
Drivers Drowsiness Detection

A Senior Project Submitted To:
The College of Computers and Information Technology
in Candidacy for the Degree of
Bachelor's Degree
in Computer Engineering
by

Abdulkarim Alahmari - 44003245

Ahmed Alrashidi- 44151822

Saad Alharithi - 44000520

Ammar Alsufyani- 44007668

Abdullah Althrmanni - 44104940

Turki Altuwairqi - 44002075

Hamza Alzahrani – 43904719

Ahmed Alharithi - 44006405

Supervised by:

Dr. Hattan Althebeiti

Department of Computer Engineering

College Computers and Information Technology


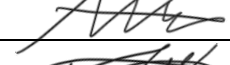
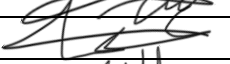



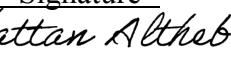

Taif University, KSA

Drivers Drowsiness Detection

Document History:

Version Number	Issued Date	Authors	Descriptions	Signature
1	9/10/2023		Abstract	
2	8/10/2023		Problem background, problem Statement	
3	7/10/2023		Objectives, scope	
4	5/10/2023		Significance	
5	6/10/2023		Activity plan	
6	23/10/2023		Literature Review	
7	25/10/2023		Methodology	
8	13/11/2023			
9	17/11/2023			
10	25/11/2023			
11	2/12/2023			

Team Members:

Student Name	Student ID	Date	Signature
Abdulkarim Alahmari	44003245	11/25/2024	
Ahmed Alrashidi	44151822	11/25/2024	
Saad Alharithi	44000520	11/25/2024	
Ammar Alsufyani	44007668	11/25/2024	
Abdullah Althmani	44104940	11/25/2024	
Turki Altuwairqi	44002075	11/25/2024	
Hamza Alzahrani	43904719	11/25/2024	
Ahmed Alharithi	44006405	11/25/2024	

Supervised by: Dr. Hattan Althebeiti

<Date>

<Signature>

11-25-2024

Hattan Althebeiti



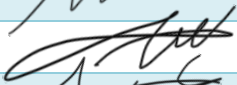


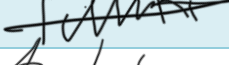
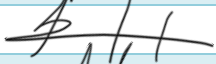

Approved by Supervisor Committee:

Supervisor Name	Date	Signature

Department of Computer Engineering
Faculty of Computers and Information Technology
Taif University, KSA

Contact Information

Authors Information

	Name		Number		Email
1.	Abdulkarim Alahmari				s44003245@students.tu.edu.sa
2.	Ahmed Alrashidi				s44151822@students.tu.edu.sa
3.	Saad Alharithi				s44000520@students.tu.edu.sa
4.	Ammar Alsufyani				s44007668@students.tu.edu.sa
5.	Abdullah Althrmmani				s44104940@students.tu.edu.sa
6.	Turki Altuwairqi				s44002075@students.tu.edu.sa
7.	Hamza Alzahrani				s43904719@students.tu.edu.sa
8.	Ahmed Alharithi				s 44006405@students.tu.edu.sa

Supervisor

Name: Dr. Hattan Althebeiti

Department of Computer Engineering

Students' Property Right Declaration

I hereby declare that the work in this capstone project at Taif University is my own except for quotations and summaries which have been duly acknowledged. This work with the title **Drivers Drowsiness Detection** has not been accepted for any degree and is not concurrently submitted for award of other degrees. It is the sole property of Taif University and it is protected under the intellectual property right laws and conventions.

Authors:

Name: Abdulkarim Alahmari	Signature: 	11/25/2024
Name: Ahmed Alrashidi	Signature: 	11/25/2024
Name: Saad Alharithi	Signature: 	11/25/2024
Name: Ammar Alsufyani	Signature: 	11/25/2024
Name: Abdullah Althmani	Signature: 	11/25/2024
Name: Turki Altuwairqi	Signature: 	11/25/2024
Name: Hamza Alzahrani	Signature: 	11/25/2024
Name: Ahmed Alharithi	Signature: 	11/25/2024

Supervisor:

Name: Hattan Althebeiti

Signature:  Date: 11/25/2024


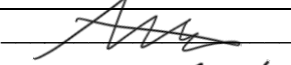
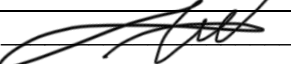


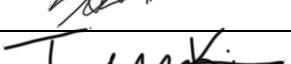
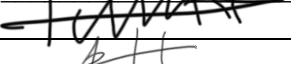
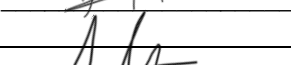
Students Anti-Plagiarism Statement

I hereby declare this report is my own work except for properly referenced quotations, and contains no plagiarism; it has not been submitted previously for any other assessed unit on this or other degree courses.

I have read and understood the School's rules on assessment offences which are available in the Taif University Handbook

انا الممضي أسفله أشهد أن هذا التقرير هو عملي الخاص انا و مجموعة الطلبة المذكورة أسماؤهم بأول هذا التقرير ما عدا ما هو مذكورة مصادره صراحة و أنه لا يحتوي على محتويات منقولة بدون عزوها لكتابتها الأصلي. و أشهد أن هذا العمل لم يسبق أن استخدم كعمل رسمي بمقررات اخرى بهذه الكلية أو غيرها. أشهد أنني اطلعت على قوانين الكلية الخاصة بتقييم الطلبة الموجود بالكتاب التقديمي للجامعة

Authors:

Name: Abdulkarim Alahmari	Signature: 	11/25/2024
Name: Ahmed Alrashidi	Signature: 	11/25/2024
Name: Saad Alharithi	Signature: 	11/25/2024
Name: Ammar Alsufyani	Signature: 	11/25/2024
Name: Abdullah Althrmari	Signature: 	11/25/2024
Name: Turki Altuwairqi	Signature: 	11/25/2024
Name: Hamza Alzahrani	Signature: 	11/25/2024
Name: Ahmed Alharithi	Signature: 	11/25/2024

Abstract

Road accidents are considered one of the main causes of death globally, as fatigue and irresponsible behavior of drivers cause many of these accidents. According to World Health Organization reports, approximately 1.25 billion people are injured annually as a result of traffic accidents. Hence, monitoring drivers' fatigue is an important opportunity to avoid many of these accidents. To solve this problem, a project was proposed to develop an artificial intelligence model that recognizes the fatigue state of drivers through eye movements, using yawning as a marker. This model will be installed in the vehicle as a camera that records the driver's movements and sends instant notifications when eye fainting or frequent yawning is detected, in cooperation with the Raspberry Pi control unit to process the data. Through this early intervention, the system aims to prevent reckless driving accidents and enhance road safety. This project also enables this project to reduce accident rates and provide important data to improve road safety policies.

Keywords: Eye movements, Yawning, Drowsiness, Machine learning, Camera.

تعتبر حوادث الطرق أحد الأسباب الرئيسية للوفاة على مستوى العالم، حيث يتسبب الإرهاب والسلوك غير المسؤول للسائقين في العديد من هذه الحوادث. ووفقا لتقارير منظمة الصحة العالمية، يصاب ما يقرب من 1.25 مليار شخص سنويا نتيجة لحوادث المرور. ومن هنا فإن مراقبة حالة إرهاب السائقين تعتبر فرصة مهمة لتجنب الكثير من هذه الحوادث. ولحل هذه المشكلة، تم اقتراح مشروع لتطوير نموذج ذكاء اصطناعي يتعرف على حالة الإرهاب للسائقين من خلال حركات العين، باستخدام التثاؤب كعلامة. وسيتم تثبيت هذا النموذج في المركبة ككاميرا تسجل حركات السائق وترسل إشعارات فورية عند اكتشاف إغماء العين أو التثاؤب المتكرر، وذلك بالتعاون مع وحدة تحكم Raspberry Pi لمعالجة البيانات. يهدف النظام من خلال هذا التدخل المبكر الى منع حوادث القيادة المتهورة وتعزيز السلامة على الطرق، كما يمكن هذا المشروع من تقليل معدلات الحوادث وتقديم بيانات مهمة لتحسين سياسات السلامة على الطريق.

الكلمات المفتاحية : حركات العين، التثاؤب، النعاس، التعلم الآلي، الكاميرا.

Contents

Contact Information	iii
Students' Property Right Declaration	iv
Students Anti-Plagiarism Statement	v
Dedication	v
Abstract	vi
(عربي) البحث مقترح ملخص	vii
List of Tables	x
List of Figures.....	xi
List of Acronyms (Abbreviations) and Symbol.....	xii
Chapter 1: Introduction.....	1
1.1 Overview and Problem Background	1
1.2 Problem Statement	2
1.3 Objectives	2
1.4 Contributions/Significance of the Project	3
1.5 Scope.....	3
1.6 Engineering Standards	3
1.7 Realistic Constrains	4
1.8 Development Time Frame and Cost	5
Chapter 2: Literature Review.....	6
2.1 Overview of Related Technology	6
2.2 Summary of Related Works	8
2.3 Comparison of Related Works	10
Chapter 3: System Analysis and Design.....	11
3.1 System Model	11
3.2 Methods.....	12
3.2.1 CNN	12
3.2.2 YOLO.....	15
3.3 Requirement Specification	18
3.3.1 Functional and Non-Functional Requirements	18
3.3.2 Hardware Specifications.....	18
3.3.3 Software Specifications	21
3.3.4 Dataset.....	23
3.4 System Block Diagrams	23
3.5 Evaluation Metrics:	24
Chapter 4: Practical Implementation and Results.....	25
4.1 System Setup and Architecture	25
4.1.1 Hardware Components	25
4.1.2 Software Components	25
4.1.3 Raspberry Pi Setup and Configuration	27
To prepare the Raspberry Pi for system operation and model deployment, it is essential to install the required libraries and dependencies. These include the tools and software necessary to support real-time image processing, data analysis, and efficient system functionality, ensuring seamless and accurate performance.	30
4.1.4 Buzzer Connection and Configuration	30

4.2 Model Development and Training	32
4.2.1 Dataset.....	32
4.2.2 Data Distribution Visualization	33
4.2.3 Data Preprocessing.....	34
4.2.4 Model Architecture	34
4.2.5 Training Process and Performance Metrics	36
4.2.6 Evaluation and Validation	37
4.3 Model Deployment on Raspberry Pi.....	38
4.3.1 Alert System Modification	38
4.3.2 Real-Time Driver Drowsiness Detection	39
4.4 Results.....	40
4.4.1 Model Accuracy.....	40
4.4.2 Confusion Matrix Analysis.....	40
4.4.3 Practical Performance on Raspberry Pi.....	40
4.5 Model Testing with Sample Scenarios.....	40
4.5.1 Testing Scenarios	40
4.5.2 Accuracy of the System.....	41
4.6 Suitability of the Model for Women Wearing Niqab and Drivers Wearing Glasses	41
4.7 Limitations.....	42
4.7.1 Limitations.....	42
Chapter 5: Conclusion and Future Work	43
References.....	44

