FATIGUE DETECTION SYSTEM IN CARS

BY

VANICHA KULMA

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONOURS) COMPUTER ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project proposal is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Technology (Honours) Computer Engineering at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Ts Dr Vikneswary a/p Jayapal, for guiding me through this Final Year Project titled "Fatigue Detection System in Cars." Her valuable advice, clear explanations, and patience made a huge difference in completing this work successfully.

I'm also grateful to the lecturers and lab assistants from the Faculty of Information and Communication Technology. Their assistance and availability whenever I faced difficulties really helped smooth the project's progress.

Additionally, I wish to thank my family and friends who were always there to support and encourage me, especially when things became stressful or challenging.

Finally, working on this project was an excellent learning opportunity. It improved my practical skills, expanded my technical understanding, and gave me confidence to take on future challenges in my career.

ABSTRACT

Fatigue among drivers continues to be one of the leading causes of road accidents particularly when people are driving for long hours, during the night, or in conditions that demand high focus. Even though technology has come a long way, a lot of cars still don't come with proper systems that can detect when the driver is starting to lose focus or show early signs of drowsiness. That is what motivated this project – to come up with a solution that can identify fatigue symptoms through facial cues in real time and alert the driver before things get dangerous.

The project uses a lightweight CNN called Mobilenet to perform real-time image-based detection efficiently on a compact device like the Raspberry Pi. It integrates EAR and MAR to detect eye closure and yawning, while also tracking blink frequency, which can help identify early signs of drowsiness, Facial landmarks are used to extract these features from the driver's face using a webcam. To further enhance reliability, the system includes head tilt detection by calculating pitch and roll angles to recognize unnatural head positions commonly associated with fatigue.

Special attention is given to real-world challenges such as low lighting conditions, reflection from spectacles, and obstructions that may affect facial visibility. These factors can impact detection accuracy, so the system is designed to be robust and adaptable in various driving environments.

Overall, this project is more than deep learning experiment. It integrates computer vision, facial analysis, real-time processing, and system integration to deliver a lightweight and practical solution. The system is cost-effective, easy to deploy, and reliable enough to help reduce the risk of accidents caused by driver fatigue.

Area of Study (Maximum 2): Fatigue Indicators

Keywords (Maximum 10): Driver Fatigue Detection, Computer Vision, Deep Learning, Facial Landamark, Real time Monitoring, Embedded Platform

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
EAR	Eye Aspect Ratio
MAR	Mouth Aspect Ratio
FPS	Frames Per Second
ROI	Region of Interest

Chapter 1

Introduction

In this chapter, I will briefly introduce the idea behind this project, along with the reasons why it was chosen. Fatigue is something many driver face, especially during long or late-night drive, and it often leads to accident when it goes unnoticed. Although some modern cars have safety features, not all of them include proper fatigue detection systems. This project focuses on creating a system that can recognize signs like eye closure, yawning, blinking patterns, and head tilt using simple hardware and deep learning methods. The chapter also explains the problem being addressed, the main goals, the direction of the project, the contributions made, and how this report is structured.

1.1 Problem Statement and Motivation

Fatigue-related driving accidents remain a major safety concern worldwide. Although many fatigue detection systems have been proposed, a significant number either depend on expensive specialized hardware [1] or fail to maintain high accuracy under real-world conditions [2]. Environmental factors such as low lighting, reflections caused by spectacles, and the presence of face masks often disrupt accurate facial feature detection [3]. Furthermore, critical fatigue indicators like frequent blinking, yawning, and subtle head tilting are sometimes missed, resulting in delayed warnings and reduced system effectiveness.

These limitations reveal a gap in current driver monitoring solutions, particularly for affordable systems intended for standard vehicles with limited computational resources.

Motivated by these challenges, this project aims to develop a real-time fatigue detection system that is accurate, lightweight, and robust across various conditions. The proposed method integrates MobileNetV2-based deep learning with traditional facial landmark techniques (EAR, MAR, and head pose estimation) to monitor key indicators such as eye closure, blinking frequency, yawning, and head posture. The goal is to create a practical and cost-effective solution that enhances road safety by reliably detecting fatigue early and issuing timely alerts to drivers.

1.2 Objectives

- To develop a real-time fatigue detection system using Python and deep learning on an embedded platform like Raspberry Pi.
- To integrate deep learning with facial landmark detection (EAR and MAR) for accurate fatigue monitoring in various conditions.
- To implement a responsive alert system that activates sound warnings when drowsiness is detected.

1.3 Project Scope and Direction

This project focuses on developing a real-time fatigue detection system that operates on an embedded platform, specifically the Raspberry Pi. The system is built using Python and deep learning to enable efficient processing within hardware constraints. A webcam is used to continuously capture the driver's facial features, which are analyzed to detect signs of fatigue.

The core detection method integrates a lightweight deep learning model with facial landmark analysis, using Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) to monitor eye closure, yawning, and blink frequency. For head posture detection, the system applies the solvePnP method to estimate the orientation of the driver's head and detect abnormal downward tilting, which may indicate drowsiness. The system is designed to perform reliably in varying conditions, including low-light environments and when the driver is wearing spectacles.

In addition, the system includes a sound-based alert feature that activates when signs of fatigue are detected, helping to regain the driver's attention. The overall goal is to create a practical, low-cost, and standalone solution that improves road safety through accurate and timely fatigue monitoring.

1.4 Contributions

This project contributes to the advancement of road safety by developing a fatigue detection system that is both practical and efficient for in-vehicle use. A major contribution is the integration of MobileNet with facial landmark analysis and classical methods such as the EAR and MAR. This combination enables robust, real-time detection of fatigue indicators including eye closure, yawning, blink frequency, and head posture. The use of solvePnP for head orientation provides an additional layer of reliability, allowing the system to identify downward or tilted head movements that are strongly associated with drowsiness.

Another key contribution lies in the system's capability to operate effectively under challenging conditions, such as poor lighting and when the driver is wearing spectacles. Furthermore, the project demonstrates that deep learning and computer vision techniques can be deployed efficiently on a compact embedded platform like the Raspberry Pi, eliminating the need for high-end computing hardware. Finally, the inclusion of an in-car sound alert system ensures immediate feedback when signs of fatigue are detected, providing timely warnings that may help prevent accidents.

1.5 Report Organization

This report is structured into seven chapters. Chapter 1 introduces the project by presenting the problem statement, objectives, scope, and contributions. Chapter 2 provides the literature review, covering related technologies, existing systems, and previous approaches to fatigue detection. Chapter 3 describes the methodology, including the system model, design diagrams, and overall approach. Chapter 4 details the system design, explaining the block diagram, component specifications, and interactions between modules. Chapter 5 discusses the implementation process, including hardware setup, software configuration, and system operation. Chapter 6 presents the system evaluation and discussion, focusing on testing, results, and performance analysis. Finally, Chapter 7 concludes the report by summarising the project outcomes and offering recommendations for future improvements.

Chapter 2

Literature Review

2.1 Previous Works on Driver Fatigue Detection Techniques

2.1.1 MobileNet: Deep Learning-Based Driver Fatigue Detection

MobileNet, especially its MobileNetV2 variant, has become a popular deep learning model for driver fatigue detection due to its lightweight architecture and suitability for real-time deployment on embedded platforms. By utilizing depthwise separable convolutions, MobileNetV2 significantly reduces model size and computational demands while maintaining high accuracy, making it ideal for systems running on devices like the Raspberry Pi.

In the study by Wagh et al., MobileNetV2 was trained using the MRL Eye dataset to detect eye closure and trigger alerts when drowsiness was detected. A Haar cascade classifier was used to identify facial regions, and MobileNetV2 classified eye states based on video input from a webcam. The system was designed to issue warnings via buzzer and simulate emergency actions like slowing down the vehicle if the driver remained unresponsive. The approach achieved an impressive accuracy of 99.33%, showing high reliability in various lighting conditions [2].

Another study conducted by Wunan et al. evaluated MobileNetV2 alongside NASNet Mobile and EfficientNetB0 using a subset of the MRL Eye dataset. MobileNetV2 recorded a training accuracy of 100%, validation accuracy of 95%, and testing accuracy of 94%. Despite EfficientNetB0 scoring slightly higher, MobileNetV2 was favored for its smaller model size of 14 MB and lower training time, making it highly efficient for embedded applications [3].

In a separate paper focusing specifically on performance benchmarking, MobileNetV2 was trained on a larger dataset with over 41,000 labeled images categorized as “Drowsy” or “Non-Drowsy.” The model achieved training accuracy of 99.97% and test accuracy of 99.86%. The study emphasized MobileNetV2’s strong generalization across different lighting conditions and face appearances, demonstrating its potential for integration into real-time, in-vehicle systems [1].

In the thesis by Dongjiang Wu, MobileNet was applied to a driver monitoring system to detect both fatigue and distraction. The method involved processing facial images to classify states

such as eye closure, mouth movement, and distracted behaviors like phone usage. The system was trained using both public and self-collected datasets and was designed to operate in real time with optimized accuracy for practical driving scenarios [7].

In summary, MobileNetV2 achieves accuracy rates between 94% and 99.86%, making it a reliable model for fatigue detection. Its small size, fast training, and suitability for embedded systems make it ideal for real-time driver monitoring in various conditions.

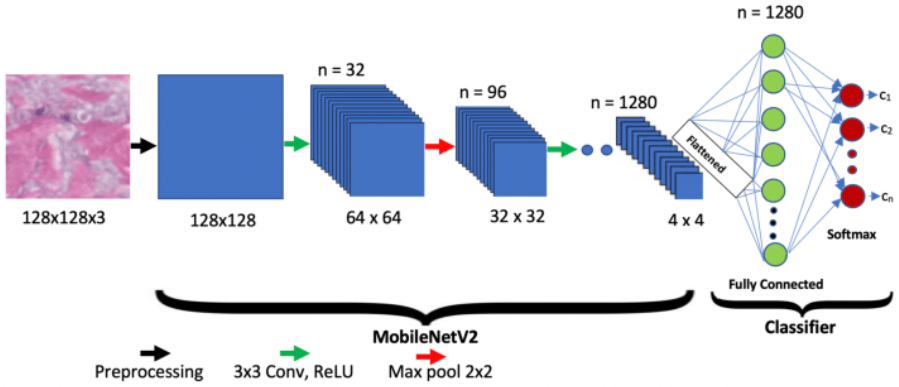
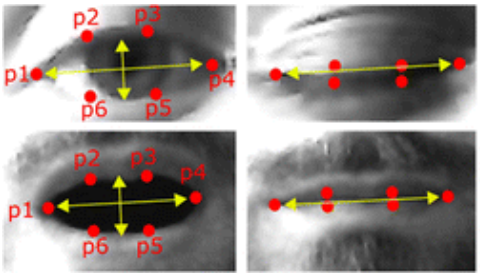


Figure 2.1: MobileNetV2 architecture

2.1.2 Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR)

The EAR and MAR are commonly used in fatigue detection to monitor eye closure and yawning. These values are calculated from facial landmarks and compared against fixed thresholds to detect signs of drowsiness in real time. EAR values below 0.25–0.3 indicate eye closure, while MAR values above 0.9 suggest yawning.



$$\mathrm{EAR} = \mathrm{MAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 2.2: EAR and MAR calculation using facial landmarks

Nizar et al. developed a system using OpenCV and Dlib's 68 facial landmarks to monitor EAR and MAR in real time. The system triggered an alarm when signs of fatigue were detected and

performed well under different lighting conditions. It achieved an accuracy of 95% during the day and 85% at night, confirming its effectiveness even when the driver wore spectacles [4].

