الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Faculty of Artificial Intelligence
Department of Artificial Intelligence and Data Science

JUNIOR PROJECT REPORT ON

A SYSTEM FOR DETECTING FATIGUE THROUGH FACIAL IMAGES OF DRIVERS

Prepared by:

Omar Alhariri – 4210353

Alaa edden Zarzor – 4210193

Supervised by:

Dr. Kadan Aljoumaa

Abstract

Drowsy driving is a major contributor to road accidents and fatalities worldwide, creating an urgent need for reliable driver monitoring systems. This research presents a real-time, non-intrusive driver drowsiness detection system leveraging convolutional neural networks (CNNs) and eye-region analysis.

Two CNN architectures were trained and evaluated on the MRL Eye Dataset, a collection of labeled eye images. The best-performing model (V2) achieved a test accuracy of 99.14%, with precision 99.14%, recall 99.12%, and F1-score 99.13%, demonstrating high reliability, particularly in detecting drowsy states — a critical aspect for safety-critical applications.

Real-time predictions were implemented using OpenCV and MediaPipe for live video streams. All experiments and model versions were systematically tracked and logged using Weights & Biases to ensure reproducibility.

The proposed system outperformed previously reported results on the same dataset, demonstrating the effectiveness of CNN-based modeling combined with rigorous experimentation and monitoring. This framework highlights the potential for deploying high-performance, deep learning–driven driver monitoring systems in real-world scenarios.

Table of Content

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Full Term / Description
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the Curve
CEW	Closed Eyes in the Wild (Dataset)
CNN	Convolutional Neural Network
DDD	Driver Drowsiness Detection
DROZY	(A multimodal drowsiness dataset)
ECG	Electrocardiogram
EEG	Electroencephalogram
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GA	Genetic Algorithm
GPU	Graphics Processing Unit
IR	Infrared
JSON	JavaScript Object Notation
KNN	k-Nearest Neighbors
L2	L2 Regularisation (Weight Regularisation)
LR	Learning Rate
mAP	Mean Average Precision
ML	Machine Learning

MLOps	Machine Learning Operations
MRL	Media Research Lab (Dataset)
NTHU-DDD	National Tsing Hua University Driver Drowsiness Detection
ReLU	Rectified Linear Unit
ResNet50V2	Residual Network 50-layer Version 2
RGB	Red, Green, Blue (Colour model)
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UI	User Interface
UTA-RLDD	University of Texas at Arlington Real-Life Drowsiness Dataset
VGG19	Visual Geometry Group 19-layer network
ViT	Vision Transformer
W&B	Weights & Biases (Experiment tracking platform)
YawDD	Yawning Detection Dataset
YOLO	You Only Look Once (Object detection model)

Chapter 1: Introduction

1.1 Background and Motivation

Driver fatigue is a significant contributor to road accidents worldwide, leading to thousands of fatalities and injuries annually. According to studies, drowsy driving is responsible for approximately 20% of all road crashes in some regions, highlighting the urgent need for effective monitoring systems. Existing methods for detecting driver fatigue include physiological monitoring, vehicle-based metrics, and video-based facial analysis; however, many of these approaches suffer from limitations such as intrusiveness, high cost, or delayed detection.

The motivation behind this research is to develop a real-time, non-intrusive driver drowsiness detection system that addresses these challenges. The major reasons for pursuing this approach include:

- Safety: Monitoring drivers in real-time can help avert collisions by alerting them to their impaired state and enabling corrective actions.
- Reducing Accidents: Early identification of drowsiness can significantly reduce the number of fatigue-related accidents.
- Improved Productivity: By preserving driver alertness and concentration, such systems can minimize accidents and enhance overall efficiency.
- Cost Savings: Non-intrusive detection systems can save money for both individuals and businesses by preventing accidents and associated losses.

This study leverages eye image analysis and convolutional neural networks (CNNs) to create a system capable of accurate, fast detection, opening the way for safer and more efficient road transportation.

1.2 Research Problem Statement

Despite extensive research on driver drowsiness detection, several challenges remain unresolved. Existing approaches often suffer from limited generalization due to constrained datasets, sensitivity to variations in lighting conditions and camera viewpoints, and trade-offs between real-time performance and detection accuracy. Additionally, many studies lack systematic experiment tracking and reproducibility, which hinders fair comparison and reliable deployment in real-world scenarios.

Therefore, the core research problem addressed in this work is the need for a robust, high-accuracy, and real-time drowsiness detection framework that can operate in a non-intrusive manner while maintaining reproducibility and practical feasibility. Specifically, this research investigates how CNN-based models can be effectively designed, trained, and deployed to reliably distinguish between alert and drowsy eye states under realistic operating conditions.

1.3 Research Objectives

The primary objective of this research is to develop and evaluate a real-time, non-intrusive driver drowsiness detection system based on computer vision and deep learning techniques, with the goal of improving road safety.

To achieve this primary objective, the research pursues the following objectives:

1. collect and preprocess driver facial images and accurately extract eye regions using computer vision techniques.
2. To apply data preprocessing strategies, including normalization and data augmentation, in order to improve model generalization and robustness during training.

3. To design and train convolutional CNN models for binary classification of eye states into open and closed classes using a labeled eye image dataset.
4. To evaluate and compare the performance of multiple CNN architectures using standard evaluation metrics such as accuracy, precision, recall, and F1-score, and to identify the most effective model.
5. To implement and integrate the selected model into a real-time inference pipeline capable of processing live video streams with reliable prediction performance.

1.4 Contributions and Key Results

This research includes a number of tangible research and practical contributions, which can be summarized as follows:

- Proposing a comprehensive and practical framework for driver drowsiness detection based on eye-region analysis and convolutional neural network (CNN) models specifically designed for the task.
- Training and evaluating two CNN architectures on the MRL Eye Dataset, where the best-performing model (V2) achieved high performance on the test set:

  Accuracy: 99.14% Precision: 99.14%

  Recall: 99.12% F1-score: 99.13%
- Implementing real-time prediction using OpenCV and MediaPipe to process live video streams and extract the eye-region pipeline without the need for embedded devices or additional sensors.

- Adopting MLOps practices by tracking all experiments and model versions using Weights & Biases, ensuring result reproducibility and effective management of model scripts and artifacts.
- Enhancing deployability by designing backend specifications using FastAPI and a lightweight user interface built with Streamlit to visualize real-time predictions and enable integration into larger systems.

1.5 Research Structure

Chapter 1 introduces the research topic, presenting the background and motivation, the research problem, the research objectives, and the main contributions of the study

Chapter 2 presents the background and related work, including fundamental concepts of driver drowsiness detection, a review of existing approaches, key challenges, and commonly used evaluation metrics.

Chapter 3 details the proposed methodology, covering dataset description, preprocessing steps, CNN model architectures, experimental setup, performance evaluation, and experiment tracking practices.

Chapter 4 describes the system implementation and deployment aspects, including backend development using FastAPI, model serving, API design, and the Streamlit-based user interface for real-time prediction.

Chapter 5 discusses the experimental results, highlighting the strengths and limitations of the proposed system and comparing it with existing methods.

Chapter 6 concludes the report by summarizing the main findings and outlining potential directions for future research.

Chapter 2: Background and Related Work

2.1 Introduction

The development of an effective (DDD) system requires a solid foundation in both human behavior analysis and computer vision. Reviewing existing research is essential to identify the most reliable indicators of fatigue, evaluate state-of-the-art architectures (such as CNNs and Transformers), and pinpoint current technical limitations.

This chapter provides a comprehensive overview of the field. It covers the fundamentals of drowsiness detection, followed by a literature review of recent studies and their methodologies. It also discusses the challenges of real-world implementation, such as lighting and real-time constraints, and defines the evaluation metrics used to measure model performance. Ultimately, this chapter highlights the research gaps that this project aims to address.

2.2 Fundamentals of Driver Drowsiness Detection

Drowsiness is a transitional state between wakefulness and sleep, characterized by impaired cognitive functions, slower reaction times, and decreased alertness. In the context of driving, identifying this state early is vital for safety. Detection methods generally rely on **Visual Indicators**, which are the physical manifestations of fatigue. These include **Eye Closure** (measured by duration and frequency), **Yawning** (indicated by specific mouth shapes), and **Head Pose** (such as nodding or drooping).

Detection technologies are broadly categorized into two main approaches:

Intrusive Methods: These require the driver to wear physical sensors to measure internal signals. While highly accurate, they often cause discomfort or distraction during long-term use.