Name	Date	Reason For Changes	Version

List of Tables

<i>Table 1 Activity Plan</i>	5
<i>Table 2 Comparison of related of work</i>	10
Table 3 Model Architecture	34

List of Figures

Figure 1 System Model Diagram	11
Figure 2 CNN architecture [13]	13
Figure 3 yolo v3 architecture [17]	16
Figure 4 Raspberry Pi 4 module.....	19
Figure 5 Raspberry Pi Camera	20
Figure 6 Python logo	21
Figure 7 Flutter logo.....	21
Figure 8 Pandas logo	22
Figure 9 NumPy logo	22
Figure 10 TensorFlow logo.....	22
Figure 11: System Block Diagram	23
Figure 12 Raspberry Pi Imager	27
Figure 13 Raspberry Pi Imager (Choosing Device)	28
Figure 14 Raspberry Pi Imager (Choosing OS)	28
Figure 15 Raspberry Pi Imager (Choosing Storage Device).....	29
Figure 16 Camera Module Port [19]	29
Figure 17 40 GPIO Pins Description of Raspberry Pi 4 [20].....	31
Figure 18 DataSet Example: Open Eye	32
Figure 19 DataSet Example: Closed Eye.....	32
Figure 20 Training Set Distribution	33
Figure 21 Test Set Distribution	33
Figure 22 Accuracy over epochs	36
Figure 23 Loss over epochs	36
Figure 24 Classification Report	37
Figure 25 Confusion Matrix.....	38

List of Acronyms (Abbreviations) and Symbol

EEG – Electroencephalographic.

DDD – Driver Drowsiness Detection.

CNN – Convolutional Neural Network.

VGG – Visual Geometry Group

Chapter 1: Introduction

1.1 Overview and Problem Background

Driver safety is a global concern, as road accidents continue to claim numerous lives and cause severe injuries each year. Driver's fatigue is a significant factor that contributes to 35%-45% of all vehicle accidents [1]. Usually, rest and inactivity relieve fatigue, however, it makes drowsiness worse. Recent statistics estimate that annually 1.3 million deaths and 20-50 million injuries can be attributed to car crashes [2]. One of the biggest challenges in the field of accident-avoidance systems is the development of technology for the detection or prevention of drowsiness while driving. Different studies have been reported on driver drowsiness.

Many factors can be the root cause of drowsiness or fatigue in driving which include lack of sleep, long driving hours, consumption of alcohol, use of sedating medications, and some driving patterns like driving at midnight, early morning, mid-afternoon hours, and especially in a monotonous style of driving environment. Ultimately, it also depends on changes in driver's performance ability.

Prior research has put forth several techniques for identifying drowsiness and distraction [3]. These approaches can be classified into two primary categories. The first category focuses on observing physical indicators of fatigue, such as the driver's head leaning, relaxed posture, and weakened grip on the steering wheel. In order to detect these bodily movements, direct sensor contacts or video cameras are utilized. Thankfully, these methods enable contactless identification of drowsiness without causing any discomfort to the driver. Hence the driver is very open to accept the usage of these techniques to check on drowsiness. However, these parameters do not always remain the same when one considers different vehicle types and driving situations. The second method focuses on assessing drivers' physiological responses, including eye activity measurements, heart rate, skin conductivity, and Electroencephalographic (EEG) patterns. Research indicates that eye blink duration and blink frequency are highly indicative of fatigue effects.

The progress made in image recognition and computer vision has demonstrated their significance in the development of driver monitoring systems. These technologies enable systems to analyze and understand the visual data captured by in-vehicle cameras. Employing deep learning algorithms, these systems can effectively identify potential hazards and acquire valuable data concerning driver conduct [4]. Several studies have specifically emphasized the importance of promptly addressing drowsiness detection with the aid of facial landmarks and eye tracking. This research underscores the critical need to take immediate action in order to prevent accidents caused by sleep-deprived driving [5].

1.2 Problem Statement

In general traffic accidents occur due to several reasons such as high speed, drunk driving, night driving, bad weather conditions etc. Drowsiness of driving has been identified as a major problem and that is the root cause of most road accidents. That dangerous behavior increases the number of deaths and injuries in every year. Ensuring driver safety is of utmost importance in today's environment, where traffic accidents still pose a significant risk. To address this issue, a project has been developed using modern technology to monitor and enhance driver safety by detecting eye movement in real time and issuing alerts using artificial intelligence technology.

1.3 Objectives

In this project, we aim to achieve the following objectives:

- Develop an efficient and accurate driver monitoring system.
- Enhance the detection of driver distraction and fatigue.
- Utilize artificial intelligence and machine learning techniques for analyzing driving behavior and use eye tracking technology to detect driver distraction.

1.4 Contributions/Significance of the Project

By the end of this project, we expect to obtain the following benefits:

- **Enhanced Driver Safety:** The system will enhance driver safety by continuously monitoring driver behavior for signs of distraction and fatigue, alerting them to refocus on the road and potentially preventing accidents
- **Reduced traffic accidents:** The system can contribute in a reduction in traffic accidents by proactively identifying and addressing potential risks such as distracted driving and drowsiness, ultimately decreasing the overall number of traffic incidents and their associated consequences
- **Road Safety Awareness:** The project also seeks to raise road safety awareness among drivers by promoting responsible driving habits and emphasizing the importance of staying alert and focused while behind the wheel, fostering a culture of road safety within the community

1.5 Scope

The scope of the project includes developing a driver monitoring system aimed at enhancing road safety by detecting driver distraction and fatigue in real-time. The targeted audience for this system primarily includes drivers of various vehicles, such as cars, trucks, and buses, to improve their awareness and prevent accidents.

1.6 Engineering Standards

Below an example of “Engineering standards”:

- The International System of Units (SI):
 - Time (Seconds),
 - Voltage (Volt),
 - Current (Ampere),
 - Electric Resistance (Ohm Ω),
 - Frequency (Hz).
 - Temperature (Degree Celsius $^{\circ}$),
 - Angle (Degree $^{\circ}$) - although not an SI unit it is mentioned as an accepted unit -,
 - Torque (Newton-Meter N-m).

- RS-232: we used this serial communication standard to connect the Arduino board with PC in order to upload the program.
- Bluetooth 4.0: We used this standard for wireless communication between two devices in our project.
- NEMA 17 Stepper Motor: 4-wire, bipolar, step angle 1.8° , current 1.7A.
- Red-Yellow-Green: We used this color standard in our traffic lights project.

1.7 Realistic Constrains

- Economic: The cost of implementing the driver monitoring system may be a constraint, especially for widespread adoption in vehicles.
- Environmental: The disposal of electronic components and potential energy consumption of the system could have environmental impacts.
- Sustainability: Ensuring the long-term maintenance and updates of the system to keep up with evolving technology could be a challenge for sustainability.
- Manufacturability: Designing the system to be easily integrated into different vehicle models and ensuring mass production feasibility may pose constraints.
- Ethical: Issues related to privacy invasion by monitoring drivers continuously could raise ethical concerns.
- Health and safety: Ensuring that the system does not cause distractions or stress to the driver, impacting their health and safety, is a critical constraint to address.
- Social: Cultural acceptance of constant monitoring and potential resistance from drivers could be social constraints to consider.

1.8 Development Time Frame and Cost

The development time frame for the driver monitoring system project will be crucial in ensuring timely completion and successful implementation. This section outlines the planned schedule and milestones for the project, providing a roadmap for the efficient execution of tasks. Table 1. represents the activity plan for our project.

Table 1 Activity Plan

Month	February				March				April			
Week	1	2	3	4	5	6	7	8	9	10	11	12
Proposal												
Chapter 1												
Chapter 2												
Chapter 3												
Chapter 4												

Chapter 2: Literature Review

2.1 Overview of Related Technology

Technological advancements in fields such as computer vision and machine learning have aided in the identification of sleepiness. Driver inattention is a prominent cause of car accidents, typically caused by factors such as fatigue-induced drowsiness. Scientists have examined many means of detecting drowsiness including vehicular, behavioral, and biological measures. Several methods have been proposed, some of which leverage machine learning, computer vision, and other technological tools. Furthermore, improvements in vehicle components and vital signs have led to the development of more efficient systems for detecting drowsiness.

2.1.1 Facial Expression and Recognition.

One of the key elements of non-verbal communication is facial expressions. It is the expression of emotions, attitudes, and intentions through gestures and facial movements. A wide range of emotions such as sadness, joy, fear, anger, contempt, and surprise can be seen on a human's face. Often, the eyes, eyebrows, lips, and other facial features change alongside these emotions. Facial expressions are influenced by cultural, social, and personal factors and can be deliberate or spontaneous. Understanding facial expressions is useful in various fields such as communications, anthropology, psychology, and sociology. It can help us comprehend ourselves and others, as well as handle social situations more efficiently [6].

Some facial recognition systems detect faces by extracting landmarks or characteristics from images. For example, an algorithm may examine the relative location, size, and/or form of the eyes, nose, cheekbones, and jaw. These attributes are then used to find more photos with similar features. Other techniques normalize a collection of face photos before compressing the data, preserving just the info that is helpful for face detection. The probe picture is then compared to the facial data. One of the first effective systems

relies on template matching techniques applied to a collection of significant facial traits, resulting in a compressed face representation.

2.1.2 Driver Drowsy Detection System

Fortunately, it is possible to detect the early stages of driving drowsiness and inform the driver, preventing future accidents. Drowsiness in drivers can be identified by recurrent yawning, frequent eye closures, and drifting across traffic lanes. Indeed, there has been a lot of study on driver drowsiness detection (DDD) algorithms in recent years [7]. Researchers have proposed many strategies for detecting these indicators of drowsiness as quickly possible in order to avoid accidents.

In general, drowsy detection systems are classed as intrusion or non-intrusion techniques.

Intrusive approaches detect drowsiness by deploying sensors that trigger an alert system. Such systems require intrusive instrumentation equipment or temperature sensing electrodes. Other invasive ways include measuring brain waves, heart rate, eye blinking, drooping posture, leaning the driver's head, and the open/closed position of the eyes.

Non-intrusive methods detect drowsiness without affecting the driver's body. Non-intrusive approaches include measuring arterial blood supply, driver reaction at regular intervals, lane departure, yawn rate computation, optical sensor, LED, video camera, head movements, eyelid movement, and Doppler components such as velocities and angles.

These techniques can also be divided into four main categories: image-based methods, which analyze the driver's movements and facial expressions using a camera; biological-based methods, which track the driver's physiological signals through the use of specialized sensors attached to the driver's body; vehicle-based methods, which monitor the behavior and motion of the vehicle; and hybrid methods, which combine two or more of these techniques.

2.2 Summary of Related Works

Several research have focused on the problem of detecting drivers' distraction behaviors and proposed wide range of solutions to that problem.