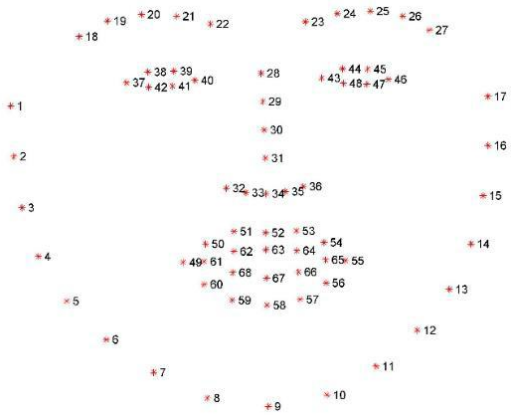


Figure 2.3: Dlib's 68 points model

Similarly, Awasekar et al. implemented an EAR and MAR-based system using a Raspberry Pi and a NoIR camera for enhanced low-light performance. Their threshold values were 0.3 for EAR and 0.9 for MAR, determined through testing to minimize false positives. Their system reached 100% accuracy in eye closure detection and an average of 79% accuracy for yawning, supported by visual and audio alerts along with emergency SMS notifications [5].

Another implementation presented on a Raspberry Pi platform showed effective use of EAR and MAR for triggering buzzer alerts when threshold values were crossed. While specific accuracy values were not reported, the system demonstrated practical performance in real-time fatigue detection [6].

Overall, EAR and MAR offer a lightweight and effective solution for non-intrusive driver fatigue detection. Their simplicity, real-time capability, and proven accuracy make them well-suited for embedded systems and practical in-vehicle use.

2.1.3 Eye Blinking Frequency in Fatigue Detection

Eye blinking frequency is a reliable indicator for detecting early signs of driver drowsiness. A decrease in blink rate or an increase in eye closure duration often reflects reduced alertness. This method is non-intrusive and can be implemented using either image-processing techniques or infrared sensors.

Danisman et al. introduced a system that detects eye blinks using horizontal symmetry analysis from webcam footage. The system identifies changes in eye symmetry to classify open and closed eye states, allowing the measurement of blink durations. Their method achieved 94.8% accuracy, 90.7% precision, and a 1% false positive rate when tested on the JZU eye-blink database, which included varied lighting conditions and subjects wearing glasses [8].

Leopoldo et al. developed a device using an IR-based eye blink sensor integrated with an Arduino. The system monitored blink rate and eye closure time, triggering a buzzer and a vibrating pillow when abnormal patterns were detected. It classified drowsiness when eye closure exceeded 2–3 seconds or when blinking rate fell outside the 12–19 blinks per minute range. The device reached an accuracy of 80% and precision of 85% across 20 trials [9].

Another approach combined blink frequency monitoring with head posture tracking. The system defined blink duration thresholds (e.g. above 400 ms) to classify drowsy behavior and issued alerts accordingly. It performed reliably under normal webcam input and maintained functionality during moderate face movement [10].

In conclusion, blink frequency and closure duration are effective fatigue indicators. With reported accuracies up to 94.8%, they serve as dependable inputs for real-time drowsiness detection systems.

2.1.4 Head Pose Estimation Techniques

Head pose estimation is a critical component in visual-based driver fatigue detection systems. It helps track orientation changes such as nodding or turning away, which often signal drowsiness or distraction. Techniques like solvePnP and facial landmark-based models are commonly used to estimate pitch, yaw, and roll angles for real-time posture monitoring.

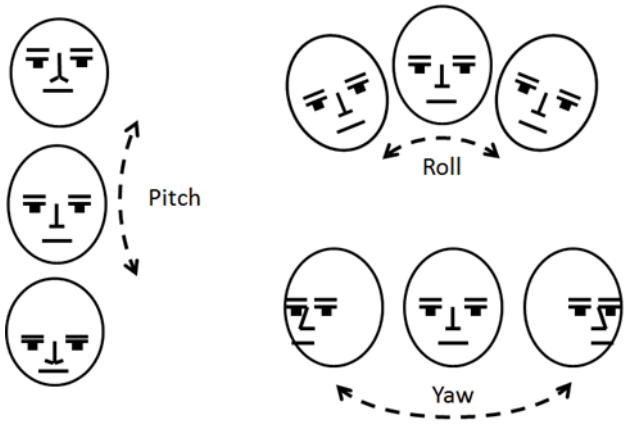


Figure 2.4: Illustration of Head Pose Angles

Meng Zhang et al. proposed a fatigue detection system that integrates head pose estimation to correct errors in EAR calculation caused by face rotation. Using solvePnP, the system determines pitch and yaw angles and compensates EAR accordingly to reduce false positives. Their experiment showed that applying head pose correction improved EAR accuracy from 89.85% to 94.94%, enhancing system performance under varying head orientations [11].

Wang et al. proposed a head movement–based fatigue detection method that estimates driver head posture using Euler angles derived from the solvePnP algorithm. The system focuses on monitoring head-down behavior as a key indicator of fatigue. Two thresholds are defined to classify fatigue severity: a head-down time greater than 3 seconds and a head-down frequency of 4 times or more are both indicators of deep fatigue. These posture-based parameters are fused with other features using a fuzzy logic approach to improve classification accuracy under real driving conditions [12].

Chadha et al. applied solvePnP with 2D–3D point mapping for head pose estimation. Their system combined this pose data with facial features like eye and mouth states to trigger alerts. While no numerical accuracy was specified, the method demonstrated real-time responsiveness and stable head tracking for embedded in-car systems [13].

Lu Yu et al. incorporated head movement analysis into a multi-modal driver monitoring system by using solvePnP to estimate head pose from 2D facial landmarks. The method tracks pitch and roll angles to monitor head orientation changes in real time, identifying signs of drowsiness through nodding or tilting behaviors. The system was evaluated using highway driving data collected from 30 drivers and achieved a fatigue classification accuracy of 97.78%, with 98.1% for alert, 98.8% for mild fatigue, and 96.5% for severe fatigue [14].

Ye et al. developed a fatigue detection system combining eye and mouth state classification with head pose estimation using the EPnP algorithm. Euler angles (pitch, yaw, roll) were calculated from facial landmarks to monitor head movement. Fatigue was detected when the angles exceeded 21° for pitch, 20° for yaw, or 20.5° for roll. The method showed strong real-time performance and effectively identified abnormal head postures linked to fatigue [15].

Lastly, a study utilizing the Pose-Extended Active Shape Model (PE-ASM) defined angle thresholds to recognize extreme head orientations. The model switches between pose templates based on preset thresholds of ±15° to ±39° for pitch, ±15° to ±75° for yaw, and ±15° to ±52° for roll, improving accuracy in cases of non-frontal head positioning and enhancing the detection of distracted or drowsy states [16].

2.2 Summary for each Article

Table 2.1: Summary of Methods, Performance, and Evaluation Metrics for Drowsiness Detection Systems

Paper	Method	Evaluation	Strengths	Weaknesses
[1]	MobileNetV2 Transfer learning Dense layers	Accuracy: 99.97% (train) 99.86% (test)	Extremely high accuracy Real-time ready Robust in varied lighting	Large dataset required Training can be resource-intensive
[2]	MobileNetV2 Haar Cascade MRL Eye Dataset	Accuracy: 99.33%	Live video analysis Includes emergency response actions Works well with simulation	Relies on eye detection only Less robust in occluded conditions
[3]	MobileNetV2 vs EfficientNetB0 vs NasNet	Accuracy: 100% (train) 95% (val 94% (test)	Lightweight(14MB Suitable for embedded system Fast training convergence	Slight overfitting observed Lower accuracy than EfficientNet
[4]	EAR & MAR using Dlib OpenCV	Accuracy: 95% (day) 85% (night)	Works in varied lighting Real-time detection Handles spectacles	Reduced accuracy in poor lighting Dependent on camera placement
[5]	EAR & MAR with Raspberry Pi + NoIR cam SMS alert integration	Accuracy: 100% (eye) 79% (yawn) Precision: 82% (eye) 100%(yawn)	Multi-level alerts (buzzer, LED, SMS) High accuracy under various conditions	Occasional false positives Yawning less accurate under wide mouth movement
[6]	EAR & MAR	Real-time functional prototype;	Simple and low-cost	No quantitative accuracy data

	with Raspberry Pi + IR webcam OpenCV	qualitative success	Effective under controlled settings	Limited testing documentation
[7]	MobileNet Facial Alignment Network (FAN)	Descriptive analysis only (no specific accuracy)	Detects fatigue & distraction Webcam compatible Real-time application ready	No numeric evaluation Methodology described qualitatively
[8]	Eye blink detection using horizontal symmetry via webcam Single-frame symmetry metric	Accuracy: 94.8% FPR: 1% (JZU dataset)	Real-time detection at 110 fps Non-intrusive Low-cost hardware	Sensitive to illumination and glasses Symmetry-based detection may miss subtle blinks
[9]	IR-based eye blink sensor + Arduino with buzzer and seat vibration alerts	Accuracy: 80% Sensitivity: 77.27% Specificity: 83.33%	Dual alert system (sound + vibration) Real-time and hardware-implementable Effective at night	Accuracy depends on fixed sensor position Performance drops in high-light environments
[10]	Eye blink + yawning detection using Adaboost, RHOG, SVM in MATLAB	Accuracy: 85.71% Sensitivity: 75 Specificity: 88.24%	Hybrid detection (blink + yawn) Robust feature extraction Logical threshold-based alert	MATLAB-based; not optimized for embedded systems Requires stable frontal video input
[11]	solvePnP + EAR correction for pose compensation	EAR improved: 89.85% → 94.94%	Reduces error caused by head rotation Improved robustness in eye closure detection	Focuses mainly on EAR correction No full fatigue detection evaluation

[12]	EAR, yawning, head-dow solvePnP + fuzzy logic	Fatigue detection accuracy based on thresholds; qualitative classification only	Integrates multiple indicators Uses clearly defined time/frequency thresholds	No angle-based thresholds More logic-based than deep learning
[13]	Dlib landmarks + solvePn Alerts on head tilt	Qualitative real-time success	Simple implementation Works in live video streams	No defined angle thresholds No detailed accuracy evaluation
[14]	Head pose via solvePnP Multi-modal detection	Fatigue classification accuracy: 97.78% Alert: 98.1% Mild: 98.8% Severe: 96.5%	Based on actual human head movement ranges Can be integrated into other systems	Not tested with specific detection thresholds No evaluation on alert precision
[15]	Face feature extraction Head pose via EPnP Triggers on angle thresholds	High detection performance using pose angle thresholds; exact accuracy not stated	Clearly defined threshold values Combines image enhancement and pose tracking	Relies on visual features only Requires system calibration
[16]	PE-ASM + POSIT HMM for fatigue behavior	Accuracy: 98% (train) 92% (test)	Handles extreme head poses Models time-based behavior with HMM	Requires 3D model matching Training and setup more complex

2.3 Limitation of Previous Studies

A significant limitation across various studies is their **sensitivity to lighting conditions**. Detection accuracy frequently decreased in low-light, night-time, or excessively bright environments, particularly when standard webcams or infrared sensors were employed without sufficient lighting support. This issue was reported in multiple studies [1], [2], [4], [10], indicating that variations in illumination significantly impact the practical reliability of fatigue detection systems.

