages Python and its ecosystem of libraries for implementation. The pipeline consists of five major modules: video acquisition, face and eye detection, eye state classification via CNN, drowsiness scoring, and alarm activation

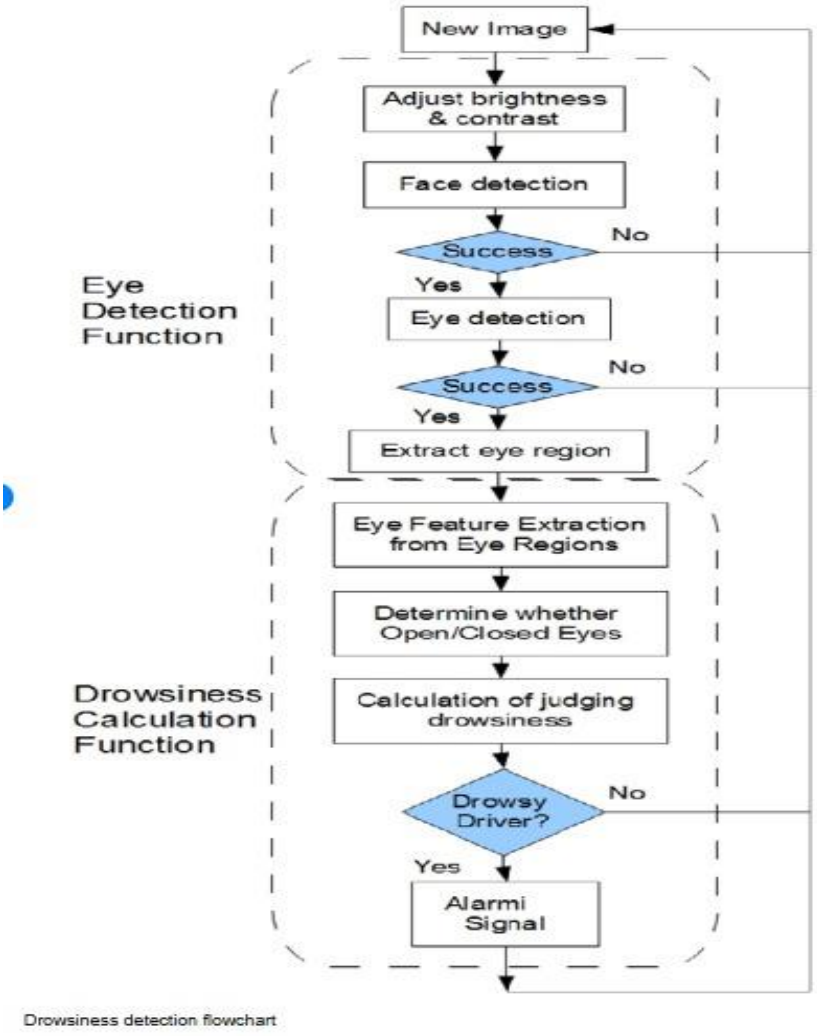


Figure 3.1 – System Architecture

3.2 DATA FLOW

1. **Video Frame Acquisition**: Live video frames are captured using the webcam at approximately 15-30 frames per second (FPS), ensuring smooth real-time analysis.
2. **Face & Eye Detection**: Each frame is converted to grayscale and passed through Haar Cascade classifiers to detect the face and eye regions. The Region of Interest (ROI) containing the eyes is extracted and resized.
3. **Eye State Classification**: The extracted eye images are preprocessed and fed into a trained Convolutional Neural Network. The CNN outputs a binary classification—'0' for closed eyes and '1' for open eyes.
4. **Scoring Mechanism**: A score is maintained and incremented for each frame where the eyes are detected as closed. If the score exceeds a threshold (e.g., 15 consecutive closed-eye frames), it is interpreted as a sign of drowsiness.
5. **Alarm Activation**: Once the drowsiness condition is met, the alarm module (using Pygame) is triggered to play a sound alert, notifying the driver.