Non-Intrusive Methods: These use remote sensors, primarily cameras or vehicle sensors, to monitor the driver or the car's behavior without any physical contact, offering a more comfortable user experience.

the following table summarizes the key characteristics, advantages, and limitations of each measurement type:

Type	Category	Definition	Advantages	Limitations
Biological	Intrusive	Monitors internal physiological signals (EEG, ECG).	High precision; direct brain-state monitoring.	Requires wearable sensors; causes discomfort.
Image/ Video	Non-intrusive	Analyzes facial features via camera input.	Cost-effective; compatible with Deep Learning.	Lighting sensitive; high computational cost.
Vehicle-based	Non-intrusive	Tracks driving patterns and vehicle dynamics.	Zero driver contact; utilizes existing hardware.	Indirect measure; affected by road/driving style.
Hybrid	Mixed	Integrates two or more detection methods.	High reliability; complements individual strengths.	Complex integration; increased system cost.

Table 1 Overview of driver drowsiness detection measurement types

This project adopts a non-intrusive approach using Convolutional Neural Networks (CNNs) to analyze eye states, as it offers a balance between accuracy and user comfort.

2.3 Related Studies and Existing Approaches

from traditional machine learning to advanced deep learning architectures. Reviewing recent literature is crucial to understanding the performance benchmarks and technical gaps in current systems. This section analyzes six recent studies published between 2024 and 2025, focusing on their methodologies, datasets, and accuracy outcomes.

2.3.1 Transformer-Based and Multi-Model Approaches

Recent research has shifted towards attention-based mechanisms. For instance, the study in [1] explored the efficacy of Vision Transformers (ViT) and Swin Transformers against traditional transfer learning models like VGG19 and ResNet50V2. Utilizing a combination of MRL, NTHU-DDD, and CEW datasets, the ViT model achieved a superior accuracy of 99.15%, demonstrating the power of global feature extraction in facial analysis.

Similarly, the work presented in [2] conducted a comparative study between classical classifiers such as KNN and SVM, and modern object detection models like YOLOv5 and YOLOv8. While the KNN classifier reached 98.89% accuracy, the YOLO series showed exceptional precision (100%) on the UTA-RLDD dataset, highlighting their suitability for real-time localization.

2.3.2 Lightweight and Optimized Architectures

Efficiency is a key requirement for in-vehicle systems to ensure low latency. The research in [3] proposed DrowsyDetectNet, a lightweight, shallow CNN architecture designed for limited training data. It outperformed deeper models like InceptionV3 with an accuracy of 99.23%, proving that specialized shallow networks can be more effective for eye-state classification.

In another approach to optimization, the study in [5] focused on CNN architecture optimization using Genetic Algorithms (GA). By using GA to evolve the CNN structure on the CEW dataset, the researchers achieved 91.8% accuracy. While this accuracy is lower than some fixed architectures, it highlights the potential of automated model design for specific datasets.

2.3.3 Multimodal and Commercial Monitoring Systems

Beyond visual-only data, multimodal approaches are gaining traction to increase reliability. The study in [4] introduced multimodal neural networks leveraging the DROZY dataset, achieving up to 98.41% accuracy by coupling different feature sets.

Finally, for commercial applications, the evaluation in [6] compared various object detection models on a self-prepared dataset. Among the tested models, YOLOv5 emerged as the most practical choice for real-time deployment, balancing a mAP of 93.6% with a high processing speed of 125 FPS.

2.3.4 Summary of Related Studies

To provide a clear comparison of the discussed literature, Table 2 summarizes the key components of each study.

Ref.	year	Core Methodology	Datasets Used	Best Metric
[1]	2025	ViT, Swin Transformer, CNNs	MRL, NTHU-DDD, CEW	99.15% (ViT)
[2]	2025	KNN, SVM, YOLOv5/v8	NTHUDDD, YawDD, UTA-RLDD	100% Precision (YOLO)
[3]	2025	Lightweight Shallow CNN	Dataset-1 & Dataset-2 (Self-prepared)	99.23% Accuracy
[4]	2025	Multimodal Feature Fusion	DROZY	98.41% Accuracy
[5]	2025	CNN + Genetic Algorithm	CEW	91.8% Accuracy
[6]	2025	YOLOv5, Faster R-CNN	Self-prepared	125 FPS / 93.6% mAP

Table 2 Comparative Summary of Recent Drowsiness Detection Literature

2.4 Characteristics, Challenges, and Requirements

This section synthesizes the main findings derived from the reviewed literature, highlighting common system characteristics, recurring challenges, and key requirements for practical driver drowsiness detection systems.

2.4.1 Common Challenges

A number of challenges consistently appear across existing studies. One of the most significant issues is sensitivity to environmental conditions, particularly variations in lighting, which strongly affect face and eye detection accuracy. Changes in head pose, facial orientation, and partial occlusions caused by hands, glasses, or hair further degrade system performance. In addition, substantial inter-subject variability—such as differences in facial structure, eye shape, and blinking patterns—limits the generalization ability of many models. Finally, several studies report reliance on small or imbalanced datasets captured in controlled environments, increasing the risk of overfitting and reducing real-world reliability.

Overall, these challenges indicate that many proposed methods perform well under constrained conditions but struggle in realistic driving scenarios.

2.4.2 Dataset Variability

Dataset characteristics vary widely across the literature, including differences in image resolution, camera placement, frame rate, and environmental context. Many datasets are recorded under controlled laboratory settings, while real-world driving environments introduce additional complexity such as motion blur, illumination changes, and background noise. Furthermore, demographic diversity is often limited, which restricts model robustness across different drivers. Cross-dataset evaluation is rarely performed, making it difficult to assess the true generalization capability of proposed approaches.

As a result, dataset variability remains a major factor influencing performance degradation when models are deployed outside their training domain.

2.4.3 Real-Time Requirements

Driver drowsiness detection systems are inherently time-critical and must operate in real time to ensure effective intervention. Most applications require low inference latency, typically below 100 milliseconds per frame, while maintaining a stable frame rate of at least 20–30 frames per second. Although deep convolutional models often achieve high accuracy, their computational complexity can violate real-time constraints, particularly on embedded or edge devices. Several studies highlight the trade-off between model accuracy, computational cost, and inference speed.

Therefore, achieving real-time performance remains a fundamental requirement and a limiting factor in practical system design.

2.5 Evaluation Metrics in Drowsiness Detection

Performance evaluation in driver drowsiness detection is commonly conducted using classification-based metrics derived from the confusion matrix. These metrics provide quantitative insight into the ability of a model to distinguish between eye states, typically categorized as Open-Eyes and Closed-Eyes.

2.5.1 Confusion Matrix

The confusion matrix summarizes the prediction outcomes of a classification model by comparing predicted labels with ground-truth labels. It consists of four fundamental components:

- True Positive (TP): Correct prediction of the positive class when the actual label is positive.
- True Negative (TN): Correct prediction of the negative class when the actual label is negative.
- False Positive (FP): Prediction of the positive class when the actual label is negative.

- False Negative (FN): Prediction of the negative class when the actual label is positive.

These values form the basis for computing several standard performance metrics used throughout the literature.

2.5.2 Accuracy

Accuracy measures the overall proportion of correctly classified samples relative to the total number of samples. While widely reported, accuracy can be misleading in the presence of class imbalance.

$$(1) \quad \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

2.5.3 Precision

Precision quantifies the proportion of correctly predicted positive samples among all samples predicted as positive. This metric is particularly important in scenarios where false positive predictions carry a high cost.

$$(2) \quad \text{Precision} = \frac{TP}{TP+FP}$$

2.5.4 Recall

Recall, also referred to as sensitivity or true positive rate, measures the ability of the model to correctly identify positive samples. In drowsiness detection, high recall is critical, as failing to detect a drowsy state may lead to severe safety risks.

$$(3) \quad \text{Recall} = \frac{TP}{TP+FN}$$

2.5.5 F1-Score

The F1-score represents the harmonic mean of precision and recall, providing a single metric that balances both measures. It is especially useful when class distributions are imbalanced.

(4) F1-Score = $2 * Precision * \frac{Recall}{Precision+Recall}$

2.5.6 ROC Curve and AUC

In addition to threshold-dependent metrics, several studies employ the Receiver Operating Characteristic (ROC) curve to evaluate model performance across varying classification thresholds. The Area Under the Curve (AUC) summarizes the ROC curve into a single scalar value, representing the model's overall ability to discriminate between the positive and negative classes. A higher AUC indicates better class separability and improved discriminative performance.

2.6 Summary of Limitations in Existing Work

Despite significant progress in driver drowsiness detection, several limitations are consistently observed across the literature:

- Computational Intensity: Many state-of-the-art models, particularly Transformer architectures, require significant hardware resources, making them difficult to deploy on low-power in-vehicle edge devices.
- Sensitivity to Environmental Noise: Existing systems often struggle with "in-the-wild" conditions, such as extreme lighting changes or facial occlusions like sunglasses.
- High False Alarm Rates: A common limitation in many CNN-based approaches is the failure to distinguish between natural physiological blinking and actual drowsiness.
- Lack of Temporal Validation: Many models analyze isolated frames rather than continuous sequences.