Another shared issue involves the **requirement for frontal facial alignment and consistent camera positioning**. Studies [6], [8], [13], and [15] highlighted that angled or non-frontal views resulted in inaccurate facial landmark tracking and pose estimation, adversely affecting detection accuracy. This dependency limits the robustness of systems when faced with natural driver movements and varying seating postures.

Additionally, systems often struggled with **occlusions or facial obstructions**, such as masks, spectacles, or hair interfering with accurate landmark detection. This limitation was explicitly noted in studies [11] and [14], where occluded facial landmarks led to inconsistent or failed detections, reducing reliability in real driving environments.

Several reviewed studies also suffered from limited **real-world validation**. Studies [3], [7], [12], and [16] primarily relied on controlled laboratory or simulation tests. Consequently, their performance under realistic driving scenarios involving prolonged use, vibrations, and varying environmental factors remains uncertain, raising concerns about generalizability and practical deployment.

Lastly, certain hardware-based approaches encountered **performance limitations and restricted scalability**. For example, study [10] reported reduced accuracy due to sensitivity issues with IR-based eye blink sensors. Similarly, study [5] noted limitations related to limited test subjects and a lack of long-term evaluations, suggesting the need for broader and extended field testing.

2.3 Proposed Solutions

This project aims to overcome the limitations identified in previous studies by introducing several targeted improvements and innovations. Specifically, the project proposes the following solutions:

1. **Enhanced Lighting Adaptability:**
   Implement adaptive preprocessing techniques, including brightness normalization and image enhancement methods, to improve fatigue detection accuracy in varying lighting conditions such as low light or high glare scenarios.
2. **Robustness to Occlusions and Facial Obstructions:**
   Integrate advanced facial landmark detection methods and robust convolutional neural network architectures like MobileNetV2 to ensure accurate tracking despite the presence of spectacles or other facial obstructions.
3. **Reliable Head Pose Detection:**
   Incorporate solvePnP based head pose estimation with clearly defined angle thresholds (pitch ±21°, yaw ±20°, roll ±20.5°) to accurately detect abnormal head movements, thus improving fatigue monitoring under realistic driver movements and varied camera placements.
4. **Multi Feature Fatigue Detection:**
   Combine multiple fatigue indicators including EAR, MAR, blink frequency, and head pose into an integrated detection model to enhance reliability and reduce false alarms compared to systems that rely on single feature detection.
5. **Real world Testing and Validation:**
   Conduct comprehensive evaluations under realistic driving conditions, incorporating prolonged testing periods, varied environmental settings, and diverse driver behaviors to ensure the developed system is robust, practical, and reliable for everyday use.

Chapter 3

System Methodology/Approach OR System Model

3.1 System Design Diagram/Equation

This section presents the overall design of the proposed fatigue detection system. The design illustrates the sequential flow from input acquisition to output generation, showing how the camera, pre-processing, feature extraction, classification, decision logic, and alert modules interact. The diagram provides a overview to guide the detailed explanation of the architecture in the following subsection.

In addition to the system flow, the design also incorporates mathematical equations used for fatigue detection:

- The **EAR** is calculated to determine the openness of the eyes:

$$EAR = \frac{||p2 - p6|| + ||p3 - p5||}{2 \times ||p1 - p4||} \qquad (1)$$

  Where p_1 to p_6 are the 6 facial landmark points around the eye.

  Usage: When EAR drops below a threshold (typically **0.25**), the system detects that the eyes are closing or blinking.

- The **MAR** is calculated to detect yawning:

$$MAR = \frac{||p3 - p11|| + ||p5 - p9||}{2 \times ||p1 - p7||} \qquad (2)$$

  Where p_1 to p_{11} represent the key landmarks around the mouth region.

  Usage: A MAR value higher than **0.6–0.9** typically indicates yawning.

- **Head Pose Angles (Pitch, Roll, Yaw):**

  Head orientation is estimated using the solvePnP algorithm with 3D-2D landmark correspondences. Distraction and head tilt are determined when the following thresholds are exceeded:

  - Pitch > 21° → indicates downward or upward head tilt.
  - Roll > 20.5° → indicates sideways head tilt.
  - Yaw > 20° → indicates lateral distraction (looking away from the road).

3.1.1 System Architecture Diagram

The proposed system architecture was designed to run efficiently on the Raspberry Pi 4 Model B. It consists of five main layers: input, pre-processing, feature extraction and classification, decision logic, and output, as illustrated in Figure 3.1.

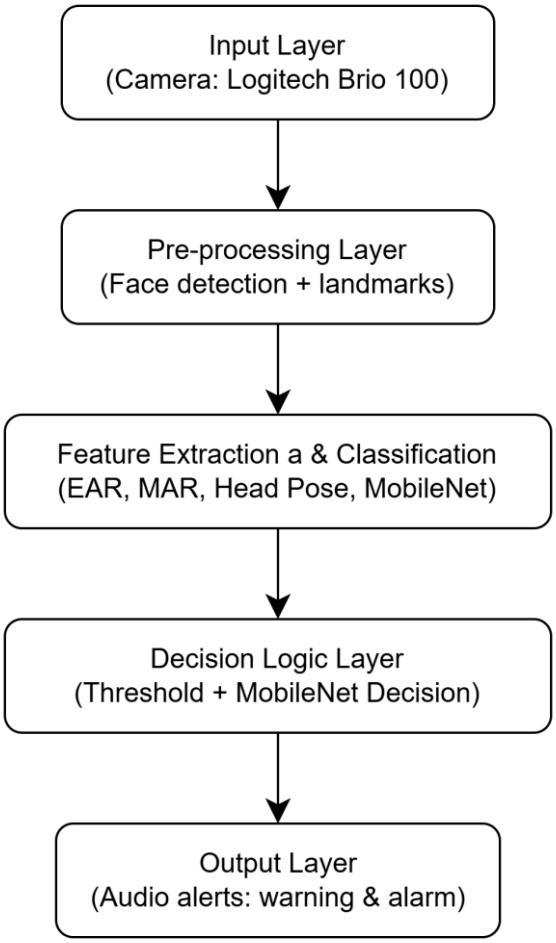


Figure 3.1: System Architecture Flowchart

Overview of System Architecture:

a. **Input Layer**

A Logitech Brio 100 webcam continuously captured the driver's face at 640 × 480 resolution, providing real-time visual data for processing.

b. **Pre-processing Layer**

Face detection was carried out using Dlib frontal face detector. Dlib 68 landmark predictor then identified key points around the eyes, mouth, and nose. Cropped face regions were resized and normalised for MobileNet classification.

c. **Feature Extraction and Classification Layer**

Three main features were derived: EAR for eye closure, MAR for yawning, and head pose angles (pitch, yaw, roll) using the solvePnP algorithm. Alongside this, a TensorFlow Lite MobileNet model classified face states into categories such as open/closed eyes and yawn/no yawn, complementing the traditional feature-based approach.

d. **Decision Logic Layer**

The decision logic combined EAR, MAR, head pose, and MobileNet outputs to evaluate fatigue. EAR and MAR thresholds detected eye closure and yawning, while head pose angles identified tilting or distraction. MobileNet provided categorical predictions (eye open, eye closed, yawn, no yawn), which were cross-checked with the traditional features to confirm events. Based on these integrated results, the system determined the fatigue stage (mild, moderate, or severe) and triggered the corresponding alerts.

e. **Output Layer**

According to the fatigue stage, the system generated alerts through warning sounds, alarm tones, and text-to-speech voice messages. Alerts were managed sequentially to ensure clarity and prevent overlap.

This modular design allowed the system to operate robustly in real driving conditions, handling challenges such as low light, spectacles, and face obstruction.

3.1.2 Use Case Diagram and Description

Figure 3.2 presents the use case diagram of the proposed fatigue detection system. The main actor is the Driver, while the system boundary is the Fatigue Detection System running on Raspberry Pi. The diagram shows the functions available to the actor without implying execution order. The use cases cover image capture, landmark extraction, state analysis (eyes and mouth), head-pose estimation, integrated evaluation, and alerts.

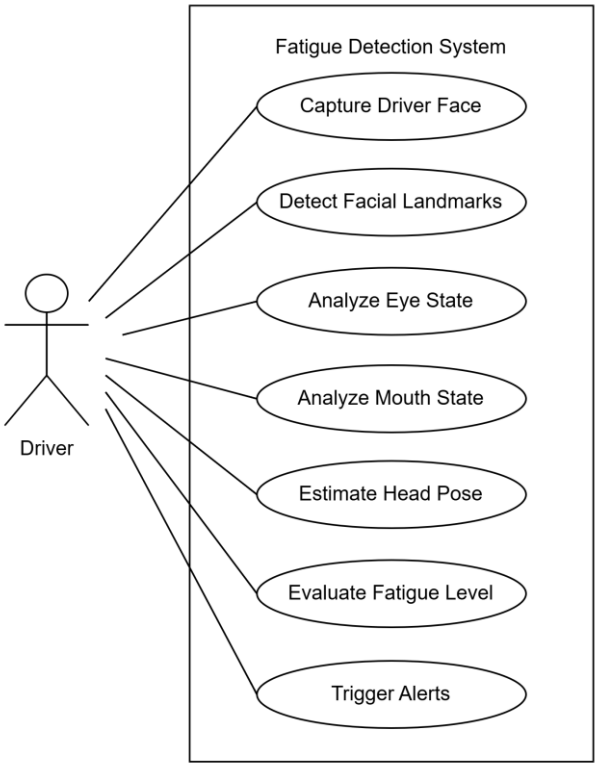


Figure 3.2: Use Case Diagram of the Fatigue Detection System

Actor

- **Driver**: the person whose face is monitored while driving.

System Boundary

- **Fatigue Detection System**: real-time embedded application on Raspberry Pi 4 Model B with camera input and audio output.

Use Cases and Descriptions

Table 3.1: Use Cases and Descriptions

Use Case	Brief Description	Primary Outcome
Capture Driver Face	Acquire live frames from the Logitech Brio 100 camera at 640×480 for real-time processing.	Captured frames are sent to the next processing steps.
Detect Facial Landmarks	Detect face region and extract 68 landmarks (eyes, mouth, nose) using Dlib.	Landmark coordinates for feature computation.
Analyse Eye State	Compute EAR and use MobileNet inference to confirm open/closed eyes.	Eye closure is detected and recorded
Analyse Mouth State	Compute MAR and use MobileNet inference to confirm yawn/no-yawn.	Yawning is detected and recorded
Estimate Head Pose	Apply solvePnP on selected landmarks to derive pitch, yaw, and roll.	Head-tilt/distraction indicators available.
Evaluate Fatigue Level	Combine EAR, MAR, head pose, and MobileNet results with set limits to decide the fatigue level (mild, moderate, or severe).	Current fatigue stage determined.
Trigger Alerts	Issue stage-appropriate warnings (voice + sounds) via the audio system.	Driver receives timely notifications.