Harshit Verma presents [8] a driver drowsiness detection system based on CNN-machine learning algorithm which is implemented completely offline and can alert with the help of alarm if the driver is feeling drowsy. The system comprises the following components: Webcam, Image Resizing, Haar Classifier, Convolutional Neural Network model, and Eye Region of Interest. The system is trained and tested on a Drowsiness Dataset from the Kaggle platform, which contains 2900 images of eyes in two categories: open and closed. The system achieved a classification accuracy of over 98% on both the training and test datasets, and successfully identified the drivers who were drowsy, and alerted them with a visual and auditory warning. The paper concludes that driver drowsiness detection systems can significantly reduce the number of accidents caused by driver fatigue, leading to a considerable reduction in fatalities and injuries on the roads.

Ahmed et al. [9] proposes a method for evaluating the level of driver fatigue based on changes in a driver's eyeball movement using a convolutional neural network (CNN). The paper also uses a VGG16 model to detect and classify facial sleepiness expressions into four categories (open, closed, yawning, and no yawning). The paper uses a dataset of 2900 images of eye conditions associated with driver sleepiness, which include different features such as gender, age, head position, and illumination. The paper reports that the CNN model achieved an accuracy rate of 97%, a precision of 99%, and recall and F-score values of 99%, while the VGG16 model reached an accuracy rate of 74%. The paper concludes that the proposed model outperforms the state-of-the-art methods in the literature for similar problems and has the potential to improve road safety by detecting driver drowsiness.

Ahuja et al. [10] presented a solution to driver drowsiness detection problem using deep learning and knowledge distillation technique, which reduced the size and complexity of the deep learning models while maintaining high accuracy. The paper used ZJU dataset,

which contains low resolution eye images, and extracts the eye from the driver's face image using dlib facial landmark detector. The paper adopted modified versions of VGG-19 and VGG-16 as the teacher and student models respectively, and trains the student model using the softened probabilities from the teacher model as soft-targets. The paper reported that the student model with 2 million parameters achieves 95% accuracy on the ZJU dataset and can be efficiently deployed on embedded boards for real time driver drowsiness detection.

Chowdhury et al. [11] proposed a method to detect driver drowsiness in real-time using a CNN model that classifies eye states as open or close. In this research, the CenterFace method is utilized to recognize faces, while the Haar-Classifer is employed to extract eye features. A CNN architecture has been constructed and trained using the MRL big eye dataset. It is determined whether or not the driver is drowsy throughout the decision-making process. If a driver is detected to be drowsy an alarm will ring to inform them. The training and validation accuracy are 97% and 98%, respectively. The test accuracy for non-glassed and glassed eye datasets is 97% and 92%, respectively.

2.3 Comparison of Related Works

The below table summarizes a comparison between the related work and our project.

Table 2 Comparison of related of work

Work	Methodology	Advantages	Issues	Dataset	Metric used
[8]	CNN-machine learning algorithm	High accuracy	Further testing is needed to evaluate the effectiveness of the system under different lighting conditions and skin tones	Drowsiness Dataset from the Kaggle platform.	Accuracy =98%.
[9]	CNN and VGG16 models	Effective and High accuracy.	The system's performance may vary depending on the face orientation, illumination, and wearing obstructions of the driver.	Dataset from Kaggle	Accuracy =97%
[10]	Knowledge distillation technique using two separate CNN models (teacher model and student model) is used along with the OpenCV image detection technique	Highly accurate, lighter than using a cumbersome CNN model	Higher computation time	Image dataset	Accuracy=87% speed
[11]	CenterFace algorithm, Haar-Classifer	high accuracy rate and can detect whether the driver is drowsy even if it wears glass or mask	The accuracy of our CNN model sometimes decreases for the eye with wearing glasses	The dataset consists of a total of 84,898 images	Accuracy=95%, sensitivity and specificity

Chapter 3: System Analysis and Design

In this Chapter, the focus is on the detailed analysis of the requirement specifications for the development and implementation of our model. This includes a comprehensive overview of both functional and non-functional requirements essential for the successful operation of the system. The chapter also delves into the Hardware Specifications necessary for the installation and functioning of the model. Moreover, the Software Specifications are thoroughly discussed in this chapter, outlining the software requirements needed for the development and operation of the model.

3.1 System Model

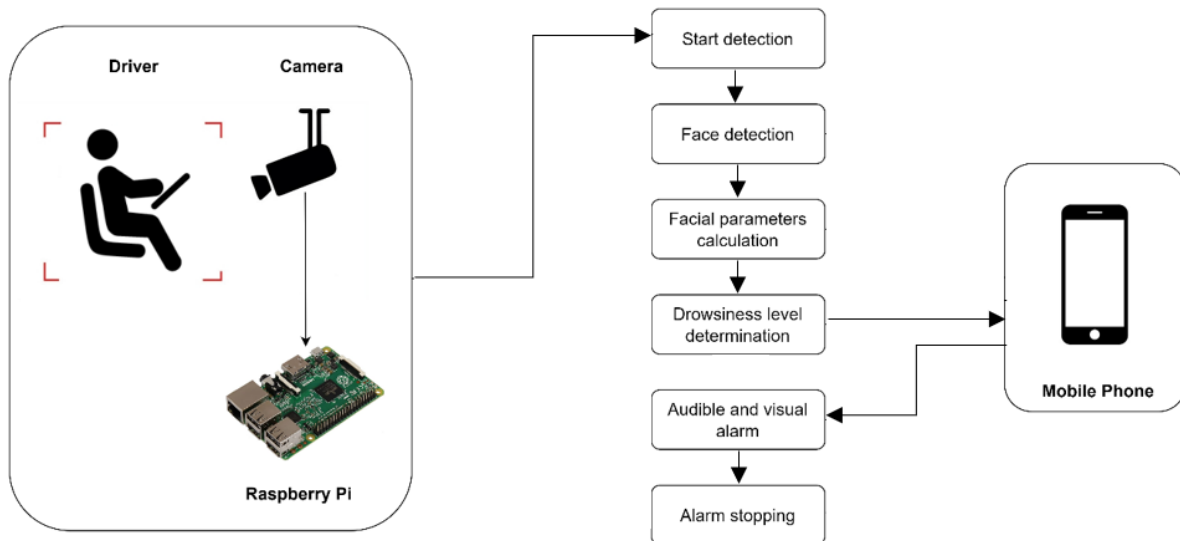


Figure 1 System Model Diagram

Figure 1 illustrates the system model for our project. The system uses a camera to monitor the driver, which is connected to a Raspberry Pi for processing the images. The process involves several steps including starting detection, face detection, facial parameters calculation, and drowsiness level determination. If the system determines that the driver is drowsy, it triggers an audible and visual alarm on a mobile phone. The system also includes a mechanism for stopping the alarm. This model represents a comprehensive approach to detect driver drowsiness and alert them in real-time to prevent accidents.

3.2 Methods

Convolutional Neural Networks (CNNs) and You Only Look Once (YOLO) are two powerful deep learning architectures commonly employed in computer vision tasks, including object detection and classification. In the context of our project, both CNNs and YOLO offer valuable techniques for analyzing visual data captured by the driver monitoring system.

3.2.1 CNN

CNNs have been extensively utilized in various driver monitoring systems for tasks such as facial recognition, eye tracking, and emotion detection. By leveraging the hierarchical feature representation learned during training, CNNs can effectively identify patterns indicative of driver fatigue, such as drooping eyelids or frequent yawning.

Convolutional layers are used in the CNNs to mimic the functionality of receptive cells of the visual cortex. Hence, the CNN model is powered by the usage of the convolution operation. A typical CNN structure is made up of one to multiple convolutional layers. The pooling/subsampling layers are usually placed after these layers [12]. At the top of the model, a final classification set of layers called fully connected layers are placed. The basic architecture of the CNN model is shown in Figure 2. There are two basic operations performed by CNNs: feature learning and classification. Feature learning is accomplished by a set of convolutional layers, pooling layers, and activation functions (commonly ReLU). For classification, fully connected layers with a sigmoid or a softmax function is used. In the traditional neural network, each neuron is connected with every other neuron, whereas in CNN, only the last layer is fully connected, which is used for classification.

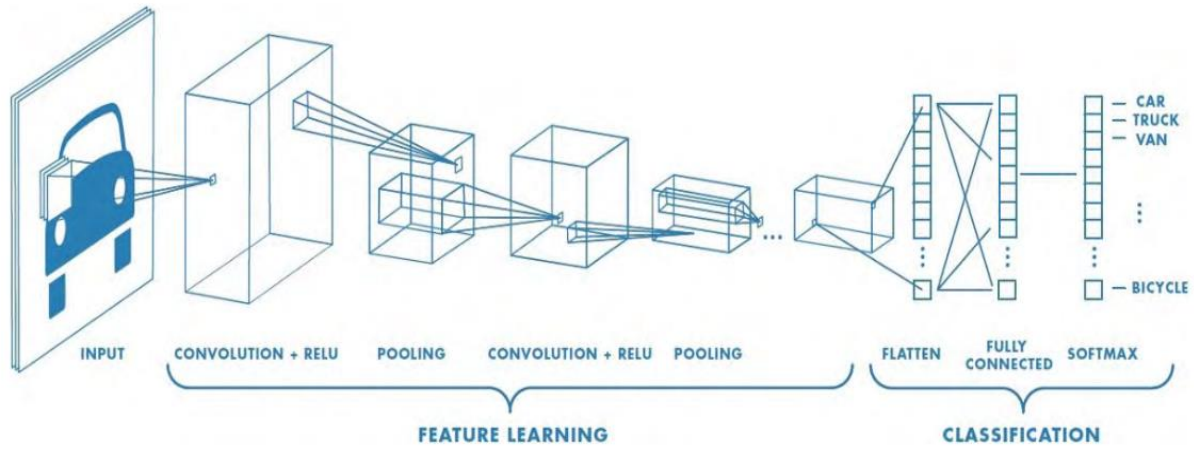


Figure 2 CNN architecture [13]

The basic building blocks of CNN are explained as follows:

1. Convolutional layers

Convolutional layers enable a CNN model to be repurposed as a feature extractor. Unlike traditional features (color or texture), the CNN features are not handcrafted, and are determined and computed by the network model directly [14]. During the training phase, the convolutional layers decide the weights of the filter kernel. These weights are saved and loaded into a new model.

A deep CNN model can extract various features as it progresses the input image through a set of convolutional layers. Initial layers are known to compute low-level descriptors, especially edges, lines, and corners. The upper set of layers derives high-level descriptors. The element involved in carrying out the convolution operation is called the kernel or filter. A matrix multiplication operation is performed between the receptive field of a matrix and a filter, where the filter is always smaller than the input data. The kernel is slid from the top left-hand corner towards the right end through each pixel in the image. This is continued for each row till the end of the image until we reach the final pixel at the bottom. This single sliding operation creates one feature. Similarly, multiple features are computed by applying various filters [8].

