iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 13 **Issue:** VII **Month of publication:** July 2025

**DOI:** https://doi.org/10.22214/ijraset.2025.73341

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue VII July 2025- Available at www.ijraset.com*

# Driver Vigilance Monitoring System

Kondamudi Manoj Kumar[1], K.S.S. Soujanya Kumari[2]
[1]*Student, Department of Information Technology & Computer Applications,*
[2]Assistant Professor, *Department of Computer Science & Systems Engineering, Andhra University College of Engineering(A), Andhra University, Visakhapatnam, Andhra Pradesh – 530003*

***Abstract:*** ***Driver doziness is a significant contributor to road accidents encyclopedically, challenging robust and real- time discovery systems. This paper presents a comprehensive motorist Alert Monitoring System using deep literacy and computer vision ways to descry motorist fatigue by assaying eye countries. The system employs a Convolutional Neural Network (CNN) trained on grayscale eye images to classify eye countries as' Open' or' Closed'. MediaPipe's Face Mesh result is employed for precise facial corner discovery and birth of eye regions, which are also preprocessed with CLAHE and regularized before feeding into the trained model. A critical aspect of the system involves covering the duration of unrestricted eye countries; if a predefined threshold is exceeded, an audio alarm is touched off to warn the motorist, along with visual cues on the display. The advanced system demonstrates high delicacy in eye state bracket and effective real- time doziness discovery, validated through comprehensive training and evaluation criteria including perfection, recall, F1- score, ROC AUC, and confusion matrices. A stoner-friendly Tkinter- grounded visual interface facilitates system control and provides real- time status updates and access to discovery logs, offering a practical result for enhancing road safety.***

***Keywords: Driver Doziness, Real-Time Detection, Convolutional Neural Network (CNN), Grayscale Eye Images, Eye State Classification, MediaPipe Face Mesh, Facial Landmark Detection, CLAHE, Preprocessing, Audio Alarm, Tkinter Interface, Detection Logs, High Accuracy, Precision, Recall, F1-Score, ROC AUC, Confusion Matrix.***

## I. INTRODUCTION

Road safety remains a paramount global concern, with driver fatigue consistently identified as a major contributing factor to vehicular accidents. Drowsy driving impairs a driver's cognitive functions, reaction time, and decision-making capabilities, making it as dangerous as, or even more dangerous than, impaired driving due to alcohol or drugs [1]. As urban and inter-city travel increases, the need for effective countermeasures against driver fatigue becomes increasingly critical.

Traditional methods of addressing driver drowsiness often rely on self-assessment, scheduled breaks, or external stimulants, which are often insufficient or reactive. The advent of advanced computer vision and machine learning technologies offers a proactive approach by continuously monitoring a driver's state in real-time. Among various physiological and behavioral indicators of drowsiness, eye-related features, particularly eye closure, are highly reliable and widely recognized indicators of fatigue [2]. The Percentage of Eyelid Closure Over the Pupil Over Time (PERCLOS) is a well-established metric for assessing drowsiness by quantifying the duration an individual's eyelids are closed over a specific period [3].

This paper details the design, implementation, and evaluation of a real-time Driver Vigilance Monitoring System. The system's primary objective is to accurately detect when a driver's eyes are closed for an extended duration, indicating potential drowsiness, and subsequently alert the driver to prevent accidents. Our approach integrates a custom-built Convolutional Neural Network (CNN) for eye state classification with MediaPipe for robust facial landmark detection. The system is designed to be non-intrusive, operating solely through a webcam, making it suitable for practical vehicle integration.

The following is the structure of the following sections of this paper: Driver tiredness detection-related research is included in Section II. Section III elaborates on the proposed system's architecture and methodology, including data preprocessing, CNN model design, and drowsiness detection logic. Section IV covers the implementation details. Section V presents the experimental results and a thorough discussion of the system's performance. Section VI wraps up the work and suggests possible directions for improvement in the future.

## II. RELATED WORKS

The field of driver fatigue detection has seen substantial research and development over the past few decades, categorized broadly into three main approaches: physiological, behavioral, and vehicular-based methods.

Physiological methods involve monitoring biological signals such as electroencephalography (EEG), electrocardiography (ECG), electrooculography (EOG), and electromyography (EMG) [4]. While highly accurate, these methods often require intrusive sensors attached to the driver, which can be uncomfortable and impractical for widespread adoption in personal vehicles.