3.3 COMPONENT INTERACTION

- The system's modular structure ensures that each component performs an isolated function, which improves debuggability and scalability.
- The classifier module is decoupled from the detection pipeline, allowing model upgrades or re-training without altering the video processing logic.
- The scoring and alarm modules are kept lightweight to minimize lag and ensure immediate response upon detecting drowsiness.

.

3.4 FRONTEND COMPONENTS

3.4.1 Video Frame Acquisition

- Utilizes OpenCV to interface with the webcam.
- Captures frames continuously at ~15–30 FPS.
- Converts each frame to grayscale to simplify processing.

3.4.2 Face and Eye Detection

- Implements Haar Cascade Classifiers for real-time detection.
- Once the face is located, the system extracts ROIs (Regions of Interest) corresponding to both eyes.
- These regions are resized (e.g., to 24×24 px) and normalized to match the CNN's input shape.

3.4.3 Alarm Notification System

- Uses the Pygame library to trigger sound-based alerts.
- Ensures immediate feedback upon detection of drowsiness without interrupting the main detection pipeline.
- Optionally allows customization of sound alerts, duration, and intensity.

3.5 BACKEND COMPONENTS

3.5.1 CNN-Based Eye State Classification

- A lightweight Convolutional Neural Network trained on custom eye image datasets.
- Input: Preprocessed eye image.
- Output: Binary prediction (1 = open, 0 = closed).
- Frameworks used: Keras (API), TensorFlow (backend).

3.5.2 Scoring and Drowsiness Logic

- A counter increments when the eyes are classified as closed for successive frames.
- Threshold (e.g., 15 consecutive frames closed) is customizable.
- Once threshold is crossed, drowsiness is assumed and an alarm is triggered.
- The score resets if eyes reopen.

3.5.3 Model Management

- CNN model is stored and loaded using Keras's .h5 file format.
- Optimized for inference using precompiled weights.
- Model can be updated or retrained independently of the main application.

3.5.4 Optional: Logging/Storage Backend

- While not implemented in this iteration, the backend design supports extensions like:
    - Logging drowsiness incidents.
    - Timestamping and storing eye state history.
    - Uploading performance statistics to a cloud database (e.g., Firebase, MongoDB).
- This can help track driver behavior over time or aid research purposes**.**

3.6 PERFORMANCE CONSIDERATIONS

- The CNN architecture is intentionally shallow to ensure inference within milliseconds per frame.
- Frame-skipping and grayscale conversion reduce computational load.
- Modular threading ensures real-time responsiveness, especially for the alarm.