2. Activation functions

CNN models broadcast indications when they identify a probable feature from hidden layers. This is done through non-linear activation functions. The Rectified Linear Unit (ReLU) is an activation function that is the most frequently used function in CNN-based deep learning models [12]. ReLU is applied after each convolution, where it replaces the negative values with zeros. It is applied to each pixel of the feature map and replaces the zero wherever it finds the negative values. While the other values remain the same in the feature map. ReLU is known to speed up the training time of the model as compared to different activation functions (like sigmoid, hyperbolic, etc.) [10, 13].

3. Pooling/Subsampling layers

The convolutional layer generates a high number of features. They are reduced further by the pooling layer through the subsampling process. To reduce the feature map dimensionality, the pooling layer is applied after the activation layer [14]. This significantly reduces the number of parameters learned by a model through a reduction in the spatial size of an image. Therefore, pooling is used for downsampling the input representations. The pooling does not affect the depth of the matrix and mainly performs to extract the dominant features from an input image. The most common approaches used in pooling are max pooling and average pooling [15]. These processes create compactness by dividing data into 2D spaces that do not overlap. In the max-pooling method, the maximum of the underlying data is considered. Whereas in the average-pooling method, the average of underlying data is considered.

4. Fully Connected layers

The fully connected layers receive the feature map generated from the convolution/pooling process as input and perform the final class predictions using the softmax layer [16]. They are situated in the classifier of the model, where each neuron of the previous layer is connected to each neuron of the subsequent layer. These layers receive the flattened vector as an input and classifies it in one of the class. They perform weighted aggregation of the descriptors abstracted from the earlier layers. Every

component of the feature from the earlier layer is utilized in the computation of the resultant descriptor. The features from the fully connected layers are complete and hold comprehensive insights. These can be easily exported by performing prediction operations.

3.2.2 YOLO

A novel technique to detect objects, YOLO, was introduced [15]. In contrast to that, here, object detection is framed into bounding boxes that are spatially separated and their class probabilities. Unitary neural networks predict these probabilities and boxes in a single evaluation. End to end optimisation on detection performance is possible as the total pipeline for detection is implemented in one network. The image at the input end is subdivided into an $S \times S$ grid scheme by the system. The grid cell where the object's center falls detects the object in case it falls to a center. Every cell provides predictions of B boxes and their respective confidence scores.

In 2017, the enhanced version, YOLOv2, was released, including batch normalization and anchor boxes. Anchor boxes are preconfigured boundary boxes with specific width and height scales. Anchors, rather than a fully-connected layer, limit the size of objects, but they divorce item classifications from bounding boxes and increase recall [16].

In 2018, a more accurate version of YOLOv3 was presented. Instead of a softmax, sigmoid activation is utilized at the network's output. Because when numerous labels correspond to the same bounding box, softmax performs poorly. A feature pyramid network-like structure is used. Upsampled features are combined with those from earlier layers in the network. Several convolutional layers will be used to process the aggregated characteristics further. Predictions on different scales are more accessible with this structure [17].

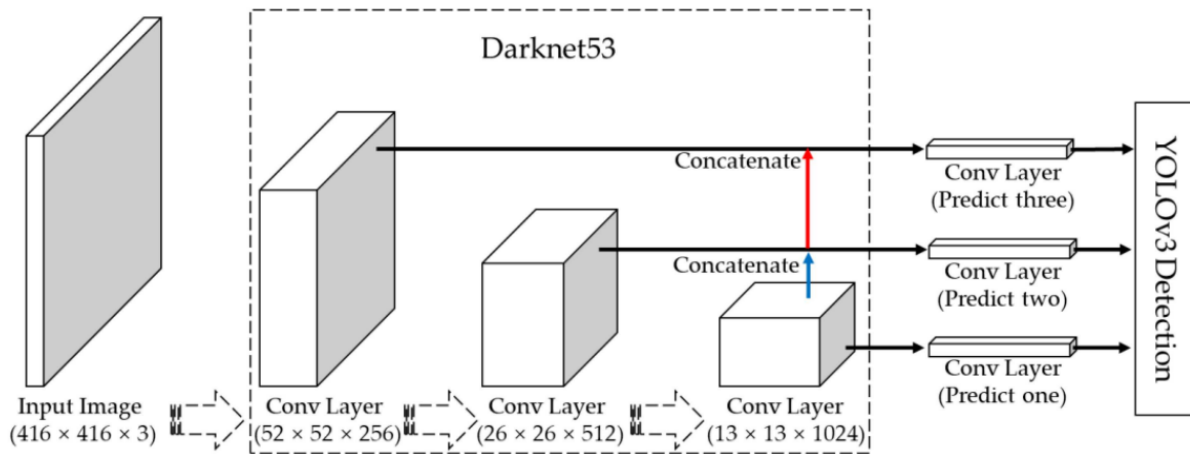


Figure 3 yolo v3 architecture [17]

YOLO is a real-time object identification technique that uses neural networks. Because of its speed and precision, this algorithm is very popular. It has been used to identify traffic signals, pedestrians, parking meters, and animals in a variety of applications.

YOLO algorithm is important because of the following reasons:

- Speed: Because it can predict objects in real-time, this approach enhances detection speed.
- High precision: YOLO is a prediction technique that yields precise findings with low background noise.
- Learning Capabilities: The algorithm has exceptional learning abilities, allowing it to learn object representations and apply them to object detection.

YOLO algorithm works using the following three techniques:

1. Residual blocks
2. Bounding box regression
3. Intersection Over Union (IOU)

1. **Residual blocks:** The image is first separated into several grids. The dimensions of each grid are $S \times S$. The graphic below shows how a grid is created from an input image. There are several grid cells of identical size in the image above. Objects that appear within grid cells will be detected by each grid cell. If an item center emerges within a specific grid cell, for example, that cell will be responsible for detecting it.
2. **Bounding box regression:** A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:
 - Width (bw)
 - Height (bh)
 - Class (person, car, traffic light, etc.). This is represented by the letter c.
 - Class Bounding box center (bx,by)
3. **Intersection over union (IOU):** The concept of intersection over union (IOU) illustrates how boxes overlap in object detection. IOU is used by YOLO to create an output box that properly surrounds the items. The bounding boxes and their confidence scores are predicted by each grid cell. If the anticipated and real bounding boxes are identical, the IOU is 1. This approach removes bounding boxes that aren't the same size as the actual box.

This efficiency makes YOLO well-suited for real-time applications, including our driver monitoring system. By employing YOLO, we can achieve faster and more accurate detection of key facial features and movements associated with driver fatigue, enabling timely intervention to prevent accidents.

Previous works in the field of driver monitoring systems have demonstrated the effectiveness of both CNNs and YOLO in detecting signs of drowsiness and distraction among drivers. However, comparative studies have suggested that YOLO tends to outperform traditional CNN architectures in terms of both speed and accuracy, making it a preferred choice for real-time applications where swift response times are crucial.

Therefore, in our project, we anticipate that leveraging YOLO will yield superior results in detecting and mitigating driver fatigue, ultimately enhancing road safety and reducing the risk of accidents.

By integrating CNNs and YOLO into our driver monitoring system, we aim to harness the capabilities of these advanced deep learning techniques to accurately analyze driver behavior and ensure timely alerts in critical situations.

3.3 Requirement Specification

3.3.1 Functional and Non-Functional Requirements

3.3.1.1 Functional Requirements:

- Real-time Drowsiness Detection: System monitors driver's eyes for drowsiness signs like prolonged closure or yawning.
- Alerting Mechanism: System generates alerts when drowsiness detected, advising driver to refocus or rest.

3.3.1.2 Non-Functional Requirements:

- The system shall be easy to use:
 - Making sure that the system is not hard for people to figure out and use.
- The system shall be reliable and accurate:
 - This means the system should provide correct data.
- The system shall be availability:
 - Means that the system it is available to users when needed.
- The system shall be secure:
 - Means that the system should protect information and keep it safe from unauthorized access or damage.

3.3.2 Hardware Specifications

The main hardware components we used in our project are the Raspberry pi 4 module and the Raspberry pi camera.

Raspberry Pi 4:

The Raspberry Pi 4 is the 4th generation of the mainline series of Raspberry Pi single-board computers. The Raspberry Pi is a debit card-sized low-cost computer that connects to a computer Desktop or TV and uses a standard mouse and Keyboard. It has a dedicated processor, memory, and a graphics driver, just like a PC. It also comes with its operating system, Raspberry Pi OS, a modified version of Linux.



Figure 4 Raspberry Pi 4 module

Raspberry Pi 4 Specifications:

- A high-performance 64-bit quad-core processor
- Dual display support with resolutions up to 4K via a pair of micro-HDMI ports
- Hardware video decoding up to 4Kp60
- 4 GB of RAM
- A connection to the dual-band wireless local area network 2.4/5.0 GHz
- Bluetooth 5.0 / Gigabit Ethernet / USB 3.0 / PoE features (via a separate HAT PoE add-on module).

Raspberry Pi Camera:

The pi Camera module is a camera that can be used to take pictures and high-definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach the PiCamera module directly. This Pi Camera module can attach to the Raspberry Pi's CSI port using a 15-pin ribbon cable.

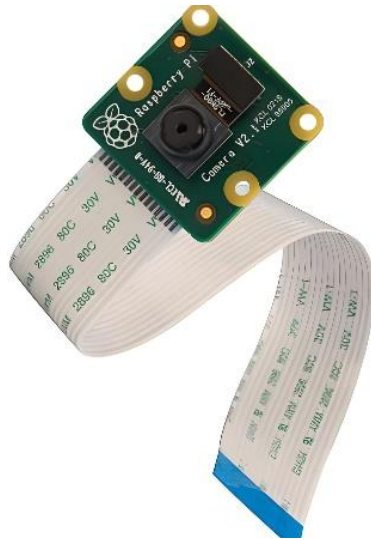


Figure 5 Raspberry Pi Camera

The Raspberry Pi Camera Board Features:

- Fully Compatible with Both the Model A and Model B Raspberry Pi
- 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g
- Fully Compatible with many Raspberry Pi cases

3.3.3 Software Specifications

The following tools and technologies will be utilized in this project:

Python: Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.



Figure 6 Python logo

Flutter: Flutter is Google's UI framework for crafting high-quality native interfaces on iOS, Android, web, and desktop. It will be employed for mobile app development.



Figure 7 Flutter logo

Pandas: Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" It will be utilized for data manipulation and analysis.



Figure 8 Pandas logo

NumPy: NumPy stands for Numerical Python, and SciPy stands for Scientific Python; both are essential Python libraries. These libraries are used to manipulate data in various ways. It will be used for numerical computations and array operations.



Figure 9 NumPy logo

TensorFlow: TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. The open source TensorFlow framework allows you to create highly flexible CNN architectures for computer vision tasks.