Vehicular-based methods analyze driving patterns, such as steering wheel movements, lane deviations, and vehicle speed fluctuations [5]. These systems infer drowsiness from erratic driving behaviors. However, their effectiveness can be limited by driving style variations, road conditions, and external factors not directly related to driver fatigue.

Behavioral methods, particularly those utilizing computer vision, have gained significant traction due to their non-intrusive nature and direct observation of driver states. These approaches commonly focus on facial features, including eye closure, yawning, head pose, and facial expressions. Early research established the significance of eye closure duration as a key indicator of drowsiness. Wierwille et al. [3] carried out the fundamental research that made PERCLOS a trustworthy metric. Subsequent studies, such as that by Ranjani and Kavitha [6], explored algorithms combining PERCLOS with other metrics for improved accuracy.

Vision-based fatigue detection systems are now much more capable because to recent developments in deep learning. Convolutional Neural Networks (CNNs) have demonstrated exceptional efficacy in image-based classification tasks, such as the recognition of eye states. Emin et al. [7] and Yang et al. [8] demonstrate the application of deep learning models for comprehensive driver assistance systems, highlighting the potential for real-time performance and high accuracy. These works often focus on detecting various aspects of driver behavior, with eye state classification being a core component. The rise of robust facial landmark detection libraries like MediaPipe has further streamlined the process of accurately localizing and extracting critical facial regions for analysis. Our proposed system builds upon these advancements by combining MediaPipe for precise eye region isolation with a refined CNN architecture and a robust real-time drowsiness detection algorithm. We specifically focus on optimizing eye image preprocessing with techniques like CLAHE and leveraging a tuned thresholding mechanism to improve the reliability and responsiveness of the drowsiness alert system.

## III. SYSTEM ARCHITECTURE AND METHODOLOGY

The Driver Vigilance Monitoring System is built upon a modular and layered architecture, designed for robust real-time performance and scalability. This section details the system's components, including data acquisition, the deep learning model, training methodology, drowsiness detection logic, and the user interface.

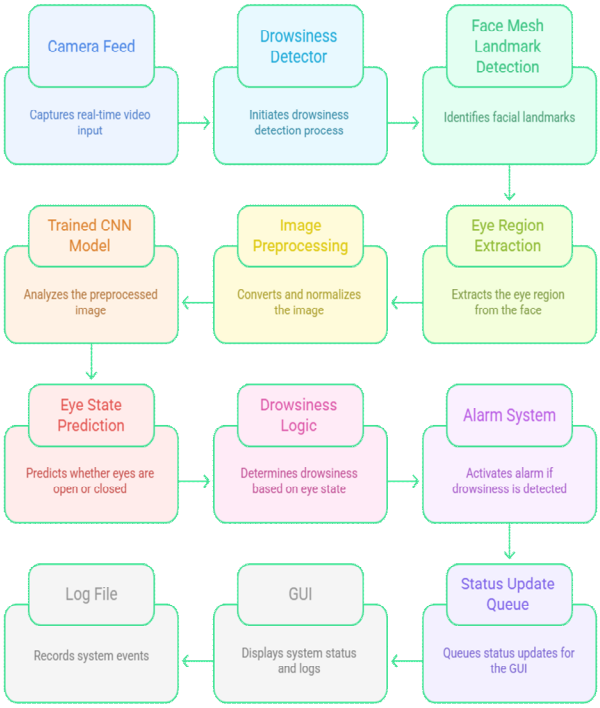



Fig. 1 Driver Vigilance Monitoring System Architecture

### A. *Data Acquisition and Preprocessing*

The system acquires live video frames from a standard webcam, which then undergo a series of preprocessing steps to prepare the data for the deep learning model.

*1)* Frame Acquisition and Flipping: Video frames are continuously captured and horizontally flipped to provide a natural mirror view to the user.

*2)* Color Conversion: Frames are converted from BGR (Blue-Green-Red), the default format for OpenCV, to RGB (Red-Green-Blue) for compatibility with MediaPipe processing.

*3)* Face and Landmark Detection: MediaPipe's Face Mesh solution is utilized to accurately detect the driver's face and identify 468 3D facial landmarks. This robust detection ensures precise localization of facial features, even under varying lighting conditions and head poses.

- Face Region of Interest (ROI) Extraction: A bounding box is dynamically computed around all detected facial landmarks. A small padding (10% on each side) is added to this bounding box to ensure the entire face is adequately captured. Next, the original frame's face ROI is clipped.
- Landmark Translation: The coordinates of the eye landmarks, initially relative to the full frame, are translated to be relative to the cropped face ROI. This ensures that subsequent eye cropping operates efficiently within the smaller face region.