CHAPTER 4

ABOUT THE PROJECT

4.1 USE CASE DIAGRAM

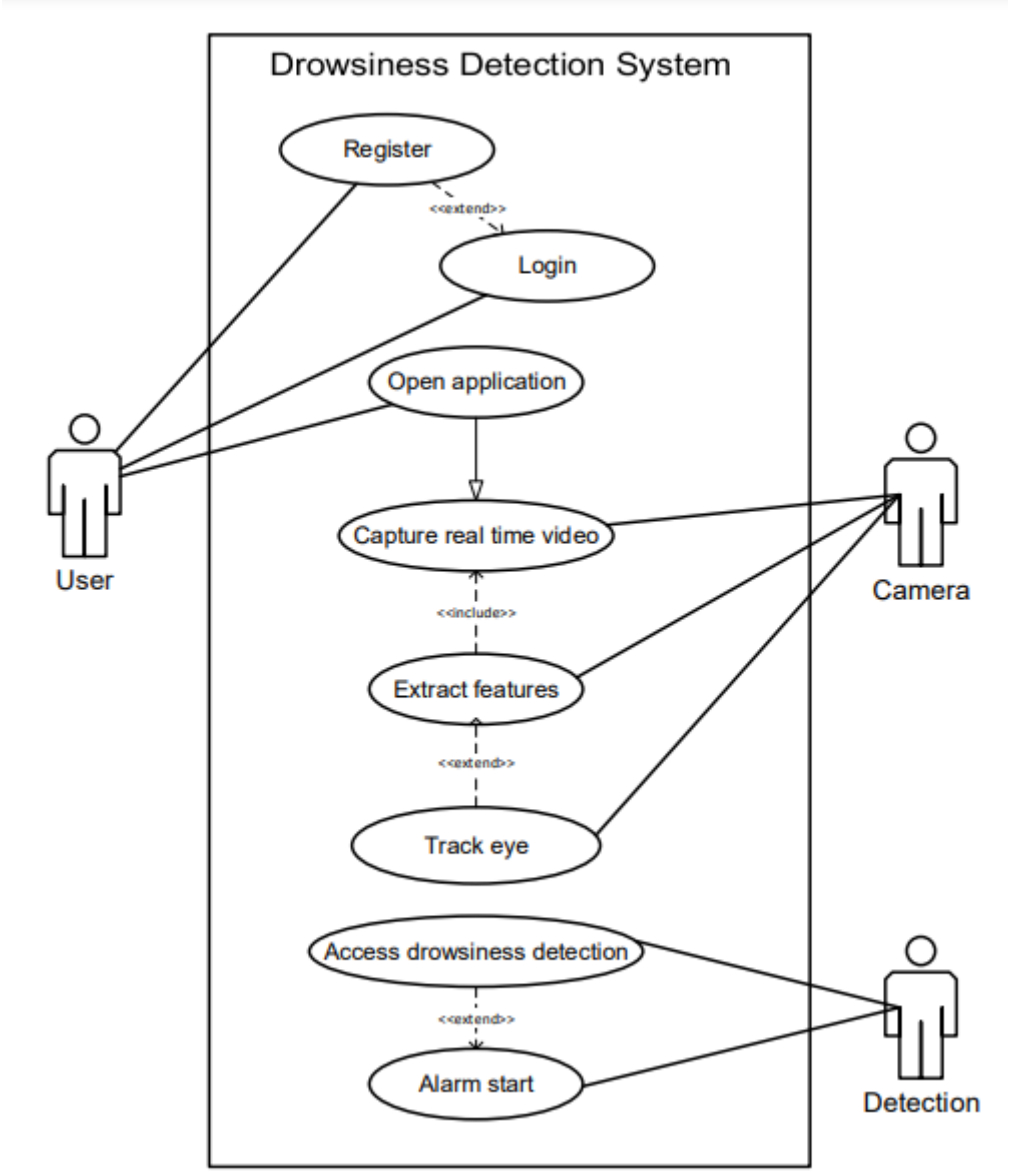


Figure 4.1 – Use Case Diagram

4.2 PROJECT MODULES

4.2.1 IMAGE CAPTURE MODULE

The first step in the detection pipeline is capturing real-time video from the webcam. This module interfaces with the hardware to:

- Continuously fetch frames from the webcam at a reasonable frame rate (15–30 FPS).
- Convert each captured frame to grayscale to reduce processing complexity.
- Pass the frame to the detection module for further processing.

This module ensures uninterrupted real-time feed with minimal latency and handles exceptions like camera disconnection or feed errors gracefully.

4.2.2 FACE AND EYE DETECTION MODULE

This module uses Haar Cascade classifiers provided by OpenCV to detect the driver's face and subsequently extract the eye regions (left and right eyes).

- Face Detection: Identifies the bounding box around the face from the grayscale frame.
- Eye Region Localization: From the face bounding box, two eye ROIs (Regions of Interest) are extracted.
- These ROIs are preprocessed by resizing (typically to 24×24 pixels), normalizing pixel values, and reshaping them to match the CNN input dimensions.

This module is performance-critical and optimized for speed and accuracy in various lighting conditions.

4.2.3 EYE STATE CLASSIFICATION MODULE (CNN)

This module classifies each eye image as either ‘Open’ or ‘Closed’ using a trained Convolutional Neural Network (CNN).

- Input: Preprocessed image of the eye.
- Output: Binary classification – 0 (Closed) or 1 (Open).
- Model: A lightweight CNN trained on a custom dataset comprising thousands of eye images under varied conditions.