3.1.3 Activity Diagram

The activity diagram in Figure 3.3 illustrates the workflow of the fatigue detection system, from camera initialisation to alert generation. The process can be explained in the following steps:

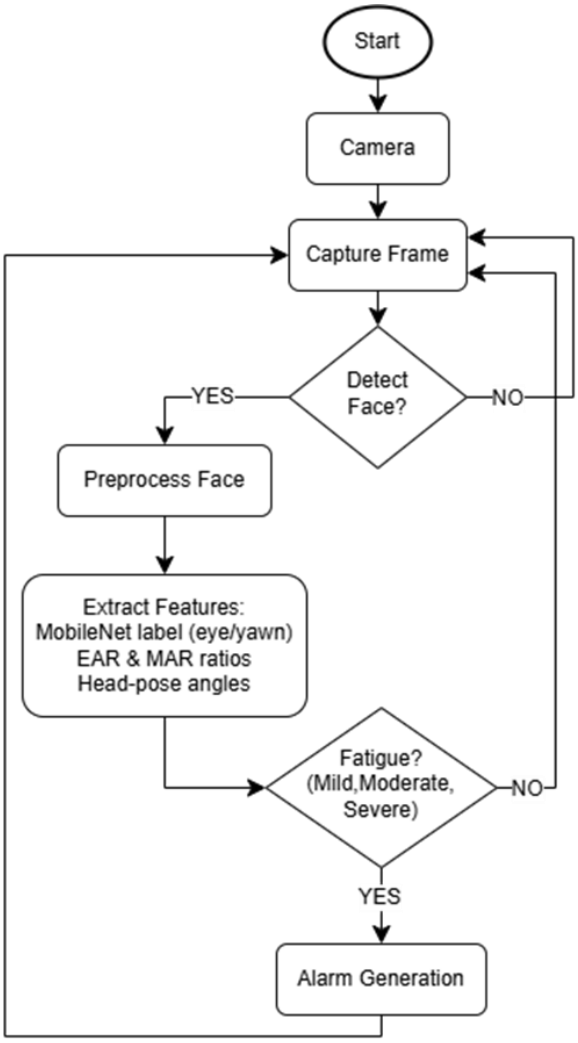


Figure 3.3: Activity Diagram for Fatigue Detection System

1. **System Initialisation** – The Raspberry Pi activates the camera, loads the MobileNet TFLite model, and sets up the thresholds for EAR, MAR, and head pose estimation.
2. **Frame Capture and Face Detection** – Frames are continuously captured from the camera. If a face is not detected, the system loops back to capture the next frame. If detected, facial landmarks are extracted.
3. **Feature Extraction and Classification** – EAR and MAR are calculated from the facial landmarks, while head pose is estimated using solvePnP. At the same time, MobileNet classifies the face state (eye open/closed, yawn/no yawn).

4. **Decision Logic and Fatigue Evaluation** – The system evaluates fatigue based on blink frequency, yawning events, and head tilt occurrences over specific time windows. These results are fused with MobileNet predictions and compared against thresholds to determine the fatigue stage.
5. **Alert Generation** – Based on the fatigue level, the system triggers an appropriate response:
    - Mild fatigue: voice message only
    - Moderate fatigue: voice message + alarm
    - Severe fatigue: extended alarm sound

    All alerts are executed on separate threads to maintain real-time processing without interrupting video analysis.

To ensure consistent classification, fatigue was divided into three stages according to the detection criteria summarised in Table 3.2.

Table 3.2: Fatigue Stage Classification

Stage	**Detection Criteria**
Mild	Blink ≥ 15 in 1 minute Yawn ≥ 3 in 1 minute Head tilt ≥ 3 times in 1 minute
Moderate	Blink ≥ 15 in 40 seconds Yawn ≥ 3 in 40 seconds Head tilt ≥ 3 times in 40 seconds
Severe	Eyes closed ≥ 3 seconds Blink ≥ 15 in 25 seconds Yawn ≥ 3 in 25 seconds Head tilt ≥ 3 in 25 seconds or tilt duration > 2 seconds

This structured classification enables the system to respond appropriately to escalating signs of drowsiness, ensuring timely and progressive alerts to the driver.

Chapter 4

System Design

4.1 System Block Diagram

The system block diagram of the fatigue detection system is illustrated in Figure 4.1. The design follows a modular structure that begins with the input stage, proceeds through several processing blocks, and ends with the output stage that generates alerts. All modules are executed on the Raspberry Pi 4 Model B, which serves as the main processing unit. This top-down design ensures that the system can be rebuilt and understood step by step.

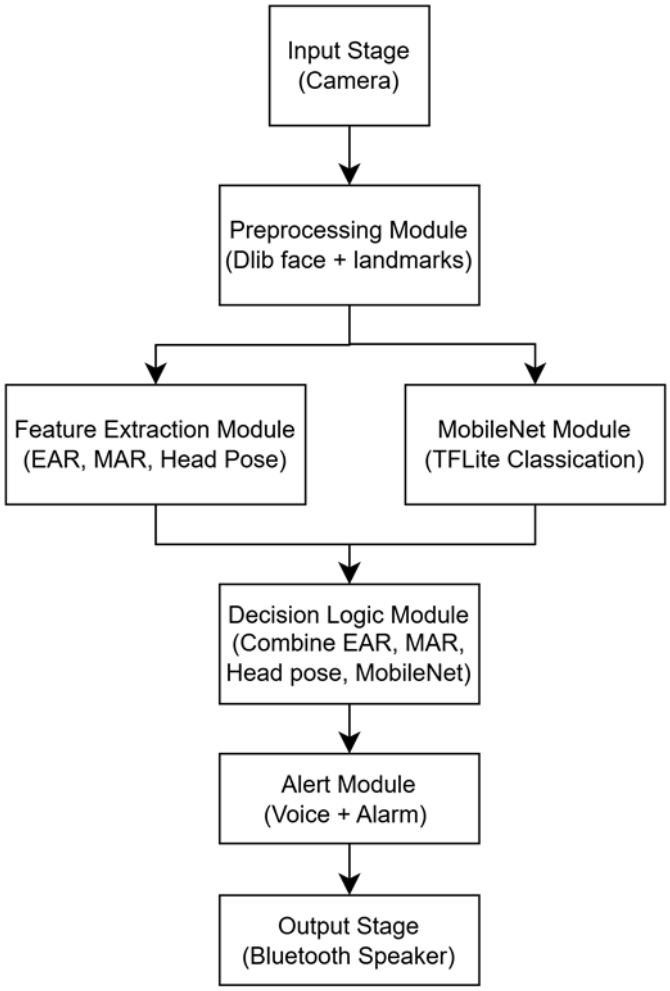


Figure 4.1: System Block Diagram for Fatigue Detection System

The **input stage** consists of the Logitech Brio 100 camera, which continuously captures real-time video of the driver's face at a resolution of 640 × 480 pixels. The camera is directly connected to the Raspberry Pi 4 Model B, which functions as the main processing unit.

The **processing stage** of the Raspberry Pi is divided into several modules:

- **Pre-processing Module**

  This block receives the captured frames and applies Dlib's frontal face detector to locate the face region. Dlib's 68-point landmark predictor is then used to identify key points around the eyes, mouth, and nose. These landmarks serve as the basis for further analysis.

- **Feature Extraction Module**

  From the detected landmarks, the EAR is calculated to determine eye closure, while the MAR is computed to detect yawning. In addition, head pose estimation is performed using the solvePnP algorithm, which calculates pitch, yaw, and roll angles to monitor head tilting and distraction events.

- **Deep Learning Module (MobileNet TFLite)**

  A lightweight MobileNet model, converted into TensorFlow Lite format, is integrated to classify the facial state into categories such as open eyes, closed eyes, yawn, and no yawn. The output of this module complements the EAR and MAR values, providing more robust detection in real-time.

- **Decision Logic Module**

  This module fuses the outputs of EAR, MAR, head pose, and MobileNet predictions. Thresholds and smoothing functions are applied to reduce false positives. Based on the evaluation, the driver's condition is categorised into three stages: mild, moderate, or severe fatigue.

- **Alert Module**

  The final block generates alerts according to the fatigue stage. For mild fatigue, a voice message is played. For moderate fatigue, both a voice message and an alarm sound are triggered. For severe fatigue, a continuous alarm is activated. To maintain real-time performance, alerts are executed on separate threads, ensuring the detection pipeline continues without interruption.

The **output stage** consists of the Raspberry Pi's audio interface connected to a Bluetooth speaker. This delivers clear voice and sound alerts to the driver.

This modular design ensures that every component, from camera capture to alert generation, is well-defined. With the block diagram and detailed explanation of each module, the system can be reconstructed and implemented by following the design.

4.2 System Component Specification

This section describes the hardware and software components used in the development of the fatigue detection system. Each component was selected based on performance, compatibility, and suitability for real-time processing on an embedded platform.

4.2.1 Hardware Components

The hardware configuration of the fatigue detection system centres on the Raspberry Pi 4 Model B as the main processing unit. A Logitech Brio 100 webcam serves as the input device, continuously capturing the driver's face in real time for fatigue analysis. The system output is delivered through a Bluetooth speaker, which provides voice and alarm alerts to the driver. For portable and stable power supply, an 18650-battery shield with dual 18650 lithium-ion batteries are used, allowing the Raspberry Pi to operate independently of a direct power adapter. This setup ensures the system is compact, reliable, and suitable for in-vehicle deployment.

Table 4.1: Specifications of Raspberry Pi

Description	Specifications
Model	Raspberry Pi 4 Model B
Processor	Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit Soc @ 1.8GHz
Memory	4GB LPDDR4 RAM
Wireless Connectivity	2.4 and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0 BLE
Graphic	OpenGL ES 3.1, Vulkan 1.0
Storage Needed	Minimum 16GB ROM of SD card is required

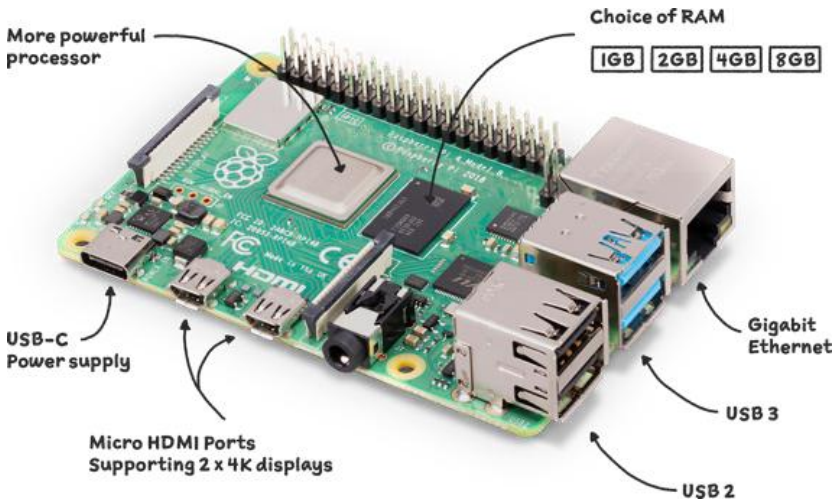


Figure 4.2: Raspberry Pi 4 Model B

Table 4.2: Specifications of Camera

Description	Specifications
Model	Logitech BRIO 100
Resolutions	1080p/30fps (1920x1080 pixels),
Camera Megapixel	2MP
Focus Type	Fixed Focus
Lens Type	Custom 4-element plastic lens with anti-reflective coating
RAM Requirement	2GB RAM or more for 1080p video streaming
USB Connectivity	USB-A plug-and-play
Auto-light Balance	RightLight 2 for challenging lighting conditions

Figure 4.3: Logitech BRIO 100 Camera

Table 4.3: Specifications of 18650 Battery Shield with Dual 18650 Lithium-ion Batteries

Description	Specifications
Output Voltage	5V (rated), up to 3A (max)
Rated Output Current	2.2 A
Charging Current	600-800 mA (Micro-USB)
Supported Baterries	2 x 18650 Li-ion cells (3.7 Li-ion, 12000mAh)
Battery Protection	Overcharge / Over-discharge
Input Voltage	5 – 8 V DC

Figure 4.4: 18650 Battery Shield with Dual 18650 Lithium-ion Batteries

Table 4.4: Specifications of MOXOM SK-39 Bluetooth Speaker

Description	Specifications
Bluetooth Version	5.0/5/1
Range	Up to 12m
Output Power	5W
Battery Capacity	1200mAh
Playtime	5 – 7 hours
Charging Input	5V/1A (USB Type-C)

Figure 4.5: MOXOM SK-39 Bluetooth Speaker

4.2.2 Software Components

The software development for the fatigue detection system is implemented in **Python**, with several supporting libraries to enable real-time image processing, fatigue detection, and audio alert functionality on the Raspberry Pi.