Figure10 TensorFlow logo

Using these tools and techniques, the driver's drowsiness detection system will be established and made available as a mobile phone application, ensuring real-time surveillance and warning of drivers to avoid stress-related incidents.

3.3.4 Dataset

For our project we used the "yawn_eye_dataset_new" dataset [18]. It is a collection of 2900 images focused on developing machine learning models for the purpose of detecting driver drowsiness, using features like yawning and eye movements. The dataset is being actively used and explored through various deep learning approaches in the provided Kaggle notebooks.

3.4 System Block Diagrams

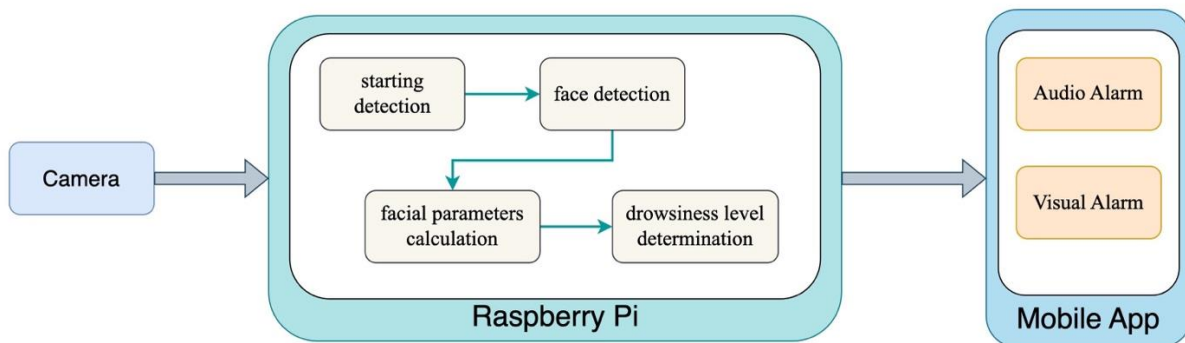


Figure 11: System Block Diagram

As Figure 10 shows, the System Block Diagram explain the connection between all the project parts. First of all, the camera detects the image and send them to raspberry pi. Raspberry pi, intern start the process at the first step is starting detection. After that, we move to face detection, then facial parameter calculation, and the final step is drowsiness level determination.

If raspberry pi detects any situation that led to alert, it will send the information to the mobile app, that in turn will shows a visual alarm in addition to audio alarm.

3.5 Evaluation Metrics:

The performance of our model will be evaluated through the following criteria. Mostly, models are assessed according to precision, recall, and F1-score, in addition to the accuracy.

Precision- It expresses the proportion of the data points our model says was relevant actually were relevant.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Recall- It refers to the percentage of total relevant results correctly classified by our model.

$$\begin{aligned} Recall &= \frac{True\ Positive}{True\ Positive + False\ Negative} \\ &= \frac{True\ Positive}{Total\ Actual\ Positive} \end{aligned}$$

F1_Score- It can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and the worst score at 0.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Chapter 4: Practical Implementation and Results

4.1 System Setup and Architecture

This driver drowsiness detection project employs a Raspberry Pi 4, a Raspberry Pi Camera, and a buzzer as the main hardware components. The system utilizes a Convolutional Neural Network (CNN) to classify driver eye states (open or closed) and detect signs of drowsiness.

4.1.1 Hardware Components

- **Raspberry Pi 4:** Chosen for its affordability and ability to perform real-time image processing, it serves as the central processing unit for running the trained model.
- **Raspberry Pi Camera:** Captures live video of the driver's face, feeding images to the Raspberry Pi for analysis.
- **Buzzer:** Connected to the Raspberry Pi, it emits sound alerts when drowsiness is detected, prompting the driver to regain alertness.

4.1.2 Software Components

This project leverages several software libraries and frameworks to build, train, and deploy the driver drowsiness detection system. Below is a summary of the primary libraries and their functions:

- **Raspberry Pi Operating System (Raspberry Pi OS):** The official operating system for the Raspberry Pi, based on Debian. Raspberry Pi OS is lightweight and optimized for the hardware, ensuring efficient performance. It serves as the foundation for running Python scripts, configuring hardware components, and deploying the trained deep learning model.
- **Python:** The primary programming language used for the project, enabling seamless integration of all required libraries for model training, evaluation, and deployment.
- **TensorFlow & Keras:**
 - **tensorflow:** The core framework for building and deploying deep learning models. It handles model creation, training, and inference.
 - **tensorflow.keras.layers:** Provides layers like Conv2D (convolutional layers), BatchNormalization (for normalizing outputs), Dropout (to prevent overfitting), and Dense (for fully connected layers), essential for constructing the Convolutional Neural Network (CNN) model.
 - **tensorflow.keras.models (Model, Sequential):** Used to define the architecture of the CNN model and layer organization.
 - **tensorflow.keras.optimizers (Adam, SGD):** Optimization algorithms for minimizing loss and improving model performance during training.

- **tensorflow.keras.preprocessing.image (ImageDataGenerator)**: Augments image data in real-time to improve model generalization.
 - **tensorflow.keras.utils (to_categorical)**: Converts categorical labels into a one-hot encoded format for easier processing by the model.
 - **tensorflow.keras.initializers (glorot_uniform)**: Initializes the weights in each layer, aiding model convergence.
 - **tensorflow.keras.regularizers (l2)**: Applies L2 regularization, which helps prevent overfitting by penalizing large weights.
- **MediaPipe**: A cross-platform library used for real-time face and eye tracking in the driver drowsiness detection system. MediaPipe simplifies detection of facial landmarks, such as eyes and mouth, which helps analyze drowsiness indicators like eye closure and yawning in real-time.
- **OpenCV (cv2)**: A comprehensive computer vision library used here for image processing tasks, such as resizing, loading, and manipulating images in real time.
- **NumPy**: Enables efficient handling of arrays and matrices, used extensively to store and process image data.
- **scikit-learn (sklearn)**:
 - **sklearn.metrics (classification_report, confusion_matrix)**: Provides functions to evaluate model performance, generating reports on precision, recall, and F1-score, and creating confusion matrices.
 - **sklearn.model_selection (train_test_split)**: Splits the dataset into training and testing sets, essential for evaluating model accuracy on unseen data.
 - **sklearn.preprocessing (LabelEncoder)**: Encodes categorical labels into numerical values, making them compatible with the model.
- **Matplotlib (matplotlib.pyplot)**: A plotting library used for visualizing training progress (accuracy and loss curves) and displaying sample images from the dataset.
- **Seaborn**: Built on top of Matplotlib, it enhances data visualization capabilities, allowing us to create more informative statistical graphics, such as heatmaps for the confusion matrix.
- **Pandas**: Provides data manipulation tools that help in organizing and analyzing data, particularly useful for handling structured data in preparation for model training.
- **imutils**:
 - **imutils.face_utils (FaceAligner, rect_to_bb)**: Provides face alignment tools, which adjust facial features to a standardized orientation, making it easier to detect eyes and other landmarks consistently.
- **dlib**: A powerful library for machine learning and computer vision, primarily used here for facial landmark detection and alignment, enabling accurate detection of facial features for driver drowsiness detection.

Together, these libraries create a cohesive environment for model training, testing, and deployment, optimizing both performance and usability. This setup ensures that the drowsiness detection system operates efficiently in real-time on the Raspberry Pi platform.

4.1.3 Raspberry Pi Setup and Configuration

Raspberry Pi Overview

The Raspberry Pi 4 serves as the central processing unit in our driver drowsiness detection system. It is a low-cost, credit-card-sized computer equipped with essential features such as a quad-core processor, USB ports, HDMI outputs, and GPIO (General Purpose Input/Output) pins, making it ideal for real-time image processing and deployment of deep learning models.

4.1.3.1 Raspberry Pi Operating System Installation

To set up the Raspberry Pi, a compatible operating system needs to be installed. We used **Raspberry Pi OS (Raspbian)**, a Debian-based operating system designed specifically for Raspberry Pi. Below are the steps to set up and configure the OS:

1. **Download Raspberry Pi Imager:**

- Download the Raspberry Pi Imager tool from the official Raspberry Pi website.



Figure 12 Raspberry Pi Imager

2. Choosing Raspberry Pi Device:

- Raspberry Pi 4, Models B.

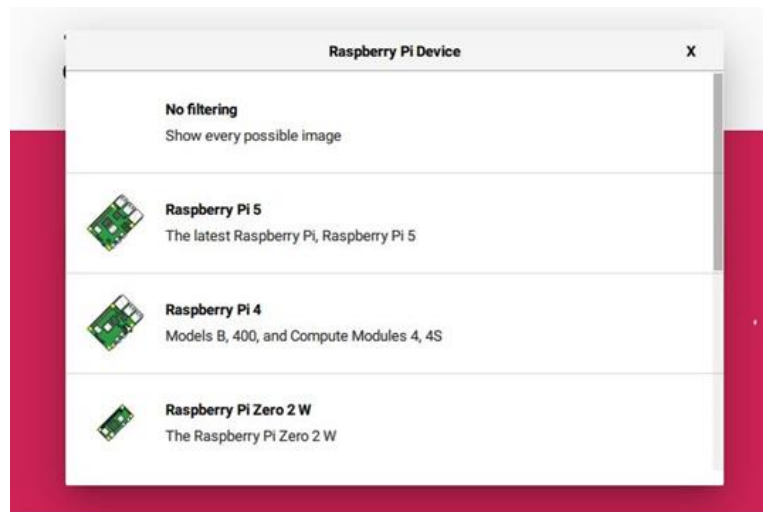


Figure 13 Raspberry Pi Imager (Choosing Device)

3. Choosing Operating System:

- Raspberry Pi OS (64-bit)

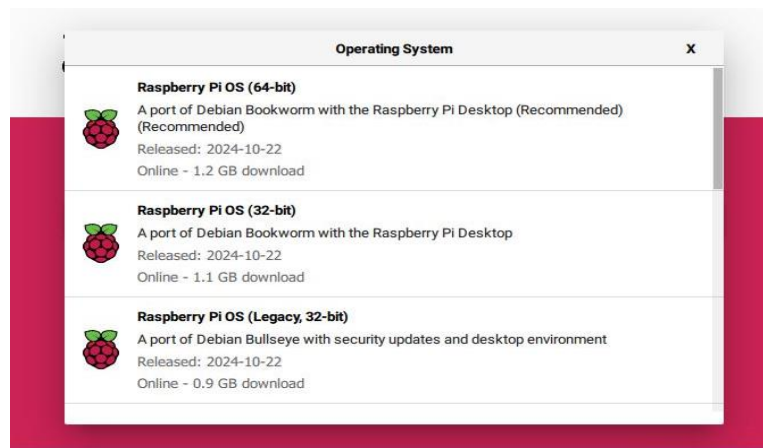


Figure 14 Raspberry Pi Imager (Choosing OS)

4. Prepare the SD Card:

- Insert a microSD card (at least 16GB recommended) into your computer using an SD card adapter.
- Open the Raspberry Pi Imager, select "Raspberry Pi OS" as the OS, and choose the microSD card as the storage medium. Click "Write" to install the OS onto the card.