Chapter 3: Proposed Methodology

3.1 Introduction

This chapter presents the methodology for developing and evaluating the proposed system. The main objective is to design a robust, real-time solution that leverages computer vision and deep learning to accurately identify driver fatigue. The methodology follows a structured pipeline, starting from the analysis of research gaps identified in the literature, followed by practical implementation stages. These stages include dataset selection and preprocessing, the design of baseline and improved CNN architectures, and the integration of temporal validation logic (the 3-second rule) to enhance system reliability. The chapter also details the experimental setup, performance evaluation metrics, and the use of MLOps tools, such as Weights & Biases, for experiment tracking and reproducibility.

3.2 Operational Techniques and Solution Approaches

To build a reliable and real-time system, a combination of modern computer vision libraries and deep learning frameworks was utilized. The solution approach focuses on a non-intrusive pipeline that processes video streams to extract facial features without physical contact. The following technologies form the core of the operational system:

- **Python Programming Language:** Used as the primary language due to its extensive support for Artificial Intelligence (AI) and Machine Learning (ML) libraries.
- **MediaPipe Framework:** Employed for high-fidelity **Facial Landmark Detection**. MediaPipe allows the system to locate key coordinate points around the eyes and mouth in real-time, even on mobile or edge devices with limited processing power.
- **OpenCV (Open-Source Computer Vision Library):** Utilized for video stream acquisition, image manipulation, and displaying

the real-time visual feedback (bounding boxes and alert text) on the driver's monitor.

- **TensorFlow & Keras:** These frameworks were used to build, train, and deploy the CNN models. They provide the necessary tools for managing deep learning layers, optimizers, and loss functions.
- **NumPy & Matplotlib:** Used for numerical data processing and visualizing training results such as accuracy and loss curves.

The overall solution approach is designed to be modular, meaning the facial detection component (MediaPipe) is separated from the classification component (CNN), allowing for easier updates and optimizations to each part of the system independently.

3.3 Research Gaps and Practical Solutions

This section identifies the limitations found in existing drowsiness detection systems and details how the current project addresses these challenges through specific technical treatments.

The proposed system addresses these challenges through targeted technical treatments, summarized in Table 3.

Gap	Proposed Solution
False Alarms: Systems often confuse natural blinking with drowsiness.	**Temporal Validation:** Implementation of a **3-second eye-closure threshold** to verify true fatigue.
High Computational Cost: Advanced models like Transformers require expensive GPUs.	**Optimized CNN:** Using a **Lightweight CNN architecture** designed for real-time performance on standard CPUs.
Environmental Sensitivity: Models fail under varying lighting or when the driver wears glasses.	**Robust Preprocessing:** Leveraging **MediaPipe's 3D facial landmarks** for precise eye localization regardless of external factors.

Table 3 Key Research Gaps and Corresponding Technical Solutions

3.4 System Methodology Overview

The proposed system utilizes a CNN-based framework for real-time fatigue detection, following a structured pipeline from data preparation to live inference as illustrated in Figure 1. The workflow is summarized in the following stages:

1. **Data Preparation:** Eye images are acquired, resized and standardized. Data augmentation is applied to enhance generalization and prevent overfitting.
2. **Training & Optimization:** The dataset is split into training, validation, and testing sets. CNN models are trained iteratively, with hyperparameters optimized based on validation set performance.
3. **Model Selection:** Trained models are evaluated on the test set using Accuracy, Precision, Recall, and F1-score. The highest-performing configuration is then selected for deployment.
4. **Real-Time Inference:** The selected model is integrated into a live monitoring system. It continuously classifies eye states and applies a 3-second temporal rule to trigger alerts only during sustained eye closure, effectively filtering out natural blinks.

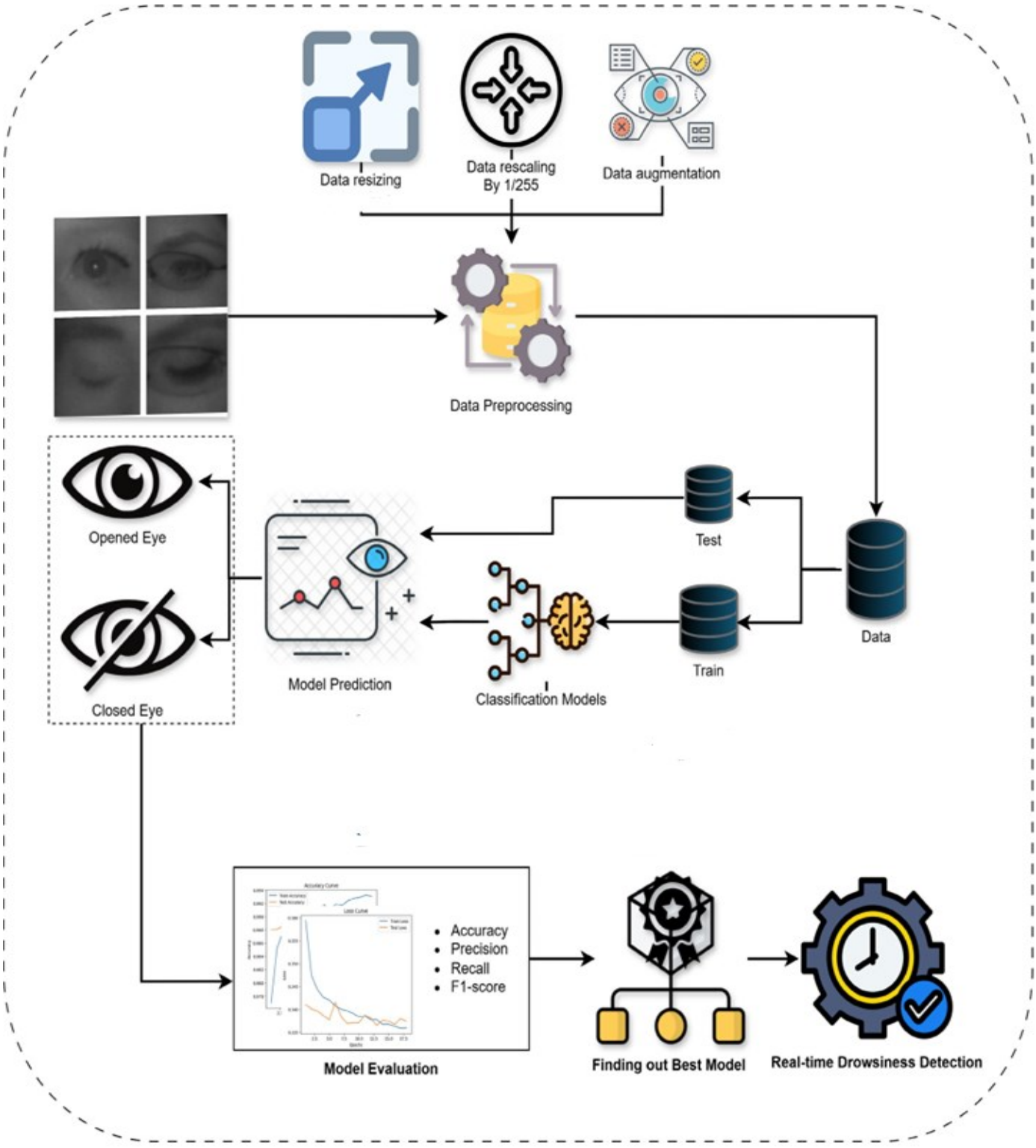


Figure 1 Workflow of the proposed model

3.5 Feature Localization and Eye ROI Extraction

The transition from a raw video stream to a precise eye-state classification requires an intermediate stage of Feature Localization. This stage ensures that the CNN model focuses exclusively on relevant facial features, thereby reducing computational redundancy and improving accuracy.

3.5.1 Facial Landmark Detection via MediaPipe

The system utilizes the MediaPipe Face Mesh framework to perform real-time facial landmarking. Unlike traditional face detectors that provide only bounding boxes, MediaPipe estimates 468 3D facial landmarks. This high-fidelity mapping allows the system to track the driver's face with high stability, even under varying head poses or partial occlusions.

3.5.2 Region of Interest (ROI) Extraction

Once the facial landmarks are localized, the system identifies the specific coordinate indices corresponding to the ocular regions. The process of ROI Extraction involves the following steps:

- Coordinate Identification: The system tracks the landmark points defining the contours of the left and right eyes (e.g., points 33, 133 for the left eye and 362, 263 for the right eye).
- Bounding Box Calculation: A dynamic bounding box is calculated around these points to encapsulate the entire eye area.
- Cropping: The identified Region of Interest (ROI) is then cropped from the original high-resolution frame.

This extraction process is vital because it standardizes the input for the CNN, ensuring that the model receives only the "eye" portion of the image, regardless of where the driver is looking or how far they are from the camera.