- **Python 3.8+**: Main programming language used for system development and deployment.
- **OpenCV**: Provides real-time image acquisition, frame handling, grayscale conversion, and display functionality.
- **TensorFlow Lite (tflite-runtime)**: Executes the MobileNet model on the Raspberry Pi with optimised inference for classifying eye and mouth states.
- **Dlib**: Performs frontal face detection and extracts 68 facial landmarks for EAR, MAR, and head pose estimation.
- **NumPy**: Handles matrix operations and numerical calculations, including preprocessing of face images.
- **SciPy (spatial.distance)**: Used to compute Euclidean distances required for EAR and MAR calculations.
- **pyttsx3**: Implements the text-to-speech engine for generating clear and non-overlapping voice alerts.
- **Subprocess (paplay/aplay)**: Plays warning and alarm sound files, ensuring compatibility with Bluetooth or wired speakers.
- **Threading & Queue**: Enable concurrent processing of audio alerts (voice + sound) without blocking the real-time video pipeline.
- **Collections (deque)**: Provides efficient rolling buffers for smoothing EAR, MAR, and head pose values.

4.3 Circuit and Components Design

The circuit and component design of the fatigue detection system integrates the Raspberry Pi 4 Model B with input, power, and output devices to form a complete monitoring unit. Figure 4.6 illustrates the connection of all hardware components.

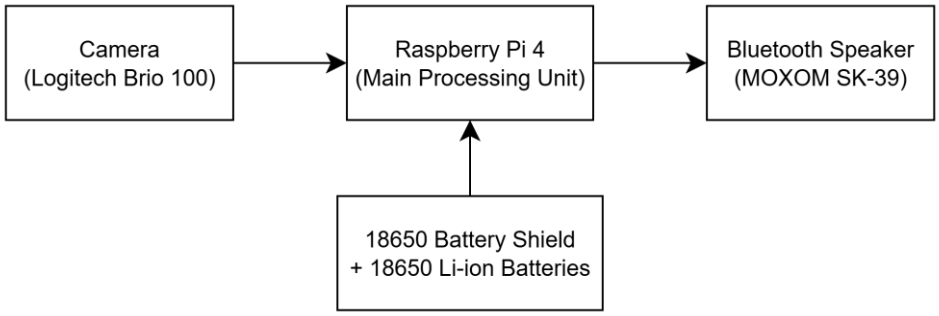


Figure 4.6: Circuit and Component Design of the Fatigue Detection System.

The **Logitech Brio 100 camera** acts as the input stage, capturing the driver's face in real time and transferring the video stream to the **Raspberry Pi 4** via a USB connection. The Raspberry Pi serves as the main processing unit, running the fatigue detection algorithms that include pre-processing, feature extraction, MobileNet classification, decision logic, and alert generation.

Power is supplied through an **18650-battery shield with dual 18650 lithium-ion batteries**, connected to the Raspberry Pi using a Micro-USB/Type-C interface. This configuration ensures a stable 5V power supply, with built-in protection against overcharging and discharging, making the system portable and suitable for in-vehicle deployment.

The **output stage** is implemented using a **MOXOM SK-39 Bluetooth speaker**, which pairs wirelessly with the Raspberry Pi. The speaker delivers voice messages and alarm sounds to alert the driver according to the detected fatigue stage.

The overall circuit flow can be summarised as:

- **Input:** Camera → Raspberry Pi
- **Processing:** Fatigue detection modules on Raspberry Pi
- **Power:** Battery shield → Raspberry Pi
- **Output:** Raspberry Pi → Bluetooth speaker

This simple yet efficient design ensures reliable real-time operation while keeping the system compact and portable for in-vehicle use.

4.4 System Components Interaction Operations

The fatigue detection system operates through continuous interaction between hardware and software components, ensuring real-time monitoring and timely alerts. The overall interaction process can be described in the following sequence:

1. **Image Capture**
    - The Logitech Brio 100 camera continuously records the driver's face in real time.
    - Captured frames are transmitted via USB to the Raspberry Pi for processing.
2. **Pre-processing and Landmark Detection**
    - The Raspberry Pi uses the Dlib library to detect the face region and extract 68 facial landmarks.
    - Key regions of interest, such as the eyes and mouth, are identified for further analysis.
3. **Feature Extraction and Classification**
    - EAR and MAR are computed from the extracted landmarks to track eye closure and yawning events.
    - Head pose estimation using solvePnP calculates pitch, yaw, and roll to detect head tilts or distraction.
    - In parallel, the MobileNet TFLite model classifies facial states (open/closed eyes, yawn/no yawn) for robust detection.
4. **Decision Logic**
    - The results from EAR, MAR, head pose, and MobileNet classification are fused.
    - Thresholds and smoothing filters are applied to reduce false positives.
    - The system determines the fatigue stage: mild, moderate, or severe.
5. **Alert Generation**
    - Based on the fatigue level, the system triggers alerts through the MOXOM SK-39 Bluetooth speaker:
        - Mild fatigue → voice message only
        - Moderate fatigue → voice message + alarm
        - Severe fatigue → extended alarm sound
    - Alerts are executed on separate threads to maintain uninterrupted video analysis.

6. **Power Management**
    - The 18650-battery shield supplies stable 5V power to the Raspberry Pi, enabling portable operation within the vehicle.
    - Built-in overcharge/discharge protection ensures safe and reliable operation.

Through this interaction, the system achieves real-time detection and response, with each component working in coordination to provide a practical and reliable fatigue detection solution for drivers.

Chapter 5

System Implementation

5.1 Hardware Setup

The hardware setup of the fatigue detection system is shown in Figure 5.1. The system is installed inside a vehicle to evaluate its real-time performance under driving conditions. The setup consists of four main components: camera, Raspberry Pi 4, battery shield with 18650 lithium-ion batteries, and Bluetooth speaker.

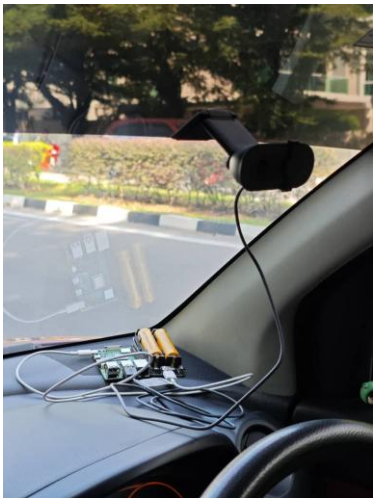


Figure 5.1 Hardware setup of the fatigue detection system inside the vehicle.

The **Logitech Brio 100 camera** is mounted on the windshield facing the driver. Its position ensures that the driver's face is clearly captured during both day and night driving conditions. The camera is connected to the **Raspberry Pi 4 Model B** via a USB port, which serves as the main processing unit for running the fatigue detection algorithms.

Power is supplied by an **18650-battery shield** equipped with two rechargeable 18650 lithium-ion batteries. This shield connects to the Raspberry Pi through a Micro-USB/Type-C interface, providing a stable 5V power output with built-in protection. This configuration makes the system portable and independent of the car's main electrical system.

The **MOXOM SK-39 Bluetooth speaker** is placed beside the driver and paired wirelessly with the Raspberry Pi. It delivers voice messages and alarm sounds, ensuring the driver receives clear and immediate notifications in case of fatigue detection.

The arrangement of these components allows the system to operate effectively inside the car. The camera placement guarantees accurate facial monitoring, while the Raspberry Pi and battery shield remain securely placed on the dashboard for stable operation, and the speaker provides alerts directly near the driver.

5.2 Software Setup

The software setup of the fatigue detection system was developed in **Python 3.8+** and deployed on the Raspberry Pi 4 Model B running Raspberry Pi OS. Several libraries and frameworks were installed to support real-time image processing, fatigue detection, and audio alerts.

The setup process is summarised below:

1. **Programming Environment**
    - Python 3.8+ was used as the primary programming language.
    - A virtual environment was created to manage dependencies and ensure compatibility of libraries.
2. **Library Installation**

    The following Python libraries were installed using pip:
    - **OpenCV** – real-time image capture and frame processing.
    - **Dlib** – facial detection and 68-point landmark extraction.
    - **NumPy** – matrix and numerical operations.
    - **SciPy** – Euclidean distance calculations for EAR and MAR.
    - **TensorFlow Lite (tflite-runtime)** – lightweight inference of the MobileNet model on Raspberry Pi.
    - **pyttsx3** – text-to-speech engine for generating voice alerts.
    - **subprocess** – plays sound files via paplay or aplay for alarm alerts.
    - **threading & queue** – enable alerts to run in parallel without interrupting video analysis.
    - **collections (deque)** – stores recent EAR, MAR, and head pose values for smoothing.
3. **Model Deployment**
    - A MobileNet model was trained and converted to TensorFlow Lite format.
    - The .tflite file was stored in the saved_model/ directory and loaded at runtime for classification of eye and mouth states.

4. **Script Execution**
    - The system runs through the main program fatigue_detection_pi.py.
    - A **systemd service** was configured to auto-start this program whenever the Raspberry Pi is powered on.
    - To make the system flexible for different hardware setups, **environment variables** were used:
        - **HEADLESS** → set to 1 when running without a monitor (no GUI display), or 0 when testing with a monitor (OpenCV window visible).
        - **CAM_DEVICE** → specifies the camera device (e.g., /dev/video0 for the default USB camera).
        - **AUDIO_DEVICE** → defines the audio sink for playback, ensuring alerts are routed to the correct output (e.g., Bluetooth speaker).
5. **Audio Setup**
    - The Raspberry Pi was paired with the **MOXOM SK-39 Bluetooth speaker**.
    - The default audio sink was configured so that both pre-recorded alarm sounds and text-to-speech voice alerts were output directly through the speaker.

This setup ensures the fatigue detection system operates in real time with efficient integration of image capture, facial analysis, deep learning inference, and multi-threaded alert handling on the Raspberry Pi.

5.3 Setting and Configuration

The fatigue detection system required several settings and configurations to ensure reliable performance on the Raspberry Pi platform. These included camera adjustments, software configuration, and audio setup for real-time alerts.

1. **Camera Configuration**
    - The Logitech Brio 100 was configured to capture frames at **640 × 480 resolution** to balance processing speed and accuracy.
    - The camera was positioned on the windshield, directly facing the driver, to maintain a clear and consistent view of the face.
    - Frame capture was stabilised at **30 FPS**, ensuring smooth real-time detection.
2. **Software Configuration**
    - The main program (fatigue_detection_pi.py) was set to run automatically using a **systemd service**, enabling the system to start immediately when the Raspberry Pi is powered on.
    - Environment variables were applied to provide flexible system behaviour:
        - HEADLESS=1 → enables headless mode (no GUI display) for in-car operation.
        - CAM_DEVICE=/dev/video0 → ensures the correct camera is used.
        - AUDIO_DEVICE → directs alerts to the MOXOM SK-39 Bluetooth speaker.
    - The MobileNet TFLite model was placed in the saved_model/ directory, while supporting files such as facial landmark predictors and alarm sound files were stored in the project directory.
3. **Audio Configuration**
    - The Raspberry Pi was paired with the MOXOM SK-39 Bluetooth speaker, and the default audio sink was updated to route all alerts through Bluetooth.
    - Both pre-recorded alarm sounds and text-to-speech voice messages were configured to play without overlapping, using threading and a shared audio lock.
4. **Power Configuration**
    - An **18650-battery shield with dual lithium-ion cells** was connected to supply a stable 5V output to the Raspberry Pi.