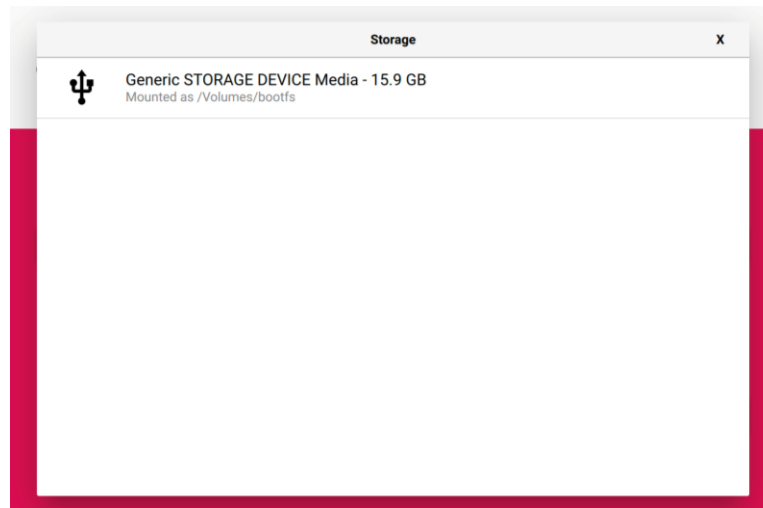


Figure 15 Raspberry Pi Imager (Choosing Storage Device)

5. Initial Configuration:

- Once the OS is installed, insert the microSD card into the Raspberry Pi.
- Connect a monitor, keyboard, and mouse to the Raspberry Pi, and power it on.
- Upon booting, go through the initial setup, including language, time zone, and network configuration.

4.1.3.2 Connecting the Raspberry Pi Camera

The Raspberry Pi Camera Module is connected to the Pi's Camera Serial Interface (CSI) port using a 15-pin ribbon cable. The connection process is straightforward:

1. **Locate the CSI Port:** This port is typically located between the HDMI and audio ports on the Raspberry Pi board.

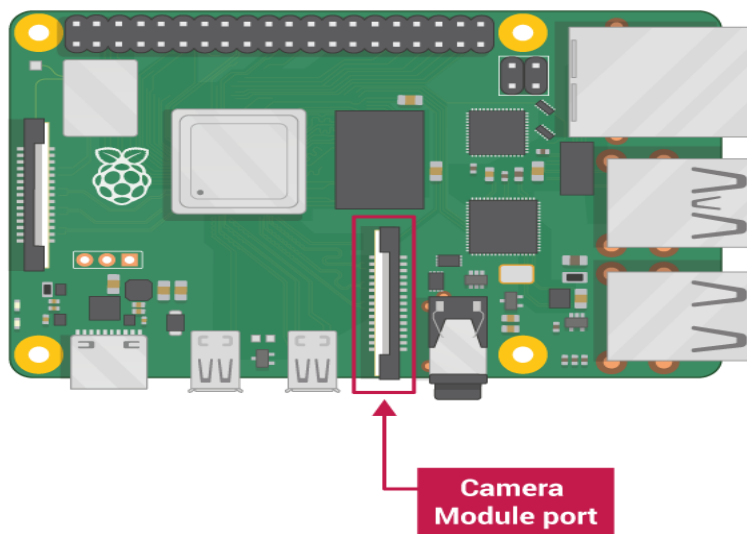


Figure 16 Camera Module Port [19]

2. **Insert the Ribbon Cable:** Gently pull up the black plastic clip on the CSI port, insert the ribbon cable with the metal contacts facing toward the HDMI port, and secure it by pressing down on the clip.
3. **Power On:** Once the camera is securely connected, the Raspberry Pi can be powered on to check the camera connection.
4. **Enabling the Camera Interface:** Open the Raspberry Pi Configuration tool from the main menu, go to the "Interfaces" tab, and enable the "Camera" option. This activates the CSI port, allowing the system to recognize the connected camera.

4.1.3.3 Installing Python on Raspberry Pi

The Raspberry Pi comes with Python pre-installed, as it is the primary language used in many projects on this platform.

4.1.3.4 Installing Necessary Libraries and Dependencies

To prepare the Raspberry Pi for system operation and model deployment, it is essential to install the required libraries and dependencies. These include the tools and software necessary to support real-time image processing, data analysis, and efficient system functionality, ensuring seamless and accurate performance.

4.1.4 Buzzer Connection and Configuration

In this project, the buzzer is used to emit an audible alert when signs of driver drowsiness are detected. Connecting and configuring the buzzer with the Raspberry Pi involves the following steps:

Components Needed

- **Buzzer:** A 5V piezo buzzer, which produces a sound when current flows through it.
- **Jumper Wires:** For connecting the buzzer to the GPIO pins on the Raspberry Pi.

Connecting the Buzzer to the Raspberry Pi

1. **Identify the Buzzer Pins:** Most piezo buzzers have two pins:
 - **Positive (+):** Connects to the power source.
 - **Negative (-):** Connects to the ground (GND).
2. **Wiring the Buzzer:**
 - **Positive Pin:** Connect the positive (+) pin of the buzzer to a GPIO pin on the Raspberry Pi (for example, GPIO18).

- **Negative Pin:** Connect the negative (-) pin of the buzzer to one of the GND (ground) pins on the Raspberry Pi.

GPIO Pins of Raspberry Pi 4

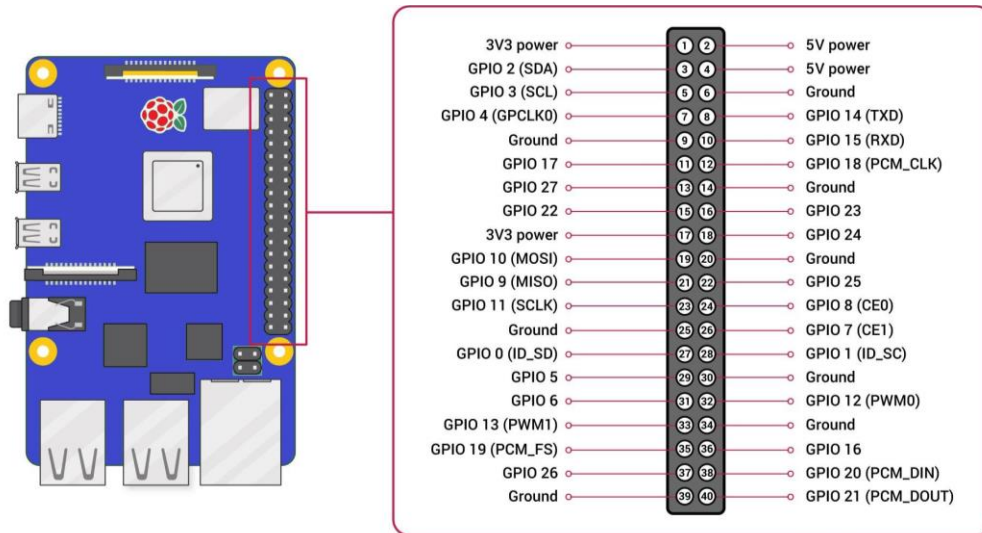


Figure 17 40 GPIO Pins Description of Raspberry Pi 4 [20]

Setting Up GPIO Control in Python

To control the buzzer using Python, follow these steps:

1. **Install the RPi.GPIO Library:**

```
sudo apt-get install python3-rpi.gpio
```

4.2 Model Development and Training

4.2.1 Dataset

The "[Drowsiness Dataset](#)" from Kaggle contains images of eyes in open and closed states, labeled to support classification. The dataset is preprocessed, with each image resized to 32×32 pixels, and labeled as either "open" or "closed".

Example Images

Below are examples from the dataset for each class:

1. Open Eye:

- These images show the driver's eyes in an open state, which the model learns to recognize as a non-drowsy condition.
 - **Example 1:** Image of an eye fully open with visible iris and pupil.



Figure 18 DataSet Example: Open Eye

2. Closed Eye:

- These images display the driver's eyes in a closed state, indicating potential drowsiness.
 - **Example 1:** Eye completely closed, with no visible iris or pupil.



Figure 19 DataSet Example: Closed Eye

4.2.2 Data Distribution Visualization

To understand the balance between classes, we created bar charts showing the distribution of images in each category for both the training and testing sets.

- **Training Set Distribution:** The chart below shows the number of images in each category (Open and Closed) for the training set. This visualization helps assess whether the model has sufficient data to learn both classes effectively.

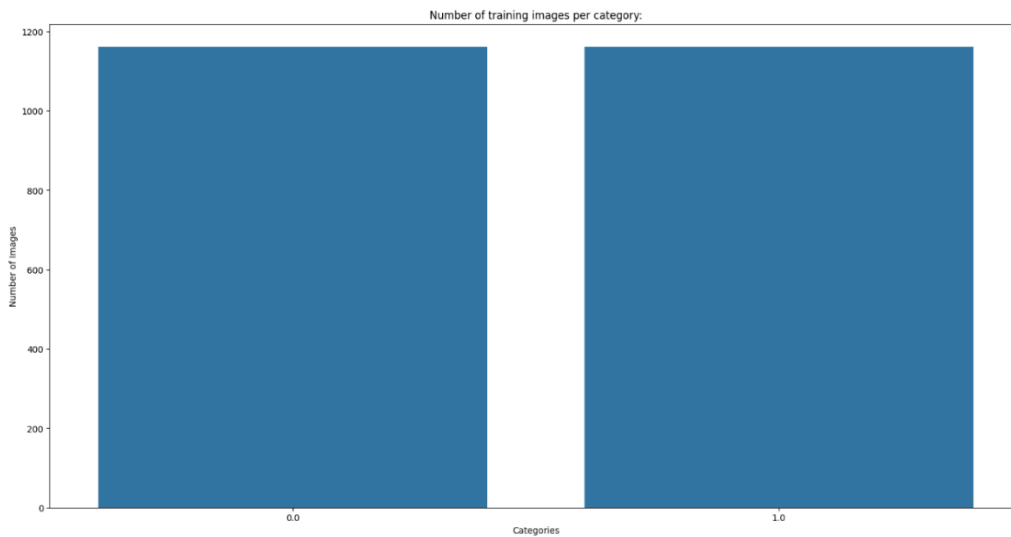


Figure 20 Training Set Distribution

- **Testing Set Distribution:** Similarly, the chart below illustrates the number of images in each category for the testing set, allowing us to ensure a balanced evaluation.