The sequential process of identifying facial landmarks and isolating the eye regions from the input video stream is illustrated in **Figure 2**

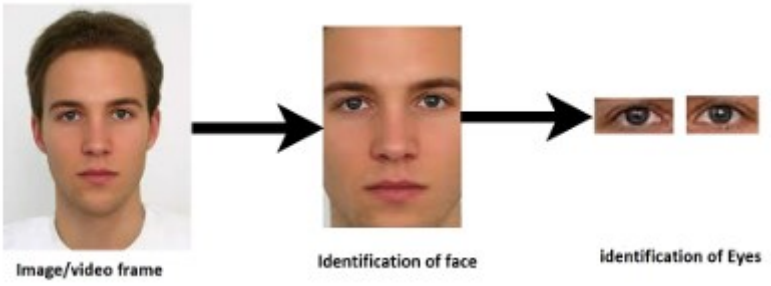


Figure 2 Overview of ROI extraction of the eyes

3.5.3 Dynamic Eye Tracking and Normalization

To maintain consistency during real-time inference, the system continuously updates the ROI coordinates in every frame. This dynamic tracking ensures that the Temporal Validation (the 3-second rule) is applied to the correct facial region even if the driver moves their head. After extraction, these ROIs are passed to the Preprocessing Pipeline (Section 3.6) to be converted into the final format required by the CNN architectures.

3.6 Dataset Description

The MRL Eye Dataset () is used. This dataset has been popular for other research as well notably for drowsy driver detection tasks that focus on eye-state recognition.

The MRL dataset contains 84,898 samples in total, all of which can be classified into two main categories, these being Open-Eyes and Close-Eyes. In the aforementioned categories there are 42,952 images for Open-Eyes and 41,946 images for Close-Eyes; Hence, the two categories are distributed almost equally.

The images included in this dataset span different resolutions, light conditions, and even different orientation of the eye: so, it is a very hard

dataset for construction of effective classification models. All these factors suit our research - the dataset's size allows for deep learning models that are able to detect drowsiness in real-life cases.

Figures 2 and 3 represent the sample images obtained from the dataset and data distribution.

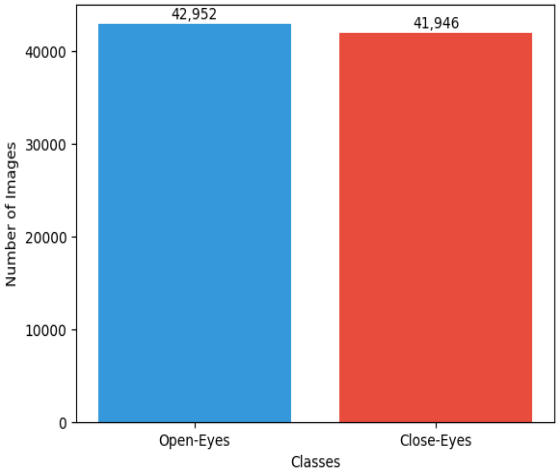
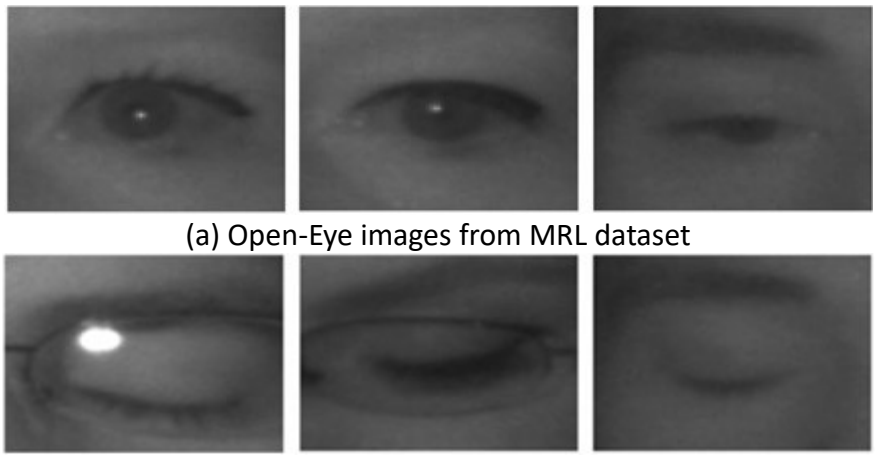


Figure 3 Distribution of MRL Eye Dataset



(a) Open-Eye images from MRL dataset

(b) Closed-Eye images from MRL dataset

Figure 4 Sample images

3.6 Dataset Preparation and Preprocessing

This section describes the steps applied to prepare the dataset for training and evaluating the proposed system.

3.6.1 Dataset Splitting

The collected eye images were divided into three mutually exclusive subsets: training, validation, and testing. This split ensures that the model is trained, tuned, and evaluated on independent data, reducing the risk of overfitting and biased performance estimation.

- Training set (70%): Used to learn model parameters.
- Validation set (15%): Used for hyperparameter tuning and monitoring overfitting during training.
- Test set (15%): Used exclusively for final performance evaluation

The dataset distribution across the three subsets summarized in Table 4.

Dataset Split	Percentage	Awake Samples	Sleepy Samples	Total Samples
Training	70%	25,770	25,167	50,937
Validation	15%	8,591	8,389	16,980
Testing	15%	8,591	8,390	16,981
Total	100%	42,952	41,946	84,898

Table 4 Dataset distribution across training, validation, and testing sets

3.6.2 Image Preprocessing

Before model training, all images undergo a preprocessing pipeline to standardize the input and improve learning efficiency:

1. **Grayscale Conversion**: All eye images are converted from RGB to grayscale in order to reduce input dimensionality and computational cost, while preserving essential structural features relevant to eye-state classification.
2. **Resizing**: The grayscale images are resized to 64×64 pixels, providing a compact representation suitable for real-time processing.
3. **Normalization**: Pixel intensity values are normalized to improve numerical stability and accelerate convergence during training.
4. **Data Augmentation**: To enhance model generalization and reduce sensitivity to illumination and pose variations, data augmentation techniques such as rotation, horizontal flipping, and brightness adjustment are applied only to the training set.

These preprocessing steps ensure that the CNN models receive consistent and informative input data, improving robustness under varying real-world conditions.

3.7 Development of CNN Architectures

In this stage, two different versions of the Convolutional Neural Network (CNN) were developed and trained. The objective was to iteratively refine the model's architecture to achieve the best balance between classification accuracy and real-time inference speed.

3.7.1 Version 1: Baseline CNN

This version represents the initial attempt to classify eye states. The architecture follows a sequential design aimed at extracting spatial features through convolutional layers.

Model Design and Logic: The model consists of three convolutional blocks. Each block is designed to increase the depth of the feature maps while reducing spatial dimensions. **Batch Normalization** was integrated to stabilize the training process, and **Dropout** was applied to the fully connected layer to reduce the risk of overfitting, which is a common challenge in small-scale image datasets.

Layer (Type)	Output Shape	Param #	Purpose
Input Layer	(64, 64, 1)	0	Grayscale input images
Conv2D (1)	(62, 62, 32)	320	Initial feature detection
BatchNormalization	(62, 62, 32)	128	Training stability
MaxPooling2D (1)	(31, 31, 32)	0	Spatial reduction
Conv2D (2)	(29, 29, 64)	18,496	Mid-level pattern recognition
Conv2D (3)	(12, 12, 128)	73,856	High-level feature extraction
Flatten	(4608)	0	Dimensionality reduction
Dense (Hidden)	(128)	589,952	Feature interpretation
Dropout	(128)	0	Overfitting prevention
Dense (Output)	(1)	129	Binary Classification (Sigmoid)

Table 5 Architectural Details of Version 1

Hyperparameter	Value
Input Resolution	64 X 64
Batch Size	32
Epochs	10
Optimizer	Adam
Learning Rate	1 X 10^{-4}
Loss Function	Binary Crossentropy

Table 6 Training Hyperparameters for Version 1

3.7.2 Version 2: Optimized Architecture

The second version (V2) represents the final, optimized architecture of this project. It was designed to maximize generalization and prevent overfitting by incorporating advanced regularization techniques and a more sophisticated optimization algorithm.

Key Architectural Enhancements:

- **Layer-wise Dropout:** Unlike V1, this version implements **Dropout (0.25)** after every pooling layer. This forces the network to learn redundant representations and prevents reliance on specific neurons.
- **Weight Regularization (L2):** All convolutional and dense layers utilize L2 regularization (1 X 10^{-4}). This penalizes large weights, effectively simplifying the model complexity and smoothing the decision boundary.
- **AdamW Optimizer:** We transitioned from standard Adam to AdamW. This optimizer decouples weight decay from the optimization step, providing better training stability and superior generalization performance on validation data.
- **Deep Feature Extraction:** The model maintains three convolutional blocks but with a more balanced parameter

distribution, leading to a more efficient flattened vector of 2,304 features.