*4)* Eye Region Extraction: Based on predefined MediaPipe landmark indices (specifically for the left and right eyes), tight bounding boxes are drawn around each eye within the face ROI. A dynamic margin (40% in X, 70% in Y) is empirically applied to these 6-point eye landmarks to include sufficient surrounding context (e.g., eyelids, eyebrows) around the eye, which is crucial for the CNN's classification accuracy.

*5)* Image Resizing and Grayscale Conversion: The extracted eye ROIs are uniformly resized to 24×24 pixels, matching the input dimensions required by the Convolutional Neural Network (CNN) model. They are then converted to grayscale to reduce computational load without significant loss of relevant information for eye state classification.

*6)* Contrast Limited Adaptive Histogram Equalization (CLAHE): To enhance the local contrast within the eye region, particularly in challenging lighting conditions, CLAHE is applied to the grayscale eye images. This technique effectively improves detail visibility while limiting noise amplification, which is vital for distinguishing open and closed eye states.

*7)* Normalization: The pixel intensity values of the grayscale eye images are normalized to a range of [0,1] by dividing by 255. This standard practice aids in faster and more stable CNN training.

*8)* Reshaping for Model Input: Finally, the normalized eye images are reshaped to (1,24,24,1) to align with the input shape expected by the TensorFlow/Keras CNN model (batch size, height, width, channels).

### B. *Deep Learning Model Architecture*

A Convolutional Neural Network (CNN) is specifically designed and trained to classify the preprocessed eye images as either 'Open' or 'Closed'. Figure 2 depicts the conceptual architecture of the model.

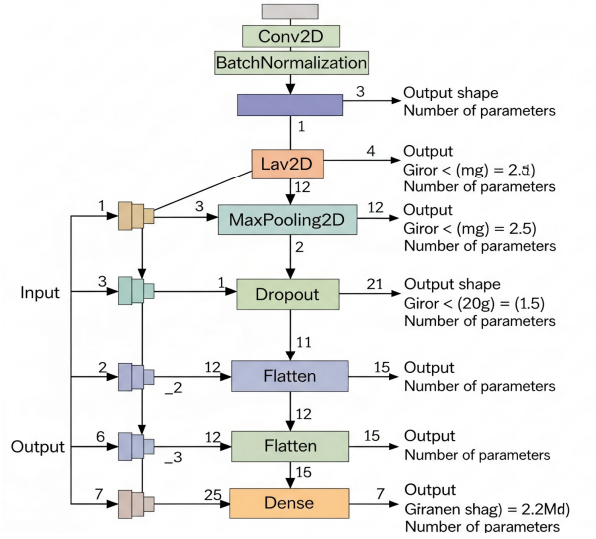



Fig. 2 Visual block diagram of the CNN architecture

The architecture comprises multiple convolutional blocks followed by fully connected layers.
Each Conv2D layer, serving as a primary feature extractor, is followed by a ReLU activation function to introduce non-linearity.
Batch Normalization layers are included after each Conv2D layer to stabilize and accelerate the training process by normalizing the activations of the preceding layer.
MaxPooling2D layers are strategically placed to down sample feature maps, thereby reducing dimensionality and enhancing spatial invariance.
Dropout layers are also incorporated to prevent overfitting by randomly deactivating a fraction of input units during training, forcing the network to learn more robust features. The output from the convolutional blocks is flattened into a 1D vector before being passed to Dense (fully connected) layers.
The final Dense layer employs a sigmoid activation function, outputting a probability score between 0 and 1, representing the likelihood of the eye being closed.

### *C. Model Training*

The CNN model is trained using a dataset comprising 'Open' and 'Closed' eye images. The training process employs several key parameters and strategies to optimize performance and prevent overfitting.

*1)* Dataset: The model is trained on a robust dataset of approximately 1,500 images, specifically the MRL Eye Dataset, which includes diverse images explicitly labeled as "Open" or "Closed" eyes.