- The battery shield provided overcharge and discharge protection, ensuring safe and continuous operation during testing.

Through these configurations, the system was able to operate seamlessly in a vehicle environment, delivering real-time fatigue detection and timely alerts without requiring manual intervention after startup.

5.4 System Operation

The fatigue detection system runs continuously to capture, analyse, and evaluate the driver's condition in real time. When powered on, the Raspberry Pi launches the fatigue_detection_pi.py program automatically through a systemd service. The system operates as follows:

1. **Face Detection and Landmark Extraction**
   - The camera captures video frames of the driver's face.
   - Dlib detects the face and extracts 68 landmarks to localise the eyes and mouth.
2. **Feature Extraction and Classification**
   - EAR is calculated to monitor eye closure.
   - MAR is computed to detect yawning.
   - Head pose is estimated with solvePnP to identify pitch, roll, and yaw movements.
   - The MobileNet TFLite model classifies the face state (open/closed eyes, yawn/no yawn).
3. **Decision Logic**
   - Extracted features and MobileNet results are combined.
   - If thresholds are exceeded (e.g., pitch > 21°, roll > 20.5°,yaw > 20°, EAR < 0.25, MAR > 0.6), the system categorises fatigue into three levels: mild, moderate, or severe.
4. **Alert Generation**
   - The MOXOM SK-39 Bluetooth speaker issues alerts depending on the fatigue stage:
     - Mild → voice message only
     - Moderate → voice message + warning sound
     - Severe → extended alarm sound

5. **Graphical User Interface (GUI) for Testing**
    - During development and testing on a laptop, the system was run with **HEADLESS=0**, enabling a GUI window to display the driver's face, EAR, MAR, blink frequency, yawns, head pose angles, and fatigue stage.
    - On the Raspberry Pi deployment, the system runs in **HEADLESS=1** mode (headless), without GUI display, while maintaining the same detection and alert functions.

5.4.1 Screenshot Demonstration

For demonstration, Figure 5.2 shows the fatigue detection system in operation with GUI enabled on a laptop:

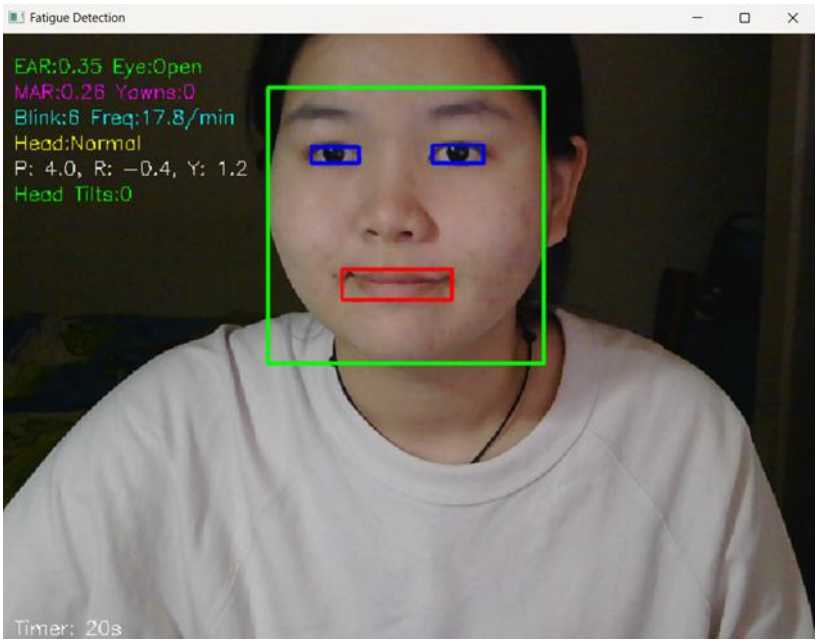


Figure 5.2(a): Normal condition – eyes open, no yawning, stable head pose.

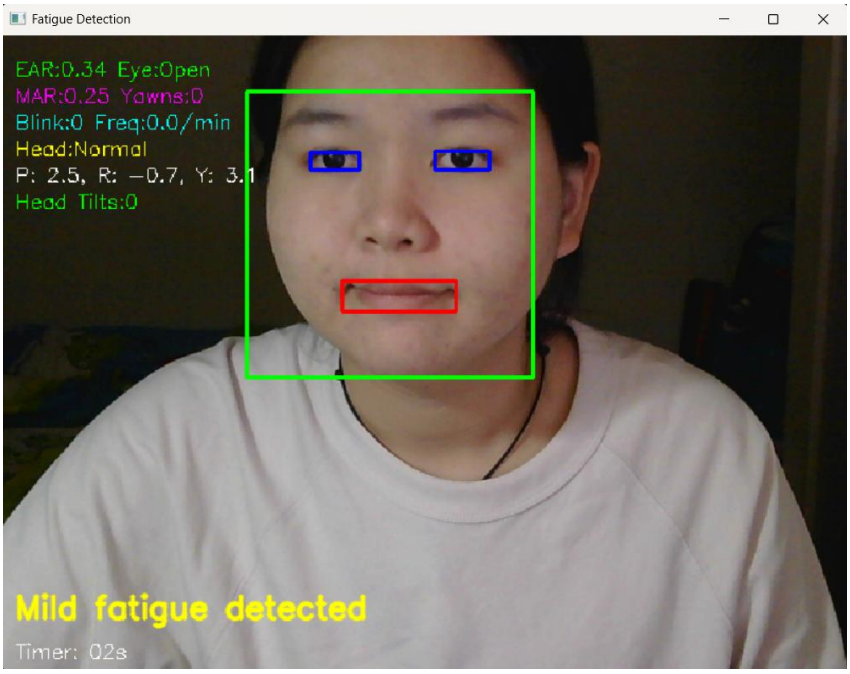


Figure 5.2(b): Mild fatigue detected – system triggers a voice advisory message.

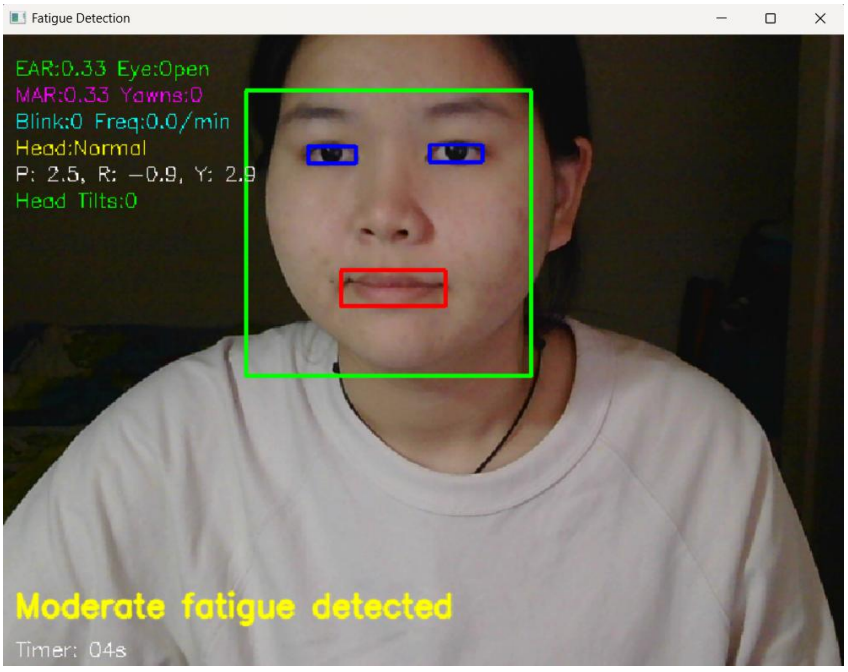


Figure 5.2(c): Moderate fatigue detected – system triggers voice + warning alarm.

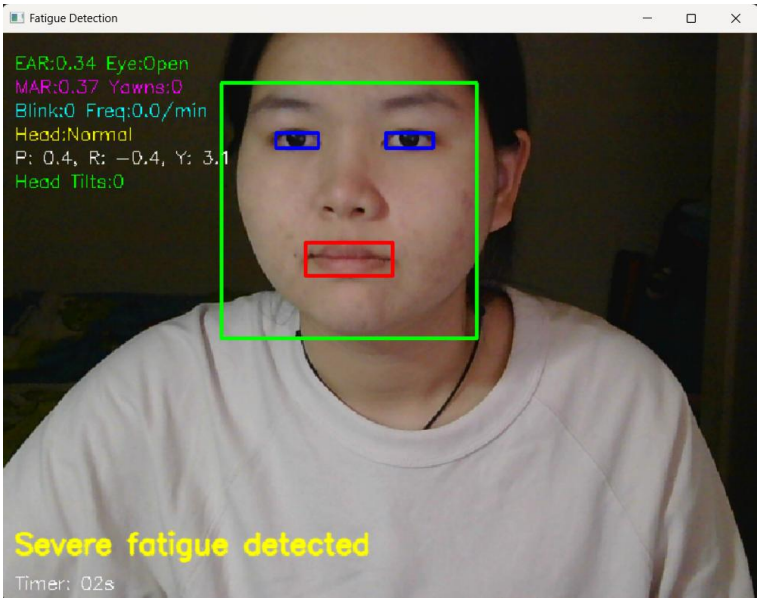


Figure 5.2(d): Severe fatigue detected – system triggers extended alarm sound.

5.5 Implementation Issues and Challenges

During the development of the fatigue detection system, several issues and challenges were encountered when transitioning from the laptop environment to the Raspberry Pi platform. These challenges mainly arose from hardware limitations, library compatibility, and the need for performance optimisation. The main issues are summarised below:

1. **Code Portability from Laptop to Raspberry Pi**

   The system could not be directly copied from the laptop to the Raspberry Pi. Several changes had to be made to ensure compatibility:

   - **Audio format support** → .mp3 files that worked on the laptop could not be played reliably on the Raspberry Pi. This was resolved by converting audio alerts to .wav format and using paplay or aplay for playback.
   - **Deep learning model execution** → TensorFlow (TF) was too heavy for the Raspberry Pi. The MobileNet model was converted into **TensorFlow Lite (TFLite)** to run efficiently on the embedded hardware.
   - **Display differences** → On the laptop, the GUI (OpenCV windows) could be used for debugging. On the Raspberry Pi, the system was configured to run in **headless mode** (HEADLESS=1) without GUI, as in-vehicle setups typically do not use a monitor.

2. **Performance and Resource Limitations**

   The Raspberry Pi 4, despite being powerful for its size, has limited CPU and memory resources compared to a laptop. Running real-time image processing and deep learning inference required optimisation, such as:

   - Reducing image resolution to 640 × 480.
   - Limiting the number of threads for TFLite inference.
   - Applying smoothing and thresholds to reduce unnecessary computations.

3. **Library and Software Compatibility**

   Some Python libraries worked differently or needed special installation on the Raspberry Pi.