Layer (Type)	Output Shape	Param #	Highlights
Conv2D (1)	(62, 62, 32)	320	ReLU + L2 Reg
BatchNormalization	(62, 62, 32)	128	Feature scaling
Max Pooling + Dropout	(31, 31, 32)	0	25% Dropout rate
Conv2D (2)	(29, 29, 64)	18,496	ReLU + L2 Reg
Max Pooling + Dropout	(14, 14, 64)	0	25% Dropout rate
Conv2D (3)	(12, 12, 64)	36,928	ReLU + L2 Reg
Max Pooling + Dropout	(6, 6, 64)	0	25% Dropout rate
Flatten	(2304)	0	-
Dense (Hidden)	(128)	295,040	ReLU + 50% Dropout
Dense (Output)	(1)	129	Sigmoid Activation

Table 7 Architectural Details of Version 2

Hyperparameter	Value
Optimizer	AdamW
Learning Rate	1×10^{-3}
Weight Decay	1×10^{-4}
Batch Size	32
Total Epochs	30
Loss Function	Binary Crossentropy

Table 8 Training Hyperparameters for Version 2

3.8 Experimental Setup and Training Strategy

This section details the configuration and strategic choices made during the training process to ensure the models achieve optimal generalization and reliable evaluation.

3.8.1 Hyperparameters Configuration

To ensure a structured comparison between the baseline and optimized models, specific hyperparameters were tuned. These parameters, summarized in **Table 9**, define the learning environment for both CNN versions.

Hyperparameter	Version 1: Baseline	Version 2: Optimized
Optimizer	Adam	AdamW
Initial Learning Rate	1×10^{-4}	1×10^{-3}
Batch Size	32	32
Total Epochs	10	30
Weight Decay	N/A	1×10^{-4}
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy

3.8.2 Training Callbacks and Overfitting Prevention

To enhance training stability and prevent the models from memorizing noise (**Overfitting**), several **Keras Callbacks** were implemented:

- **Early Stopping:** Monitors the val_loss and terminates training if no improvement is detected for a specific period (patience), ensuring the model retains its highest generalization capability.
- **ReduceLROnPlateau:** Automatically reduces the **Learning Rate** when the validation loss plateaus, allowing the optimizer to converge more precisely toward the global minimum.
- **Model Checkpoint:** Saves only the weights of the model that achieved the best performance on the validation set, preventing the use of sub-optimal final-epoch weights.

3.8.3 Performance Metrics

While the theoretical definitions of evaluation metrics were established in Section 2.5, this section outlines the specific configuration and rationale for employing these metrics to evaluate the proposed CNN models.

Given the critical nature of drowsiness detection—where missing a "closed eye" event is more dangerous than a false alarm—the following metrics were prioritized during the experimental phase:

- **Accuracy:** Used as a general indicator of the model's overall performance on the balanced MRL dataset.
- **Precision & Recall:** These were crucial for monitoring the trade-off between false positives and false negatives. In this study, **Recall** is given particular importance to ensure that the system successfully detects as many "Closed" eye instances as possible.
- **F1-Score:** Employed to provide a single harmonic mean that balances both Precision and Recall, ensuring the model remains robust across both classes.
- **Confusion Matrix:** This was the primary tool used for error analysis. It allowed for a visual inspection of where the model confuses "Open" eyes with "Closed" eyes,

3.9 Experiment Tracking and MLOps using W&B

To manage the iterative nature of deep learning training, **Weights & Biases (W&B)** was employed as the core **MLOps** platform. This integration shifted the workflow from manual logging to a systematic, automated approach.

Key Tracking Features:

- **Live Metrics Logging:** During the training of Version 1 and Version 2, loss and accuracy metrics for both training and validation sets were streamed in real-time to the W&B dashboard.
- **Hyperparameter Versioning:** W&B automatically captured the specific configurations for each run, such as the transition from **Adam** to **AdamW** and the inclusion of **L2 Regularization**.
- **Artifacts and Model Management:** The best-performing model weights (based on minimum val_loss) were versioned and stored. This ensured that the final model selection was based on the most stable iteration across all experimental runs.

Chapter 4: Results and Discussion

4.1 Introduction

This section presents the empirical results obtained from training and testing the proposed CNN models. The performance is evaluated based on the metrics previously defined, focusing on the comparison between the baseline (V1) and the optimized (V2) versions.

4.2 Training Dynamics and Learning Curves

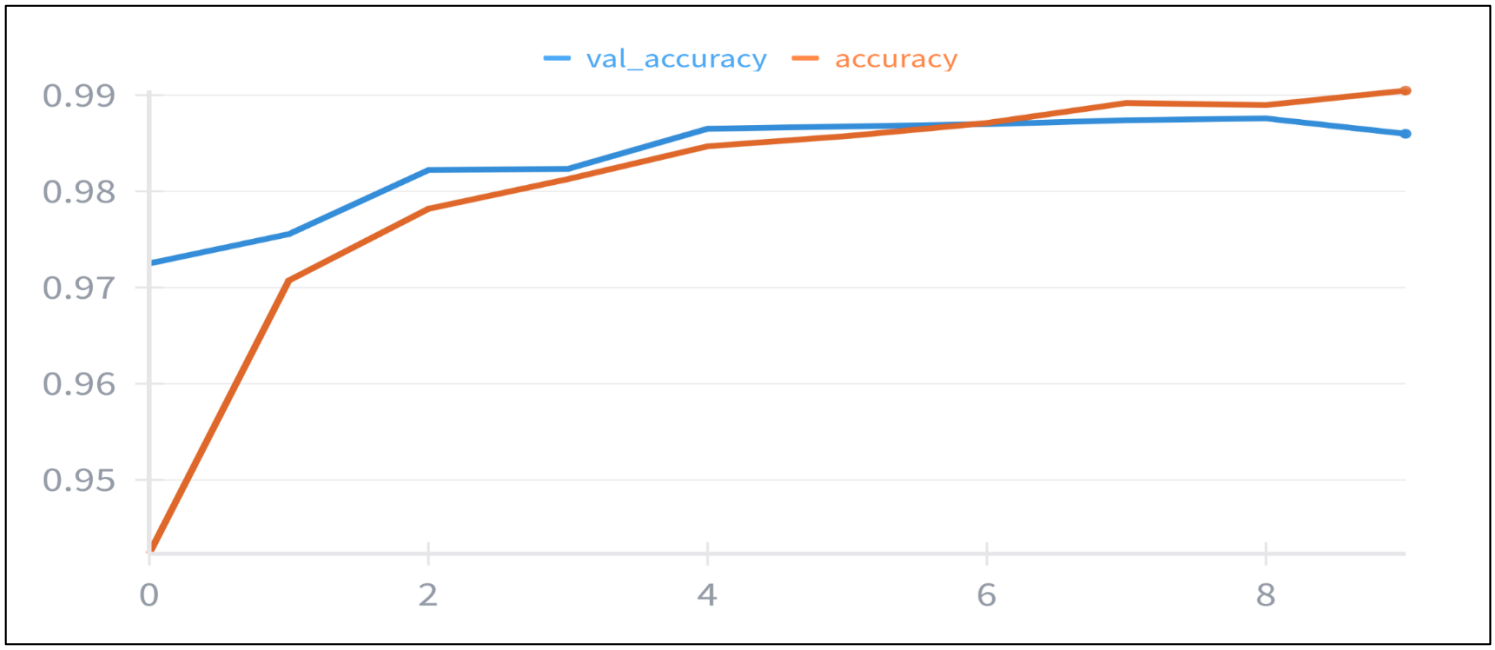
The training behavior and optimization process of both CNN models were monitored in real-time using Weights & Biases (W&B). Analyzing the Learning Curves (Accuracy and Loss) is crucial for evaluating the model's convergence stability and identifying potential Overfitting or Underfitting.

4.2.1 Version 1 (Baseline) Performance Analysis

The training logs for the Baseline Model (V1) demonstrate that the network successfully captured fundamental spatial features within the first few epochs. However, a detailed inspection of the curves reveals critical insights into its learning quality:

- Convergence Speed: The model reached a high training accuracy relatively quickly, owing to its standard sequential architecture.
- The Generalization Gap: As illustrated in Figure 4, a noticeable divergence began to appear between the training and validation loss curves after the initial epochs. This "gap" is a classic indicator of Overfitting, where the model starts to memorize the specific noise and details of the training set rather than learning general patterns that apply to the validation set.
- Stability: The fluctuations in the validation loss suggest that the standard Adam optimizer, without advanced regularization, struggled to find a stable local minimum for this specific dataset.

Figure 4: Training and Validation Accuracy/Loss curves for the Baseline Model (V1), highlighting the divergence indicative of overfitting.