The classifier is built using Keras with TensorFlow backend and is stored as a .h5 model file, loaded dynamically at runtime. It allows easy re-training or fine-tuning for enhanced accuracy.

4.2.4 DROWSINESS SCORE TRACKER

This module maintains a score to quantify driver fatigue based on the output of the eye state classifier.

- For each frame where eyes are classified as ‘Closed,’ the score is incremented.
- For frames where eyes are ‘Open,’ the score is decremented or reset.
- If the score crosses a predefined threshold (e.g., 15), it indicates sustained eye closure — a sign of drowsiness.

This logic allows the system to distinguish between brief eye blinks and actual drowsy states, reducing false positives.

4.2.5 ALARM TRIGGER MODULE

This module serves as the primary feedback mechanism, alerting the driver when drowsiness is detected.

- Uses the Pygame library to play an audible alarm.
- The alarm sound can be customized for duration and volume.
- Runs in a separate thread or asynchronous process to avoid blocking the main detection pipeline.

This ensures the driver is alerted instantly without interrupting ongoing detection operations.

4.2.6 Optional: LOGGING AND ALERT HISTORY MODULE

Though not mandatory in the initial prototype, this module can be added to:

- Log each instance of drowsiness detection with a timestamp.
- Store the number of times an alarm was triggered.
- Enable long-term analysis of driver fatigue patterns.
- Store logs locally or in a remote database (e.g., Firebase, MongoDB).

This feature enhances the system's usefulness in fleet management or scientific research.

4.2.7 CONTROL INTERFACE MODULE (Start/Stop)

This module allows the driver or system user to:

- Start the monitoring session (initializing camera, model, and detection loop).
- Stop monitoring and safely release system resources.
- Display system status on a simple graphical or command-line interface.

It offers a user-friendly way to operate the system with basic interaction, improving usability.

4.2.8 ERROR HANDLING AND FAILSAFE MODULE

This background module ensures the reliability of the system by:

- Detecting missing webcam feed or unreadable frames.
- Handling CNN model loading errors.
- Managing unresponsive modules or unexpected behaviors gracefully.
- Providing console messages or logs for debugging and system health monitoring.