   - pyttsx3 worked differently depending on available voices; additional adjustments were needed to select a clear English voice on Raspberry Pi OS.
   - playsound, which worked on the laptop, was unreliable on the Raspberry Pi. This was replaced with subprocess calls to paplay or aplay for stable sound output.
   - Dlib installation was more complex on Raspberry Pi due to compilation requirements.

4. **Hardware Integration Challenges**

   - Pairing and maintaining a stable connection with the **MOXOM SK-39 Bluetooth speaker** sometimes required reconfiguration of the default audio sink.
   - The **18650-battery shield** provided stable power, but current limitations required careful testing to ensure the Raspberry Pi would not crash under load.

5. **Environmental Constraints**

   - The system needed to function under different lighting conditions (day, night, low light). This required adjusting camera placement and relying on both EAR/MAR and MobileNet classification to maintain accuracy.
   - Vibrations and movement inside the vehicle affected camera stability, making secure mounting essential for consistent landmark detection.

5.6 Concluding Remarks

The implementation of the fatigue detection system on the Raspberry Pi 4 successfully integrated both hardware and software components into a compact and portable in-vehicle solution. The system was designed to operate in real time, capturing the driver's facial features through the Logitech Brio 100 camera, analysing fatigue indicators using Dlib, EAR/MAR equations, head pose estimation, and MobileNet classification, and generating timely alerts through a Bluetooth speaker.

Several challenges were encountered during implementation, such as library compatibility, TensorFlow to TensorFlow Lite conversion, and audio format adjustments from .mp3 to .wav. These issues were resolved through careful configuration, optimisation, and testing, ensuring that the system could run efficiently on the Raspberry Pi. The use of a systemd service further enhanced practicality by enabling the program to launch automatically on startup, making the system suitable for real driving conditions without manual intervention.

Overall, the system implementation demonstrates that low-cost embedded platforms can effectively support advanced fatigue detection techniques. The completed setup achieves the project objectives of real-time monitoring and timely alerts, laying the foundation for further testing, performance evaluation, and improvements in subsequent chapters.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

The system was tested under real driving conditions during both daytime and night-time to evaluate its performance in different lighting environments. The camera was mounted on the windshield facing the driver, while the Raspberry Pi processed the video in real time and triggered alerts through the Bluetooth speaker.

- Daytime Testing:

Figure 6.1(a) shows the system setup during daytime driving, where the camera captured the driver's face under natural lighting conditions.

- Night-Time Testing

Figure 6.1(b) shows the system setup during night-time driving, where the system relied on cabin lighting to detect facial features in low-light conditions.

Final Method Result

The final approach, which combines **MobileNet, EAR & MAR, and head pose estimation**, achieved **high accuracy across all conditions**. By integrating both deep learning and geometric-based methods, the system was able to detect fatigue reliably in normal and low-light scenarios, as well as when small variations in head pose occurred.

Comparison with Other Methods

To justify the choice of the combined approach, a comparison was conducted against MobileNet-only and EAR & MAR-only methods. The results are shown in Table 6.1.

Table 6.1: Detailed Comparison of Fatigue Detection Methods

Method	**Accuracy**	**Strengths**	**Weaknesses**
MobileNet Only	Normal light: 9/10 Low light: 7/10 Accuracy = **80%**	-Good at detecting closed eyes and yawns -Works well in low light -Lightweight and runs on Raspberry Pi	-Slow for blink and yawn frequency -Weak against motion blur or occlusion
EAR&MAR Only	Normal light: 10/10 Low light: 6/10 Accuracy = **80%**	-Fast and low-computation -Tracks blink and yawn duration in real-time -Easy to implement	Sometimes fails with spectacles and low light conditions
MobileNet + Head Pose	Normal light: 9/10 Low light: 8/10 Accuracy = **85%**	-Adds head tilt detection -Detects microsleeps better -More robust than single model	Still can't track blink frequency
EAR&MAR + Head Pose	Normal light: 10/10 Low light: 6/10 Accuracy = **80%**	-Fast blink/yawn tracking + posture -Lightweight -Good for real-time alerts	Sometimes fails with spectacles and low light conditions
MobileNet + EAR&MAR + Head Pose	Normal light: 10/10 Low light: 9/10 Accuracy = **95%**	-Combines deep learning with real-time blink/yawn tracking and head tilt detection -Strong against lighting changes, spectacles, and face angles	Combining outputs from different methods increases implementation complexity

Explanation of Accuracy Levels

The accuracy values (e.g., **9/10**) represent the results of **10 real-time video trials**, where the system's detection was compared against actual observed outcomes.

- Methods achieving **60–70% accuracy** are considered moderately reliable.
- Methods in the **80–90% range** are considered highly reliable.
- Methods achieving **above 90% accuracy** are considered very reliable and suitable for real-world deployment.

It is important to note that this evaluation was conducted on a **laptop under indoor testing conditions** to compare the effectiveness of different methods. The combined approach (MobileNet + EAR & MAR + Head Pose) achieved the highest and most consistent accuracy across both normal and low-light scenarios, demonstrating its robustness and suitability for practical use.

6.2 Testing Setup and Results

To evaluate the accuracy of the fatigue detection system, videos were recorded under both **daytime (normal light)** and **night-time (low light)** conditions during **real-world driving**. For each fatigue indicator including blinking frequency, yawning frequency, eyes closed duration, and head tilt, five test videos were taken. The accuracy is presented in a ratio format (e.g., 4/5 means the system correctly detected fatigue in 4 out of 5 test videos).

Additionally, the system was tested across three fatigue levels (mild, moderate, and severe), also based on five recorded trials per stage.

- **Daytime Testing Results**

Table 6.2: Accuracy of Fatigue Detection (Daytime)

Fatigue Indicator	**Accuracy (5 Trials)**
Blinking Frequency	4/5
Yawning Frequency	5/5
Eyes Closed	5/5
Head Tilt	5/5

Accuracy of Fatigue Indicator = **95%**

Table 6.3: Fatigue Level Accuracy (Daytime)

Fatigue Level	**Accuracy (5 Trials)**
Mild	5/5
Moderate	5/5
Severe	4/5

Accuracy of Fatigue Level = **93%**

- **Night-time Testing Results**

Table 6.4: Accuracy of Fatigue Detection (Night-time)

Fatigue Indicator	**Accuracy (5 Trials)**
Blinking Frequency	4/5
Yawning Frequency	4/5
Eyes Closed	5/5
Head Tilt	4/5

Accuracy of Fatigue Indicator = **85%**

Table 6.5: Fatigue Level Accuracy (Night-time)

Fatigue Level	**Accuracy (5 Trials)**
Mild	4/5
Moderate	4/5
Severe	4/5

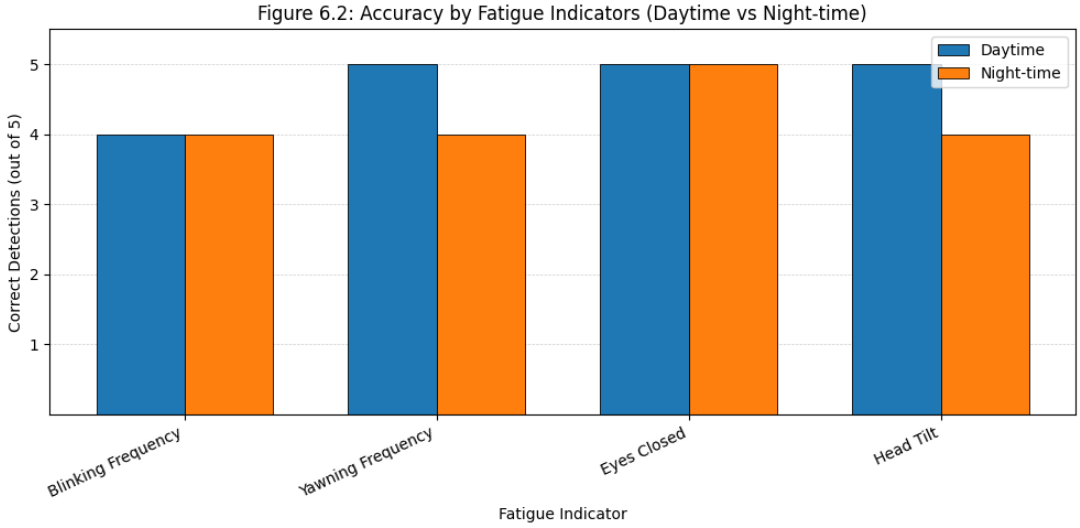
Accuracy of Fatigue Level = **80%**

Explanation

- The ratio format (e.g., 4/5) means that the system was tested using **five videos for each condition**, and the value shows how many times the system correctly detected the intended fatigue sign.
- Results indicate that the system performs more accurately in **daytime conditions** compared to **night-time**, though the combined MobileNet + EAR/MAR + Head Pose approach still provides consistent detection across both scenarios.

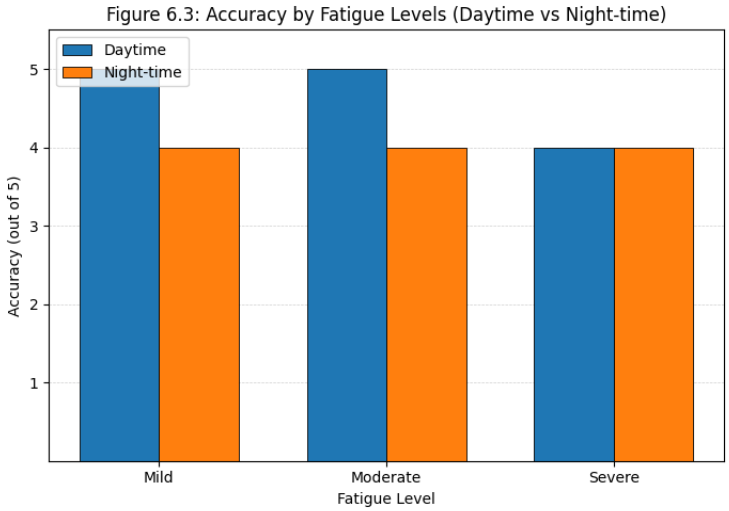
Comparison of Daytime and Night-Time Accuracy

Figure 6.2: Accuracy by Fatigue Indicators (Daytime vs Night-time)



This figure illustrates the detection accuracy for different fatigue indicators, including blinking frequency, yawning frequency, eyes closed duration, and head tilt. The results show that daytime performance is consistently higher than night-time, with clearer visibility of facial features improving accuracy.

Figure 6.3: Accuracy by Fatigue Levels (Daytime vs Night-time)



This figure compares the accuracy of fatigue stage classification (mild, moderate, severe) under daytime and night-time conditions. Similar to the indicator results, daytime testing achieved higher accuracy, while night-time accuracy was slightly reduced due to low lighting, camera quality limitations, and environmental factors.

6.3 Project Challenges

During the development and implementation of the fatigue detection system, several challenges were encountered:

1. **Code Portability to Raspberry Pi**

   The Python code developed on a laptop could not be directly transferred to the Raspberry Pi. Adjustments were required to ensure compatibility, such as converting TensorFlow models to TensorFlow Lite for lightweight execution and replacing .mp3 audio files with .wav format, which is supported by the Raspberry Pi audio drivers.

2. **Camera Limitation**

   The Logitech Brio 100 webcam performed well during indoor testing in controlled environments such as a lab or room, where lighting conditions were stable. However, during real driving at night, the camera struggled under very low-light cabin conditions. This limitation reduced the accuracy of detecting key fatigue indicators such as eye closure and yawning, showing that while the camera is suitable for testing in bright environments, it is less reliable in actual night-time driving scenarios.