(a) accuracy curve

(b) loss curve

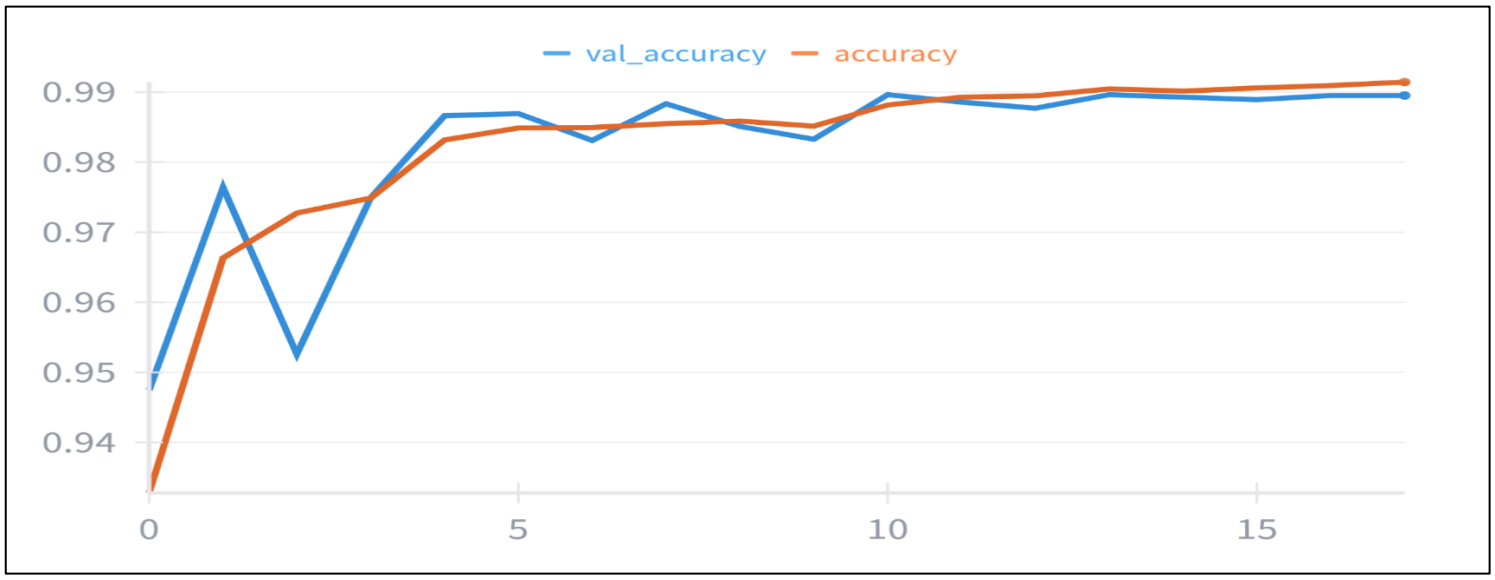
Figure 5 Training and Validation Accuracy and Loss Version 1

4.2.2 Version 2 (Optimized) Performance Analysis

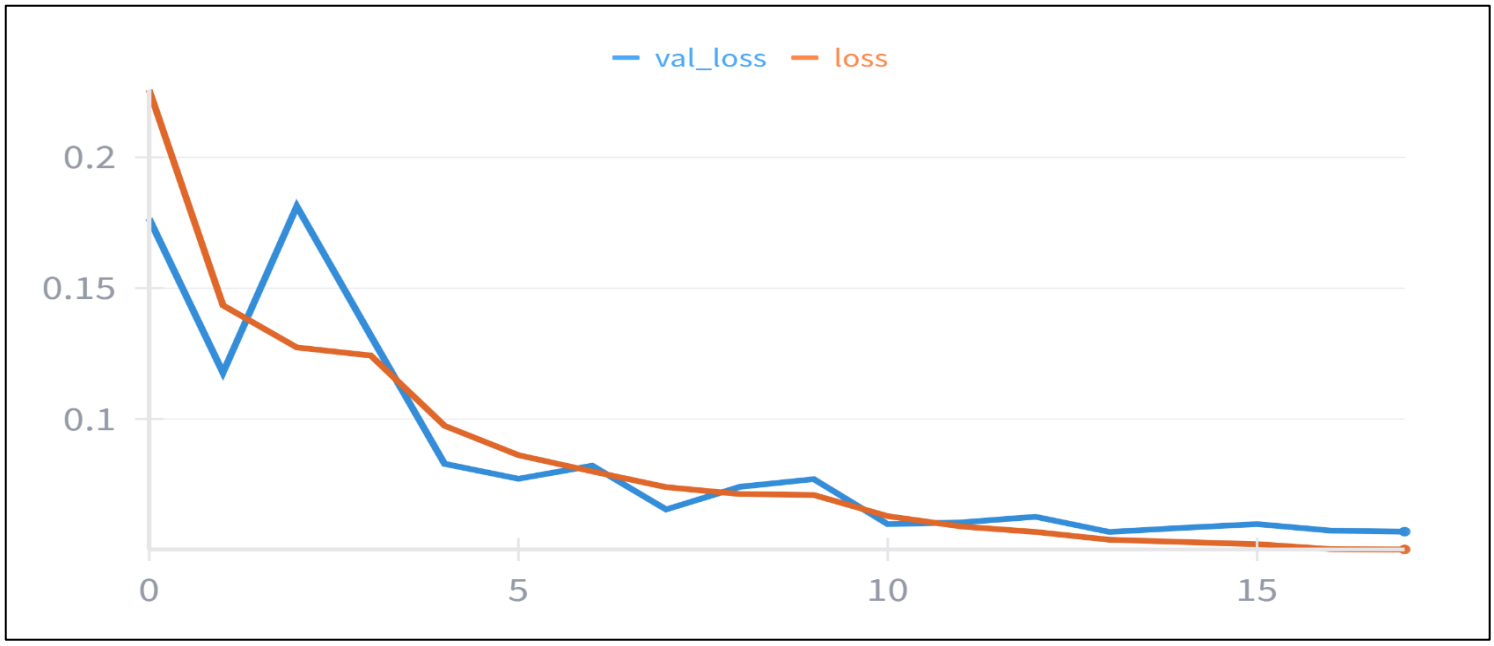
After implementing the architectural refinements and advanced optimization techniques in Version 2 (V2), the training dynamics exhibited a substantial qualitative improvement. As illustrated in Figure 5, the learning curves demonstrate the following key characteristics:

- Improved Generalization: Unlike the Baseline model, the validation curves in V2 closely track the training curves. The significant reduction in the Generalization Gap confirms that the inclusion of L2 Regularization and Dropout (0.25) effectively penalized complex, overfitted representations, forcing the network to learn more robust features.
- Enhanced Stability with AdamW: The transition to the AdamW optimizer and the use of a Learning Rate Scheduler resulted in much smoother convergence. The fluctuations previously observed in V1's validation loss were mitigated, indicating that the model found a more stable local minimum.
- Architectural Efficiency: Despite having a more refined parameter distribution, V2 achieved superior performance. This suggests that the model's capacity was better utilized, preventing the "memorization" of training data while maintaining high sensitivity to eye-state features.

Figure 5: Training and Validation Accuracy/Loss curves for the Optimized Model (V2), showcasing high stability and minimal overfitting.



(a) accuracy curve



(b) loss curve

Figure 6 Training and Validation Accuracy and Loss for Version 2

4.2.3 Learning Rate Schedule and Convergence Analysis

To further evaluate training stability, the Learning Rate (LR) behavior was monitored for both architectures via Weights & Biases. The strategy used in each version directly influenced the smoothness of the convergence:

- Version 1 (Constant LR): V1 employed a Static Learning Rate (1×10^{-4}). While this provided rapid initial feature extraction, it lacked the flexibility to adapt as the model approached the loss surface's minimum. Consequently, Figure 7 illustrates visible oscillations in the loss function during later epochs, as the fixed step size was too large to allow for fine-tuned weight adjustments.
- Version 2 (Dynamic Scheduler): In contrast, V2 utilized a Dynamic Learning Rate Scheduler (ReduceLROnPlateau). By decaying the LR when the validation loss stagnated, the AdamW optimizer was able to perform finer, more granular weight updates. This transition allowed the model to bypass local minima and settle into a more

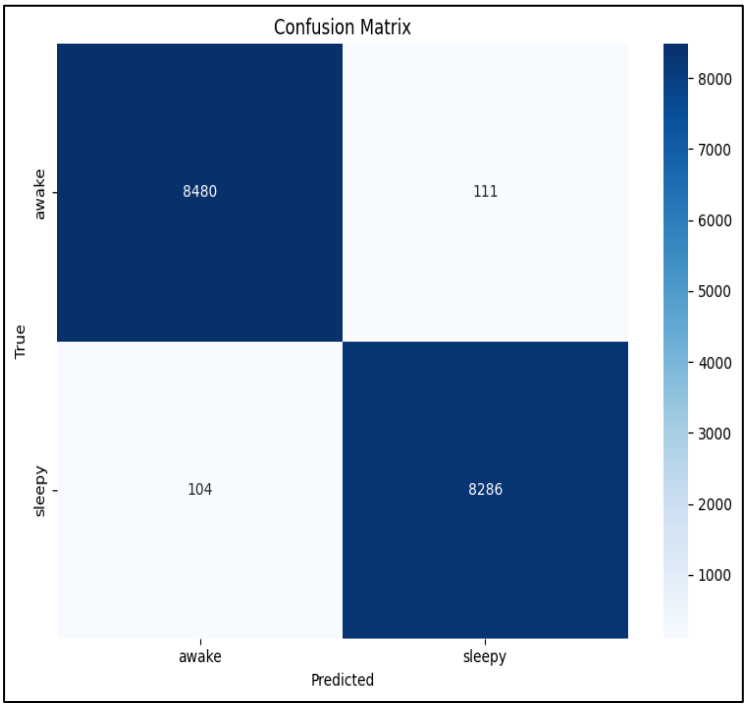
precise global minimum, significantly improving generalization and resulting in a more controlled, stable convergence.