4.3 DATA FLOW DIAGRAM

4.3.1 Stock Flow

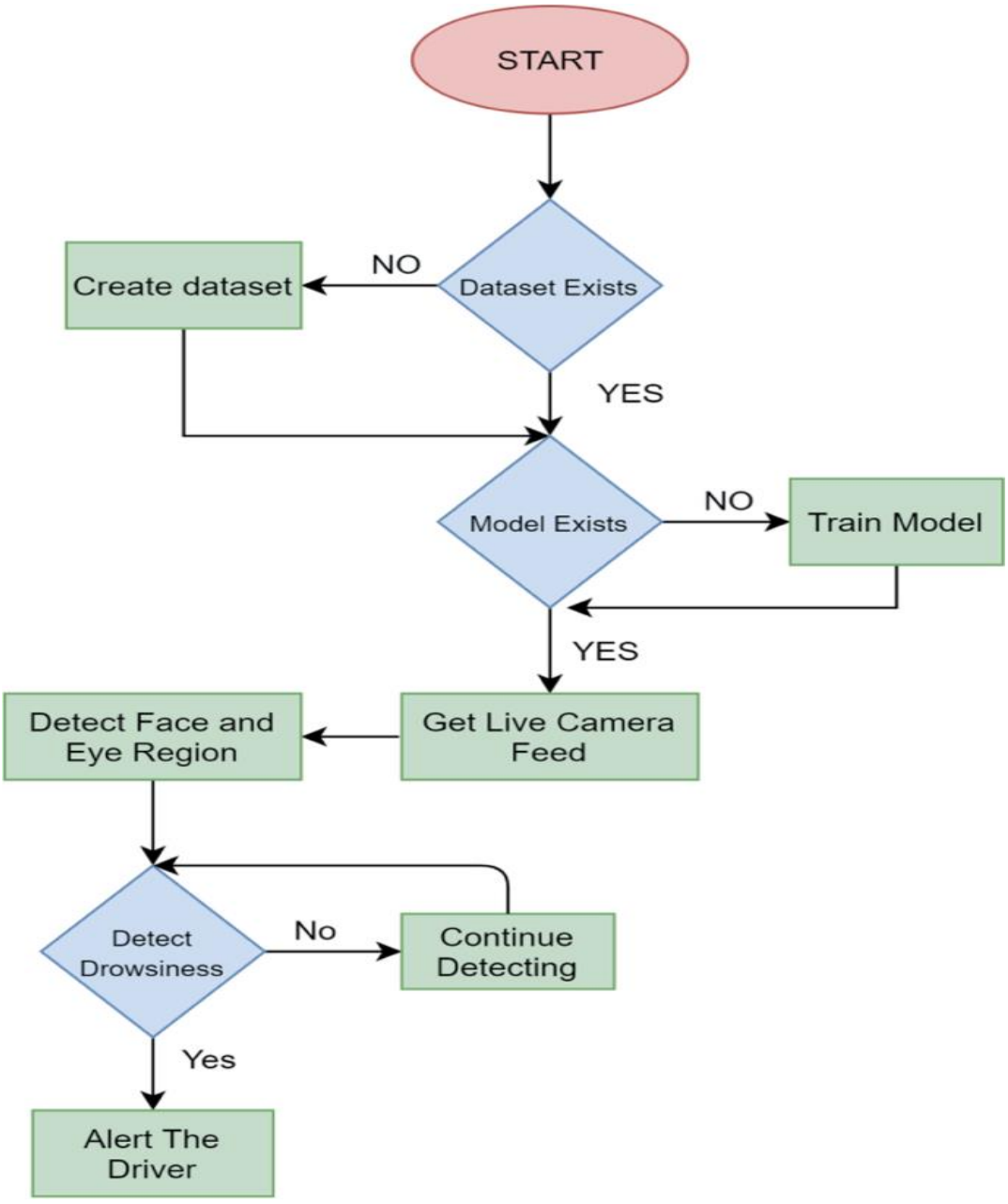


Figure 4.2 – Stock Flow Diagram

4.3.2 User Flow

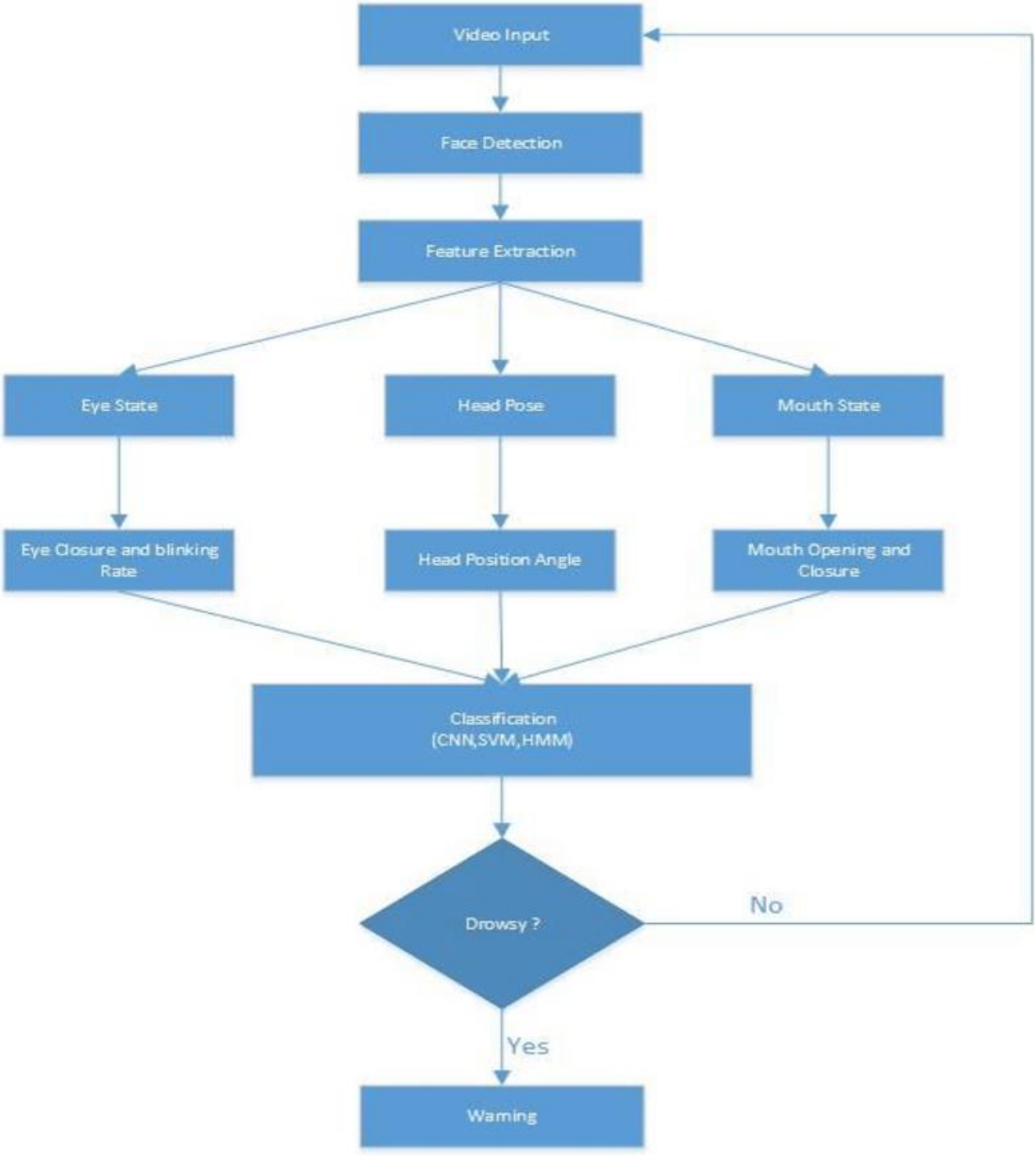


Figure 4.3 – User Flow Diagram

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

The Driver Drowsiness Detection System developed through this project represents a critical step toward enhancing road safety using computer vision and deep learning technologies. With drowsy driving contributing significantly to global road accidents, there is an urgent demand for intelligent systems capable of alerting drivers before they lose control due to fatigue. This project successfully addresses that need by offering a lightweight, real-time, and cost-effective solution that does not rely on intrusive or expensive hardware.

Throughout the development cycle, the system demonstrated the viability of using widely available tools such as OpenCV for image processing, Keras and TensorFlow for model inference, and Pygame for alert generation. The modular architecture ensures scalability and adaptability, allowing future developers to enhance or integrate additional features such as yawning detection, head pose estimation, or even real-time logging and cloud connectivity.

The implemented Convolutional Neural Network achieved impressive accuracy in classifying eye states, even under varied lighting and facial orientations. By adopting a threshold-based scoring system, the solution was able to differentiate between natural blinking and prolonged eye closure—a crucial feature that reduces false positives and increases driver trust in the system.

Moreover, the design supports real-time operation on standard computing hardware, including laptops and embedded devices, making it practical for real-world deployment. The ease of integration with existing dashboard systems or mobile platforms further increases its commercial potential and societal impact.

REFERENCES

1. OpenCV: https://docs.opencv.org/

2. Keras https://keras.io/

3. TensorFlow https://www.tensorflow.org/

4. Pygame https://www.pygame.org/docs/

5.Python https://docs.python.org/3/

6.Real-Time Face and Eye Detection with OpenCV https://learnopencv.com/face-and-eye-detection-with-opencv/

7. Pygame Alarm Implementation https://realpython.com/pygame-a-primer/

8. Closed Eyes In The Wild (CEW Dataset)
http://www.cbsr.ia.ac.cn/english/ClosedEyeDatabases.asp