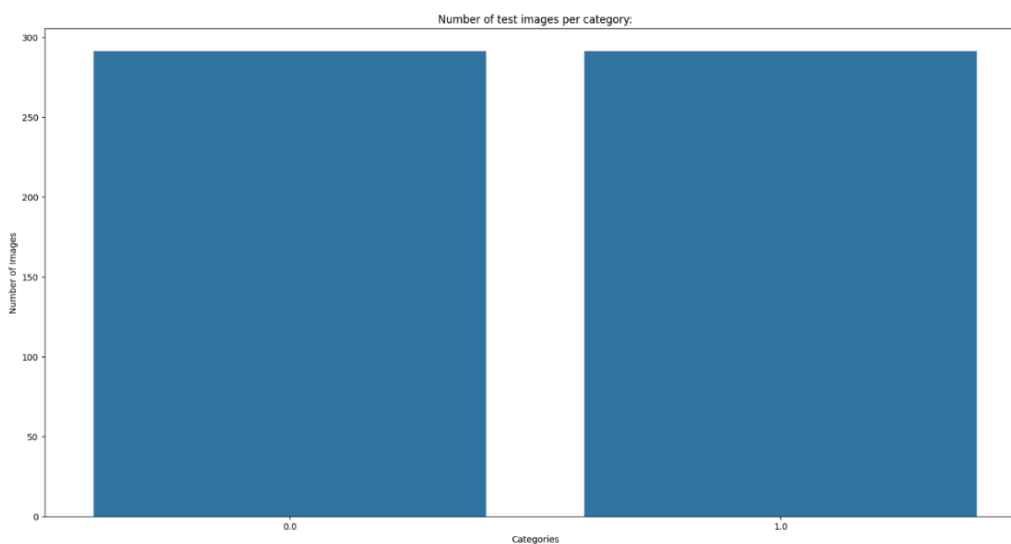


Figure 21 Test Set Distribution

4.2.3 Data Preprocessing

- **Image Resizing:** All images were resized to $32 \times 32 \times 3$ to standardize input size for the model.
- **Label Encoding:** Labels were encoded into binary format, with "0" representing closed eyes and "1" for open eyes.
- **Data Augmentation:** To increase model robustness, data augmentation techniques such as rotation, zoom, and horizontal flips were applied.

4.2.4 Model Architecture

A custom CNN model was designed to classify open and closed eye states. Key layers include:

- **Convolutional Layers:** Extract features such as edges, textures, and patterns.
- **Batch Normalization:** Accelerates training and reduces overfitting.
- **Max Pooling:** Reduces spatial dimensions to focus on the most salient features.
- **Fully Connected Layers:** After flattening, fully connected layers learn high-level patterns for classification.

The model was trained using the Adam optimizer with a learning rate of 0.0001, optimizing for categorical cross-entropy loss.

The table below summarizes the model architecture:

Table 3 Model Architecture

Layer (type)	Output Shape	Param #
Conv1 (Conv2D)	(None, 32, 32, 32)	896
batch_normalization_14 (BatchNormalization)	(None, 32, 32, 32)	128
Conv2 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_15 (BatchNormalization)	(None, 32, 32, 32)	128
dropout_10 (Dropout)	(None, 32, 32, 32)	0
max_pooling2d_8 (MaxPooling2D)	(None, 16, 16, 32)	0
Conv3 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_16 (BatchNormalization)	(None, 16, 16, 64)	256

max_pooling2d_9 (MaxPooling2D)	(None, 8, 8, 64)	0
Conv4 (Conv2D)	(None, 8, 8, 64)	36928
batch_normalization_17 (BatchNormalization)	(None, 8, 8, 64)	256
dropout_11 (Dropout)	(None, 8, 8, 64)	0
max_pooling2d_10 (MaxPooling2D)	(None, 4, 4, 64)	0
Conv5 (Conv2D)	(None, 4, 4, 64)	36928
batch_normalization_18 (BatchNormalization)	(None, 4, 4, 64)	256
Conv6 (Conv2D)	(None, 4, 4, 64)	36928
batch_normalization_19 (BatchNormalization)	(None, 4, 4, 64)	256
conv7 (Conv2D)	(None, 4, 4, 64)	36928
batch_normalization_20 (BatchNormalization)	(None, 4, 4, 64)	256
dropout_12 (Dropout)	(None, 4, 4, 64)	0
max_pooling2d_11 (MaxPooling2D)	(None, 2, 2, 64)	0
flatten_2 (Flatten)	(None, 256)	0
fc1 (Dense)	(None, 128)	36896
dropout_13 (Dropout)	(None, 128)	0
fc2 (Dense)	(None, 128)	16512
dropout_14 (Dropout)	(None, 128)	0
fc3 (Dense)	(None, 2)	258

This comprehensive architecture ensures that the model can effectively extract patterns and classify images with high accuracy.

4.2.5 Training Process and Performance Metrics

The model was trained for 200 epochs using an 80-20 split for training and validation. The following metrics were used to evaluate performance:

- **Accuracy:** Monitored during training and validation.
- **Loss:** Observed to ensure convergence and avoid overfitting.

Training Results

Graphs were plotted to visualize the changes in accuracy and loss over the 200 training epochs:

1. Accuracy over epochs:

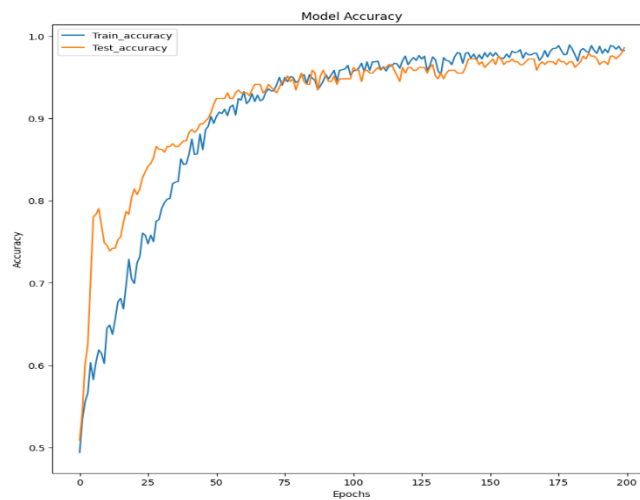


Figure 22 Accuracy over epochs

2. Loss over epochs:

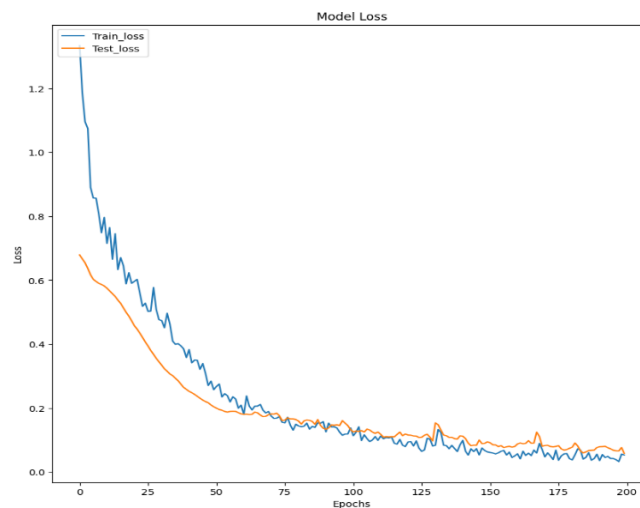


Figure 23 Loss over epochs

4.2.6 Evaluation and Validation

After training, the model was evaluated on the test set to measure its performance. The evaluation results showed that the model achieved a test accuracy of approximately **99%**, demonstrating its ability to classify open and closed eye states effectively. Additionally, the test loss was recorded at **0.06**, indicating a low level of prediction error.

4.2.6.1 Classification Report

A detailed classification report was generated to evaluate the model's performance for each class (open and closed eyes). Key metrics included:

- **Precision:** The proportion of correctly predicted positive samples to the total predicted positives.
- **Recall:** The proportion of correctly predicted positive samples to the total actual positives.
- **F1-Score:** A weighted average of precision and recall, providing a balanced measure of the model's performance.

The classification report is as follows:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	142
1	0.99	0.99	0.99	149
accuracy			0.99	291
macro avg	0.99	0.99	0.99	291
weighted avg	0.99	0.99	0.99	291

Figure 24 Classification Report

4.2.6.2 Confusion Matrix

The confusion matrix provides a visual representation of the model's performance, displaying the number of true positives, true negatives, false positives, and false negatives for each class. It helps assess the model's ability to distinguish between open and closed eye states.

Below is the confusion matrix:

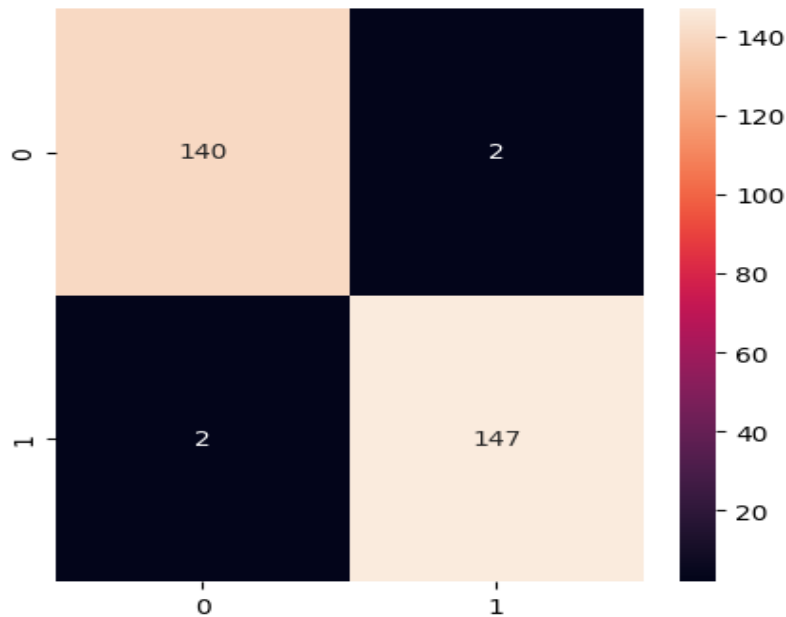


Figure 25 Confusion Matrix

4.3 Model Deployment on Raspberry Pi

4.3.1 Alert System Modification

Initially, the system design included a mobile app to provide an audible alert when drowsiness was detected. However, this approach was replaced with a hardware-based buzzer connected directly to the Raspberry Pi. This decision was made for several reasons:

1. **Reliability:** Using a dedicated buzzer ensures that the alert will sound regardless of mobile connectivity or app performance, reducing potential failure points in alert delivery.
2. **Real-Time Response:** The buzzer offers immediate, direct feedback to the driver without requiring smartphone interaction, which is essential for rapid response in a drowsiness detection system.
3. **Simplified Implementation:** Integrating the buzzer with the Raspberry Pi streamlined the setup, making the overall system simpler and more effective, particularly in real-time testing environments.

This change ensures that the alert system functions reliably and promptly, enhancing the project's primary goal of improving driver safety.