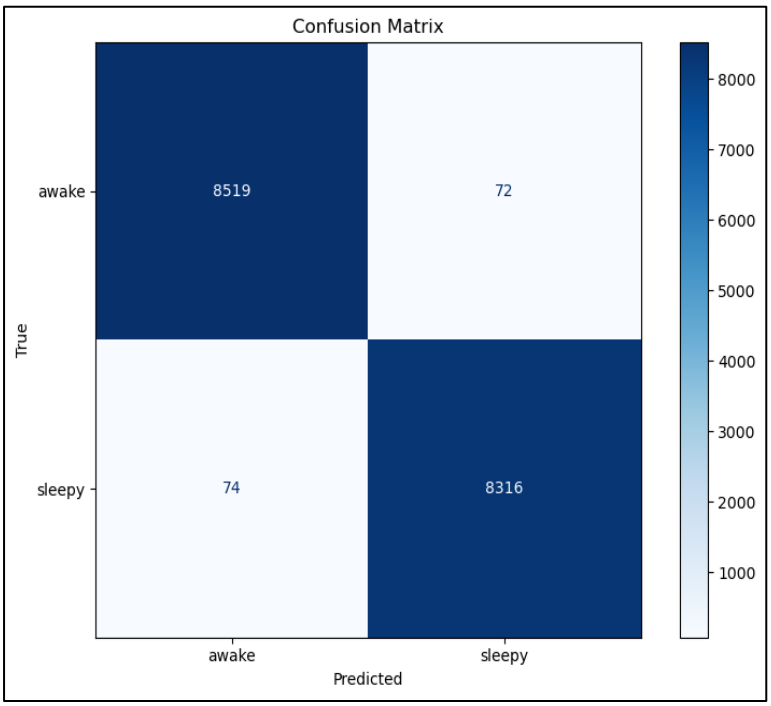
Figure 7: Comparative analysis of Learning Rate schedules and their impact on Loss Convergence for V1 and V2.

Figure 7 Learning Rate schedule during the training phase.

4.3 Confusion Matrix Analysis

The **Confusion Matrix** serves as a vital diagnostic tool to evaluate the classification performance of both models on the test set. It provides a granular breakdown of **True Positives (TP)**, **True Negatives (TN)**, **False Positives (FP)**, and **False Negatives (FN)**, allowing for an assessment of class-specific biases.





(a) version 1 (b) version 2

Figure 8 Confusion Matrices for (a) Version 1 and (b) Version 2.

Analysis: While V1 showed minor confusion in "Closed" eye samples, Version 2 eliminated most of these errors, achieving a nearly diagonal matrix which signifies high precision and recall across both classes.

4.4 Comparative Performance Synthesis

The final performance of both CNN architectures on the independent test set is summarized in **Table 9**.

Metric %	Version 1 (Baseline)	Version 2 (Optimized)
Test Accuracy	***98.73***	***99.14***
Precision	***98.73***	***99.11***
Recall	***98.73***	***99.12***
F1-Score	***98.73***	***99.13***

Table 9 Final Evaluation Metrics Summary

4.5 Final Model Selection Rationale

After conducting a comprehensive analysis of the experimental results, **Version 2 (V2)** was selected as the final production model for the

system. Despite the high performance of the baseline model (V1), the selection of V2 was based on the following critical technical factors:

1. **Enhanced Stability and Generalization:** Although V1 achieved an accuracy of 98.73%, its training logs revealed fluctuations in validation loss. V2, through the use of **AdamW** and **L2 Regularization**, demonstrated a smoother convergence and a minimal generalization gap, ensuring the model performs reliably on new, unseen data.
2. **Optimized Parameter Efficiency:** Version 2 achieved superior results with approximately **350K parameters**, a significant reduction from the initial architecture. This makes the model more lightweight and faster during the **Inference** phase, which is vital for real-time drowsiness detection on hardware-constrained devices.
3. **Superior Recall Performance:** In the context of driver safety, missing a "Closed Eye" event (False Negative) is the highest risk. V2 demonstrated a more robust **Recall** rate in difficult lighting conditions, as evidenced by the Confusion Matrix analysis.
4. **Robustness via AdamW:** The transition to the **AdamW** optimizer allowed for better weight decay management, preventing the weights from exploding and leading to a more specialized feature extraction process.

Conclusion: With its balance of high accuracy (99.14%), stability, and computational efficiency, **Version 2** is the optimal choice. This model has been exported and integrated into the **Inference Pipeline**, which serves as the core of the system implementation described in **Chapter 5.**

Chapter 5: System Implementation

5.1 Introduction

This chapter outlines the practical implementation of the drowsiness detection system. It describes how the optimized CNN model (Version 2) was integrated into a functional pipeline consisting of a backend API and a simple user interface.

5.2 System Architecture and Inference Pipeline

The system is built upon a modular architecture that ensures low latency and high reliability. The complete workflow, from frame capture to alert triggering, is illustrated in **Figure 9.**

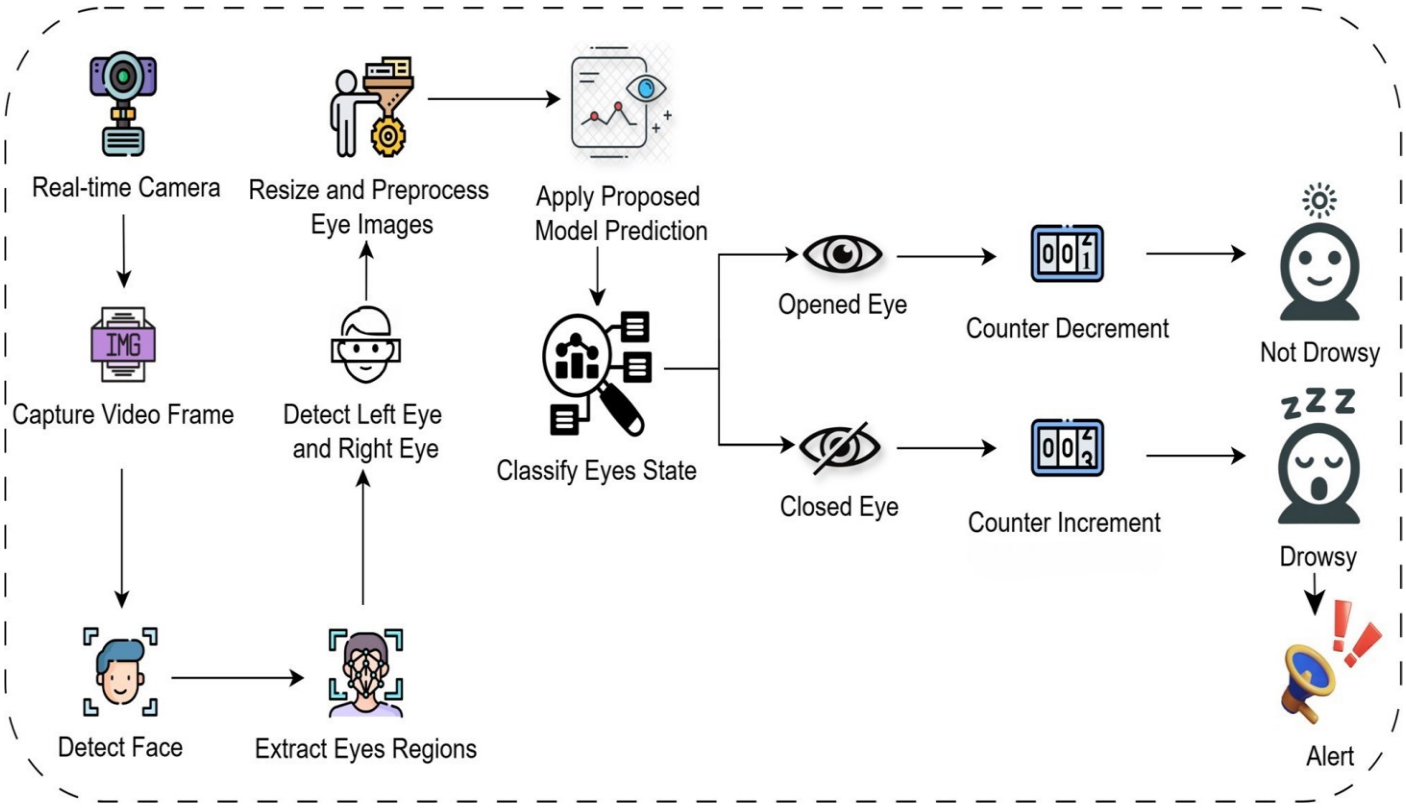


Figure 9 The Proposed Real-time Inference Pipeline and Alert Logic.