3. **Bluetooth Speaker Integration**

   Setting up the MOXOM SK-39 Bluetooth speaker was challenging, especially in making sure the Raspberry Pi always sent alert sounds to the correct audio output. Modifications to PulseAudio settings and environment variables were required to achieve a stable connection.

4. **Testing in Real Driving Conditions**

   Conducting real-world tests inside a moving vehicle introduced challenges such as vibration, sunlight glare, and very low lighting at night. These conditions sometimes affected detection accuracy and highlighted the importance of testing across different environments to verify system robustness.

6.4 Objective Evaluation

The project objectives outlined in Chapter 1 were evaluated against the implementation and testing outcomes. The results demonstrate that all objectives were successfully met:

1. **To develop a real-time fatigue detection system using Python and deep learning on an embedded platform like Raspberry Pi.**

   This objective was achieved by implementing the complete system on the Raspberry Pi 4 Model B. The system processed video input in real time using Python, integrating deep learning models with traditional algorithms. Testing confirmed that the system operated effectively on the embedded platform without external dependency on a laptop.

2. **To integrate deep learning with facial landmark detection (EAR and MAR) for accurate fatigue monitoring in various conditions.**

   This objective was achieved by combining MobileNet with EAR, MAR, and head pose estimation. The hybrid approach improved accuracy across both daytime and night-time conditions, as demonstrated in the testing results (Tables 6.2, 6.3, 6.4 and 6.5). Compared to single-method approaches, the combined method provided stronger robustness against lighting variations, facial angles, and distractions.

3. **To implement a responsive alert system that activates sound warnings when drowsiness is detected.**

   This objective was achieved by integrating a MOXOM SK-39 Bluetooth speaker for audio output. The system triggered alerts according to fatigue severity: voice message for mild fatigue, voice plus alarm for moderate fatigue, and extended alarm for severe fatigue. Alerts were executed on separate threads to ensure that sound playback did not interrupt real-time video processing.

Conclusion:

All project objectives were successfully achieved. The integration of deep learning with traditional methods provided high detection accuracy, the Raspberry Pi ensured portability and in-vehicle operation, and the alert system offered timely warnings to the driver.

6.5 Concluding Remarks

The evaluation confirmed that the fatigue detection system works effectively in real driving conditions. By combining MobileNet with EAR, MAR, and head pose estimation, the system achieved reliable detection accuracy during both daytime and night-time testing. This hybrid approach proved more consistent than using MobileNet or EAR and MAR alone.

During development, several challenges were encountered, including code portability from laptop to Raspberry Pi, library and dependency compatibility, Bluetooth audio integration with the MOXOM SK-39 speaker, and testing in real driving conditions with vibration, glare, and low-light environments. Each of these issues required adjustments such as converting TensorFlow models to TensorFlow Lite, replacing unsupported libraries, configuring PulseAudio settings, and tuning detection thresholds to maintain performance.

In conclusion, the project successfully met its objectives by delivering a practical, real-time fatigue detection system with timely audio alerts to improve driver safety. The results also highlight opportunities for future refinement, such as expanding datasets and improving model efficiency for even greater reliability.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

This project successfully developed a real-time fatigue detection system on an embedded platform using the Raspberry Pi 4. The system integrates deep learning through MobileNet with geometric-based methods such as EAR, MAR, and head pose estimation to monitor driver drowsiness. By combining these approaches, the system achieved high accuracy in detecting fatigue indicators including blinking frequency, yawning, prolonged eye closure, head tilt, and distraction.

Testing in both daytime and night-time driving conditions demonstrated that the hybrid method outperformed single-method approaches, maintaining reliable accuracy across different environments. The system was able to classify fatigue into mild, moderate, and severe levels and trigger appropriate alerts through a Bluetooth speaker, ensuring timely warnings for the driver.

Despite facing challenges such as adapting the code to the Raspberry Pi environment, resolving library compatibility issues, integrating Bluetooth audio, and ensuring robust detection under real driving conditions, the system met all project objectives. The outcomes confirm that the developed solution is both practical and feasible for in-vehicle deployment, contributing towards improving road safety by reducing fatigue-related risks.

7.2 Recommendation

Based on the system development and testing, the following recommendations are proposed for future improvements:

1. **Improve Low-Light Performance** – Use infrared (IR) cameras or night-vision modules to enhance detection in very dark conditions.
2. **Expand Dataset** – Collect more diverse driver data across age groups, lighting conditions, and driving scenarios to strengthen model robustness.
3. **Conduct Large-Scale Testing** – Test the system with more drivers and in longer driving sessions to validate accuracy and reliability in real-world conditions.
4. **Enhance Alert Mechanisms** – Improve the clarity of voice messages and consider adding alternative alert modes such as vibration or LED indicators for better driver awareness.

REFERENCES

[1]Arnav Poswal, Aayushi Golchha, and Hema M, “FocusFleet: Driver Drowsiness Detection using CNN mobilenetv2,” Jan. 2025, doi: https://doi.org/10.2139/ssrn.5172891.

[2]D. Wagh, J. Hire, Mayuri Phad, Ajay Bhosale, Chudaman Sukte, and Manohar Kodmelwar, “Driver Drowsiness Detection Using MobileNetV2 and Deep Learning,” pp. 911–917, Dec. 2024, doi: https://doi.org/10.1109/icscna63714.2024.10864200.

[3]T. D. Wunan, Pretty Calista Jappy, S. Aurelia, I. S. Edbert, and Derwin Suhartono, “Driver Drowsiness Detection Using NasNet Mobile, MobileNetV2, and EfficientNetB0,” pp. 1–4, Feb. 2024, doi: https://doi.org/10.1109/aims61812.2024.10512773.

[4] K. R. M. Khariol and H. Jabbar, "Driver Drowsiness Detection with an Alarm System using a Webcam," *Evolution in Electrical and Electronic Engineering*, vol. 4, no. 1, pp. 87-96, 2023, doi: 10.30880/eeee.2023.04.01.011.

[5]P. Awasekar, M. Ravi, S. Doke, and Z. Shaikh, “Driver Fatigue Detection and Alert System using Non-Intrusive Eye and Yawn Detection,” *International Journal of Computer Applications*, vol. 180, no. 44, pp. 1–5, May 2018, doi: https://doi.org/10.5120/ijca2018917140.

[6] "Realtime Drowsiness and Yawn Detector using Raspberry Pi," *Instructables*, accessed Sep. 2023. [Online]. Available: https://www.instructables.com/Realtime-Drowsiness-and-Yawn-Detector-Using-Raspbe/

[7] D. Wu, *Improving Automatic Detection of Driver Fatigue and Distraction Using Machine Learning*, M.Sc. thesis, School of Computer Science, University of Birmingham, Dubai, United Arab Emirates, Sept. 2023.

[8] T. Danisman, I. M. Bilasco, C. Djeraba, and N. Ihaddadene, “Drowsy driver detection system using eye blink patterns,” *IEEE Xplore*, Oct. 01, 2010. https://ieeexplore.ieee.org/abstract/document/5648121 (accessed Feb. 19, 2022).

[9] O. J.A, O. L.T, and A. I.A, “Fatigue Detection in Drivers using Eye-Blink and Yawning Analysis,” *International Journal of Computer Trends and Technology*, vol. 50, no. 2, pp. 87–90, Aug. 2017, doi: https://doi.org/10.14445/22312803/ijctt-v50p115.

[10] “Drowsiness Detection for Vehicle Drivers with Alert System,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 5, pp. 9125–9130, Oct. 2020, doi: https://doi.org/10.30534/ijatcse/2020/319952020.

[11] M. Zhang and F. Zhang, “A Driver Fatigue Detection Method Based on Eye Aspect Ratio Compensated by Head Pose Estimation,” *2022 7th International Conference on Signal and Image Processing (ICSIP)*, vol. 59, pp. 161–165, Jul. 2023, doi: https://doi.org/10.1109/icsip57908.2023.10270911.

[12] Y. Wang, B. Liu, and H. Wang, “Fatigue detection based on facial feature correction and fusion,” *Journal of Physics: Conference Series*, vol. 2183, no. 1, p. 012022, Jan. 2022, doi: https://doi.org/10.1088/1742-6596/2183/1/012022.

[13] K. Ju, B.-S. Shin, and R. Klette, “Novel Backprojection Method for Monocular Head Pose Estimation,” *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 13, no. 1, pp. 50–58, Mar. 2013, doi: https://doi.org/10.5391/ijfis.2013.13.1.50.

[14] L. Yu, X. Yang, H. Wei, J. Liu, and B. Li, “Driver fatigue detection using PPG signal, facial features, head postures with an LSTM model,” *Heliyon*, vol. 10, no. 21, pp. e39479–e39479, Oct. 2024, doi: https://doi.org/10.1016/j.heliyon.2024.e39479.

[15] M. Ye, W. Zhang, P. Cao, and K. Liu, “Driver Fatigue Detection Based on Residual Channel Attention Network and Head Pose Estimation,” *Applied Sciences*, vol. 11, no. 19, p. 9195, Oct. 2021, doi: https://doi.org/10.3390/app11199195.

[16] I.-H. Choi, C.-H. Jeong, and Y.-G. Kim, “Tracking a Driver’s Face against Extreme Head Poses and Inference of Drowsiness Using a Hidden Markov Model,” *Applied Sciences*, vol. 6, no. 5, p. 137, May 2016, doi: https://doi.org/10.3390/app6050137.

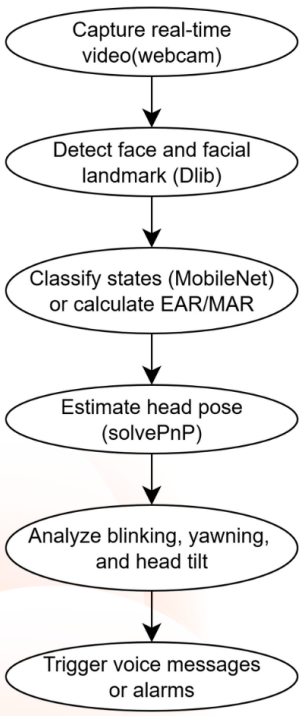
POSTER

FATIGUE DETECTION SYSTEM IN CARS

INTRODUCTION

This project aims to develop a real-time ststem that **detects driver fatigue** by analyzing eye closure, yawning, and head posture, providing early alerts to enhance read safety

METHODOLOGY



OBJECTIVES

- Develop a real-time fatigue detection system on **Raspberry Pi**.
- Integrate **deep learning** with **fatigue landmark detection** for monitoring.
- Implement a sound-based **alert system**.

RESULTS

The system was tested in both **daytime** and **night-time** driving conditions.

Fatigue Indicators (blinking, yawning, eyes closed, head tilt):
- Daytime Accuracy: **95%**
- Night-time Accuracy: **85%**

Fatigue Levels (mild, moderate, severe):
- Daytime Accuracy: **93%**
- Night-time Accuracy: **80%**

CONCLUSION

- A real-time fatigue detection system was **successfully** developed on **Raspberry Pi**.
- By combining MobileNet, EAR & MAR, and head pose, the system **achieved higher accuracy** and robustness compared to using a single method.
- Testing **confirmed** the system can deliver **timely alerts** to reduce risks of driver drowsiness.

Developer:
Vanicha Kulma (CT)

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

Supervisor: Ts Dr Vikneswary a/p Jayapal
Moderator: Ms Oh Zi Xin