4.3.2 Real-Time Driver Drowsiness Detection

The real-time implementation of the driver drowsiness detection system integrates multiple indicators (eye state, yawning, and head tilt) to assess the driver's overall state. The system processes live video from a camera and outputs a final result: "**Drowsy**" or "**Not Drowsy**". Below are the main functionalities of the system:

4.3.2.1 Combined Drowsiness Detection

The system determines the driver's state based on a combination of the following:

1. **Eye State:** The trained CNN model predicts whether the driver's eyes are open or closed.
2. **Yawning:** Yawning is detected by measuring the vertical distance between the upper and lower lips using facial landmarks from **MediaPipe**.
3. **Head Tilt:** The system detects head tilts (e.g., leaning to the left or right) using angles calculated from eye landmarks.

A predefined logic integrates these factors:

- The driver is classified as "**Drowsy**" if:
 - Eyes are closed for a prolonged period.
 - Yawning is detected.
 - Significant head tilt indicates a loss of focus.
- Otherwise, the driver is classified as "**Not Drowsy**."

4.3.2.2 Real-Time Video Processing

- The camera captures frames in real-time, and **MediaPipe** detects facial landmarks.
- Eye regions are cropped and preprocessed before being passed to the CNN model for eye state prediction.
- Yawning and head tilt are analyzed directly from facial landmarks.
- The final result, "**Drowsy**" or "**Not Drowsy**," is displayed on the video feed.



4.3.2.3 Alert System

When the system detects a "**Drowsy**" state, a buzzer connected to the Raspberry Pi is activated to alert the driver immediately. This ensures a timely response to prevent accidents caused by drowsiness.

4.4 Results

4.4.1 Model Accuracy

The final model achieved an accuracy of **99%** on the test set, as detailed in the classification report. Precision, recall, and F1-scores showed that the model accurately differentiated between open and closed eye states.

4.4.2 Confusion Matrix Analysis

A confusion matrix was generated to visualize classification performance. The model displayed a high true positive rate for both open and closed eye states, with low false positives and negatives, indicating reliable detection.

4.4.3 Practical Performance on Raspberry Pi

The model's performance was tested in various real-world conditions:

- **Lighting Variability:** The system performed well in different lighting conditions after data augmentation.
- **Processing Speed:** The model achieved frame processing rates sufficient for real-time detection, although adjustments were required to optimize for Raspberry Pi's hardware.
- **False Positives:** Occasionally, normal blinks were detected as closed eyes; however, tuning the alert threshold minimized these instances.

4.5 Model Testing with Sample Scenarios

To evaluate the model's performance, it was tested with various sample images and simulated scenarios representing drowsy and non-drowsy states. The goal was to verify the system's ability to make correct overall classifications.

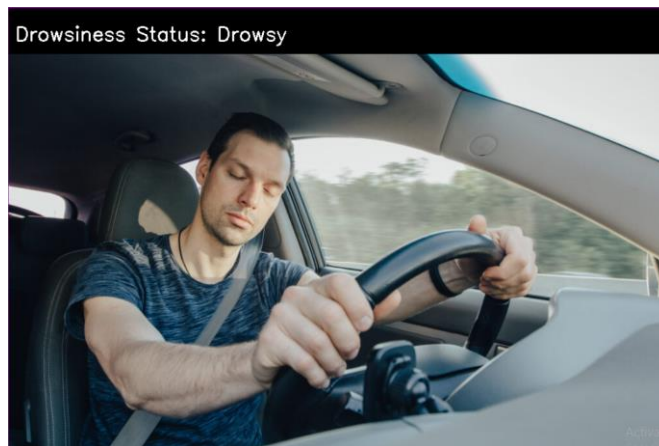
4.5.1 Testing Scenarios

1. **Non-Drowsy State:**
 - **Condition:** Eyes open, no yawning, head straight.
 - **Result:** "Not Drowsy."



2. Drowsy State:

- **Condition:** Eyes closed, yawning detected, or significant head tilt.
- **Result:** "Drowsy."



4.5.2 Accuracy of the System

The system showed high reliability in detecting drowsy states under controlled conditions. It successfully integrated multiple indicators to make accurate classifications of "**Drowsy**" and "**Not Drowsy**."

4.6 Suitability of the Model for Women Wearing Niqab and Drivers Wearing Glasses

Our model has shown robust performance in detecting signs of driver drowsiness, even when the driver is wearing a niqab (face covering), where only the eye area is visible. The system uses advanced eye-tracking technology, which primarily focuses on eye movements, such as blinking and yawning, as key indicators of drowsiness. Despite the facial obscuration caused by the niqab, the model effectively identifies these eye-related cues, enabling it to accurately monitor the fatigue state of female drivers who wear a niqab.

To address these unique scenarios, the model was intentionally designed to focus on the state of the eyes (open or closed) as the primary indicator of drowsiness. This approach ensures reliability in cases where facial features are partially or entirely obscured. Since the niqab prevents the system from identifying facial features like the mouth or nose, additional indicators such as yawning and head tilt were analyzed using other methods and libraries, such as MediaPipe. This complementary integration of eye-focused detection with advanced landmark analysis enables the system to maintain high accuracy, even in challenging conditions.



Furthermore, the model performs with high precision for drivers wearing prescription glasses. Glasses can sometimes obscure facial clarity, but by concentrating on eye-state detection, the model effectively monitors drowsiness without being affected by visual obstructions. This multi-faceted approach ensures robust detection across diverse real-world scenarios.



4.7 Limitations

4.7.1 Limitations

- **Environmental Sensitivity:** Changes in lighting and angles impacted detection accuracy.
- **Processing Constraints:** The Raspberry Pi struggles with frame rates under certain conditions, limiting detection speed.

Chapter 5: Conclusion and Future Work

The development of an innovative artificial intelligence model designed to effectively monitor and detect drivers' fatigue through the analysis of eye movements, with a specific focus on eye as a fatigue indicator, represents a pivotal step towards enhancing road safety and mitigating the prevalence of accidents stemming from driver fatigue-related incidents. By integrating this advanced model into vehicles as a surveillance system capable of identifying signs of fatigue such as eye drooping and frequent yawning, the project endeavors to proactively address the issue of reckless driving behaviors, thereby fostering a safer driving environment for all road users.

The project successfully developed an artificial intelligence model for detecting driver fatigue through eye movements, yawning, and head tilt. The system, to be implemented in vehicles, aims to enhance road safety by monitoring drivers in real-time and sending alerts upon detecting signs of drowsiness. By utilizing technologies like CNN, the project addresses functional requirements such as real-time drowsiness detection and alerting mechanisms, while also considering non-functional requirements like ease of use, reliability, and accuracy. The hardware components include Raspberry Pi 4, Raspberry Pi Camera, and Buzzer. with software components like Python, TensorFlow, Pandas, and NumPy. The project not only contributes to reducing accidents but also raises awareness about responsible driving habits. Overall, the project aligns with engineering standards and provides a comprehensive solution for driver drowsiness detection in a cost-effective and ethical manner.

References

- [1] V. Ramalingam, S. Sheth and A. Singhal, "Driver Drowsiness Detection System using Machine Learning Algorithms," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 990-993, 2020.
- [2] D. & Diaz, "CAR ACCIDENT STATISTICS YOU NEED TO KNOW IN 2024," Darrigo & Diaz, 2024. [Online]. Available: <https://www.ddlawtampa.com/resources/car-accident-statistics-you-need-to-know-in-2021/>.
- [3] P. Deepthi M., C. Binu P. and B. K.P. Mohamed, "Driver distraction detection using machine learning techniques," *Materialstoday: Proceedings*, vol. 58, no. 1, pp. 251-255, 2021.
- [4] D. Md. Tanvir Ahammed, H. Syeda Sumbul, A. Yeasir and B. R. Fatama, "Real-time driver drowsiness detection using deep learning," (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 7, pp. 3675-3685, 2021.
- [5] H. Maryam, M. Alireza and S. Aliasghar Beheshti, "Driver Safety Development: Real-Time Driver Drowsiness Detection System Based on Convolutional Neural Network," *arXiv*, vol. 22, no. 3, pp. 1567-1577, 2021.
- [6] K. A. Al Shalfan and M. Zakariah, "Detecting driver distraction using deep-learning approach," *Computers, Materials & Continua*, vol. 68, no. 1, p. 689, 2021.
- [7] M. Ramzan, H. Khan, S. Awan, A. Ismail, M. Ilyas and A. Mahmood, "A survey on state-of-the-art drowsiness detection techniques.," *IEEE Access*, p. 61904–61919, 2019.
- [8] V. Harshit, K. Amit, M. Gouri Shankar, d. Ujjwal, M. Pradeep Kumar and N. Parma, "DRIVER DROWSINESS DETECTION," *Journal of Data Acquisition and Processing*, 2023.
- [9] I. B. A. Mohammed, A. Halah, A. Fatimah, A. Dana, A. Manar, A. Munira, A. Shoog, R. Atta, Y. Mustafa and Z. Gohar, "A Deep-Learning Approach to Driver Drowsiness Detection," *Safety*, vol. 9, no. 65, 2023.
- [10] H. Ahuja, S. Saurav, S. Srivastava and C. Shekhar, "Driver Drowsiness Detection using Knowledge Distillation Technique for Real Time Scenarios," *2020 IEEE 17th India Council International Conference (INDICON)*, pp. 1-5, 2020 .
- [11] A. I. Chowdhury, A. R. Niloy and N. Sharmin, "A deep learning based approach for real-time driver drowsiness detection," *2021 5th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pp. 1-5, 2021.
- [12] A. Dhillon and G. K Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection.," *Progress in Artificial Intelligence*, vol. 9, no. 2, p. 85–112, 2020.
- [13] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint*, 2015.
- [14] D. d. R. a. H. H. Michael Egmont-Petersen, "Image processing with neural networks—a review," *Pattern recognition*, vol. 35, no. 10, p. 2279–2301, 2002.
- [15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement [Internet].," *arXiv*, 2016.

- [16] X. Zhu, S. Lyu, X. Wang and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," *arXiv*, 2021.
- [17] F. Dridi, S. El Assad, Y. W. El Hadj, M. Machhout and R. Lozi, "The design and FPGA-based implementation of a stream cipher based on a secure chaotic generator," *Appl Sci*, vol. 11, no. 2, p. 625, 2021.
- [18] S. RAJU, "yawn_eye_dataset_new," Kaggle, 2020. [Online]. Available: <https://www.kaggle.com/datasets/serenaraju/yawn-eye-dataset-new/code>. [Accessed April 2024].
- [19] r. foundation. [Online]. Available: <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/1>.
- [20] A. Singh, "Raspberry Pi 4 Pin Diagram Specifications and Application," 12 11 2021. [Online]. Available: <https://hackatronic.com/raspberry-pi-4-specifications-pin-diagram-and-description/>.