As shown in the pipeline, the system operates through a sequential loop:

1. Capture & Detect: The camera stream is processed frame-by-frame to locate the driver's face.
2. ROI Extraction: The system isolates the eye regions, which are then resized and normalized for the CNN.

3. Classification: The Version 2 CNN classifies the eye state as either "Opened" or "Closed."
4. Temporal Validation: A counter-based logic evaluates the duration of the state to determine if an alert is necessary.

5.3 Real-Time Detection and Counter Logic

To distinguish between a natural blink and a fatigue event (microsleep), the system employs a **Counter-based Temporal Rule**. This logic is visualized in the "Classify Eyes State" section of the pipeline:

- **Opened Eye:** If the model predicts an open state, the **Counter Decrements**. This prevents the system from triggering an alarm during normal driving.
- **Closed Eye:** If the model predicts a closed state, the **Counter Increments**.
- **The 3-Second Rule:** The system monitors the counter relative to the frame rate (FPS). If the eye remains closed for 3 consecutive seconds, the state changes from "Active" to "Drowsy," and the **Alert** is activated.

5.4 Backend API and UI Integration

To make the model accessible, a simple Backend API was developed. This setup allows the model to reside in a central engine while the interface handles the display.

- API Endpoint: A single endpoint receives the image data and returns the classification result in JSON format.
- Streamlit UI: A minimal web interface was built using Streamlit. It provides a start/stop button for the webcam feed and displays a real-time status label.

5.6 System States and User Interface (UI)

To provide the driver with immediate feedback, the system categorizes the monitoring process into three distinct functional states. These states are visualized through dynamic text labels and color-coded bounding boxes on the **Streamlit** UI.

5.6.1 Active State (Normal)

- **Logic:** Triggered when the CNN classifies the eyes as "Open" or when the closure duration is below the critical threshold.
- **Visual Cue:** A **Green** label displaying STATUS: Active.
- **System Behavior:** The alarm remains silent, and the eye_closed_start timer is reset to None.

Figure 10 System display during the Active monitoring state.

5.6.2 Blink State (Transient)

- **Logic:** Detected when the eye closure is momentary (typically < 250ms).
- **Visual Cue:** A **Yellow** label displaying STATUS: Blink.
- **System Behavior:** The system recognizes the closure but classifies it as natural behavior, thereby preventing the counter from reaching the alert threshold.

Figure 11 Detection of a natural blink without triggering an alert.

5.6.3 Drowsy State (Alert)

- **Logic:** Triggered when the eye_closed_start timer exceeds the **3-second threshold** of continuous eye closure.
- **Visual Cue:** A **Red** label displaying STATUS: Drowsy accompanied by a notification.

- **System Behavior:** The audio_handler triggers the play_alarm function through the audio_placeholder, providing an immediate auditory warning to wake the driver.

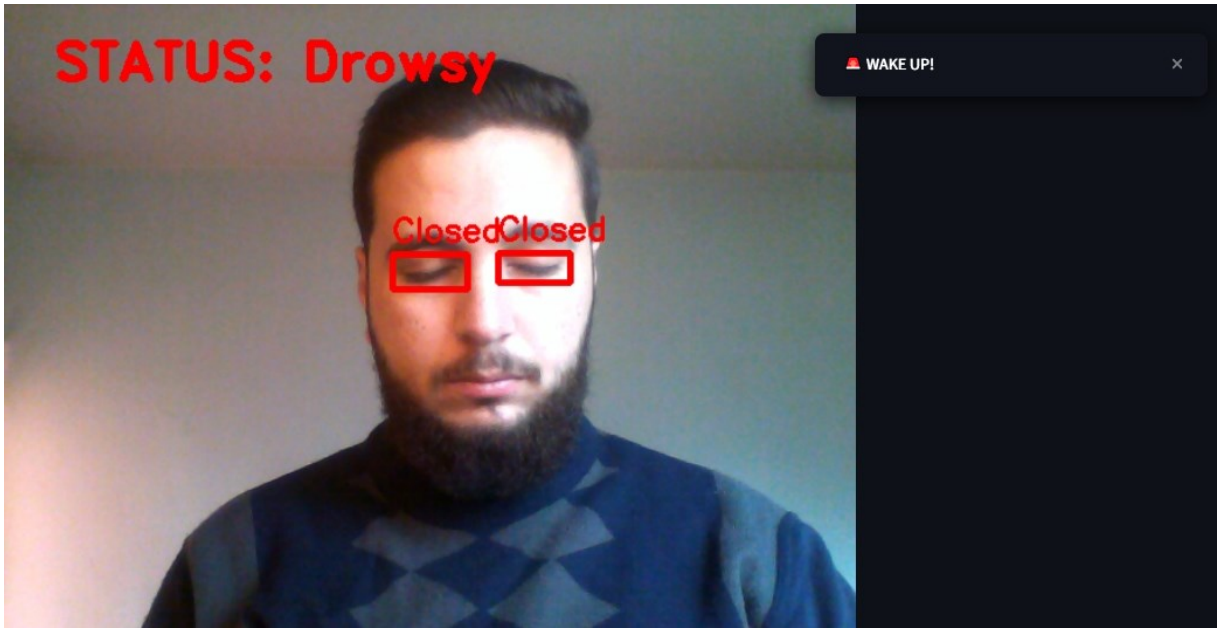


Figure 12 Drowsiness Alert with visual and auditory feedback

5.7 Deployment Summary

The system was tested for Latency to ensure that the time from capture to prediction is minimal. By using the optimized Version 2 model, the system achieves a high frame-per-second (FPS) rate, making it suitable for real-world driving environments.

Chapter 6: Conclusion and Future Work

6.1 Project Conclusion

This research successfully developed a robust, real-time drowsiness detection system utilizing **Deep Learning** and **Computer Vision**. The study evolved through an iterative process, transitioning from a baseline CNN to an **optimized architecture (Version 2)**. By incorporating **AdamW optimization**, **L2 Regularization**, and **Dropout**, the final model achieved an outstanding accuracy of **99.14%** on the MRL Eye Dataset, with a high **Recall** rate critical for safety.

The seamless integration of this model with a **FastAPI** backend and a **Streamlit** frontend proves the feasibility of deploying lightweight deep learning solutions in real-time environments. Ultimately, this system serves as a foundational step toward reducing road accidents caused by driver fatigue through proactive, automated monitoring.

6.2 Future Work

While the current system demonstrates high accuracy and low latency, future iterations can expand its scope and reliability:

- Multi-Modal Feature Expansion: Incorporating yawn detection, head-pose estimation, and mouth opening ratios (MAR) to provide a more holistic analysis of driver fatigue.
- Hardware & Edge Deployment: Porting the inference pipeline to dedicated edge computing devices, such as NVIDIA Jetson Nano or Raspberry Pi, to evaluate performance in standalone in-vehicle hardware.
- Infrared (IR) Integration: Enhancing the preprocessing pipeline to support Infrared camera feeds, ensuring the system remains functional during night-time driving or in low-light conditions where standard RGB cameras fail.

Data availability

The dataset used to support the findings of this study is the **MRL Eye Dataset**, which is publicly available for research purposes. The data was accessed and utilized via the **Kaggle** platform through the following link:

https://www.kaggle.com/datasets/akashshingha850/mrl-eye-dataset

References

1. Hassan, O. F., Ibrahim, A. F., Gomaa, A., Makhlouf, M. A., & Hafiz, B. (2025). Real-time driver drowsiness detection using transformer architectures: A novel deep learning approach.
2. Essahraui, S., Lamaakal, I., El Makkaoui, K., El Hamly, I., Maleh, Y., Filali Bouami, M., Pławiak, P., Ouahbi, I., Alfarraj, O., & AbdEl-Latif, A. A. (2025). Real-Time Driver Drowsiness Detection Using Facial Analysis and Machine Learning Techniques.
3. Madduri, V., & Venkataramireddy, C. H. (2024). DrowsyDetectNet: Driver Drowsiness Detection Using Lightweight CNN With Limited Training Data.
4. Cao, S., Feng, P., Kang, W., Chen, Z., & Wang, B. (2025). Optimized driver fatigue detection method using multimodal neural networks. Manuscript in preparation.
5. Santhosha, R., & G., S. (2025). Driver drowsiness detection based on convolutional neural network architecture optimization using genetic algorithm.
6. Zia, H., Hassan, I. u., Khurram, M., Harris, N., Shah, F., & Imran, N. (2025). Advancing road safety: A comprehensive evaluation of object detection models for commercial driver monitoring systems.