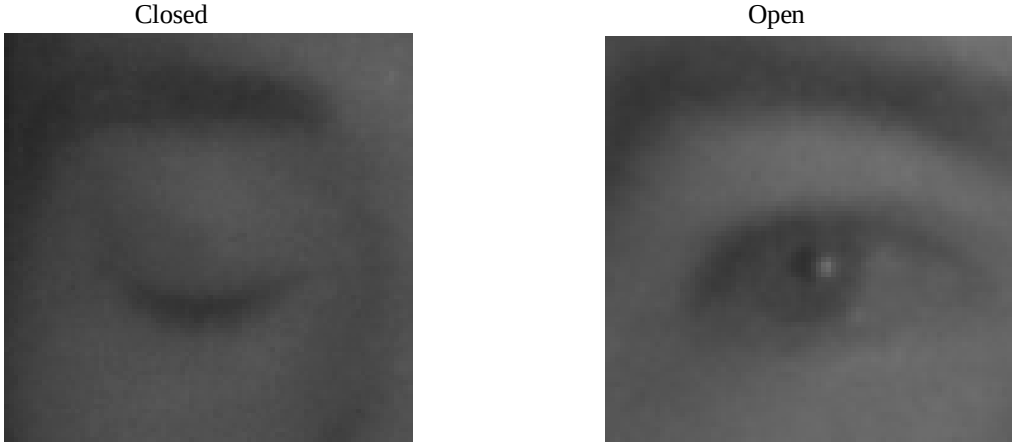



Fig. 3 Example Images from the Eye State Dataset (Closed and Open)

*2)* Image Size: All images are uniformly resized to 24×24 pixels and converted to grayscale for consistent CNN input.
*3)* Batch Size: A batch size of 32 is used during training.
*4)* Epochs: The training proceeds for up to 50 epochs, with an Early Stopping callback to determine the optimal training duration.
*5)* Optimizer: The Adam optimizer is employed, known for its efficiency and strong performance in deep learning tasks.
*6)* Loss Function: For binary classification issues, the loss function that is employed is Binary Cross-Entropy.
*7)* Metrics: Accuracy is monitored as the primary performance metric during training.
*8)* Data Augmentation: To prevent overfitting and enhance the model's generalization capabilities, extensive data augmentation techniques are applied using Keras's ImageDataGenerator. These techniques include:

- Rescaling pixel values to normalize them.
- 20% of the data is set aside for validation in order to assess model performance during training.
- Random rotations (rotation_range=15).
- Width and Height Shifts (width_shift_range=0.15, height_shift_range=0.15).
- Shear transformations (shear_range=0.1).
- Zoom operations (zoom_range=0.15).
- Brightness adjustments (brightness_range=[0.7, 1.3]).
- Horizontal flipping (horizontal_flip=True).

- fill_mode='nearest' for handling pixels outside the boundaries after transformations.

*9)* Callbacks:

- Early Stopping: This callback monitors validation loss and halts training if no improvement is observed for a set patience (e.g., 10 epochs), restoring the best weights achieved.
- Model Checkpoint: This callback saves the model with the lowest validation loss to eye_state_model.h5, ensuring the best performing model is preserved.

### *D. Drowsiness Detection Logic*

The real-time drowsiness detection system operates on the predictions generated by the trained CNN model, employing a robust state machine logic to differentiate between normal blinks and prolonged eye closures indicative of fatigue.

*1)* Eye State Prediction: For each incoming video frame, the pre-processed left and right eye images are fed into the trained CNN model. The model outputs a probability score for each eye, and the average of these two probabilities (avg_pred) is used to determine the overall eye state.

*2)* Binary Classification: A prediction threshold of 0.50 is utilized: if the average probability (avg_pred) is below 0.50, the eye is classified as 'Closed'; otherwise, it is considered 'Open'. This threshold was determined through empirical tuning to achieve a balanced sensitivity and specificity for eye closure detection.

*3)* Consecutive Closed Frames: To prevent false positives from normal, momentary blinks, the system tracks the number of consecutive frames where both eyes are classified as 'Closed'.

a. A CONSECUTIVE_CLOSED_FRAMES_TO_COUNT of 3 is set as a minimum requirement before initiating the drowsiness timer.

*4)* Drowsiness Threshold: If the eyes remain closed for a total duration exceeding DROWSINESS_THRESHOLD_SEC, which is set to 1.0 second, the system confidently registers a state of drowsiness.

*5)* Alarm Triggering: Upon confirmed drowsiness detection, an insistent audible alarm is immediately triggered using pygame.mixer. This alarm continues to play until the driver's eyes are detected as open for a SUSTAINED_OPEN_EYE_THRESHOLD_SEC (set to 4.0 seconds), indicating a recovery of alertness. A separate threading mechanism ensures the alarm plays asynchronously, preventing it from blocking the main video processing loop.

*6)* Logging: All significant operational events, including the start and stop of monitoring, instances of drowsiness detection, and alarm activations/deactivations, are meticulously timestamped and recorded to a persistent text file (detection_logs.txt) for later review and performance analysis.

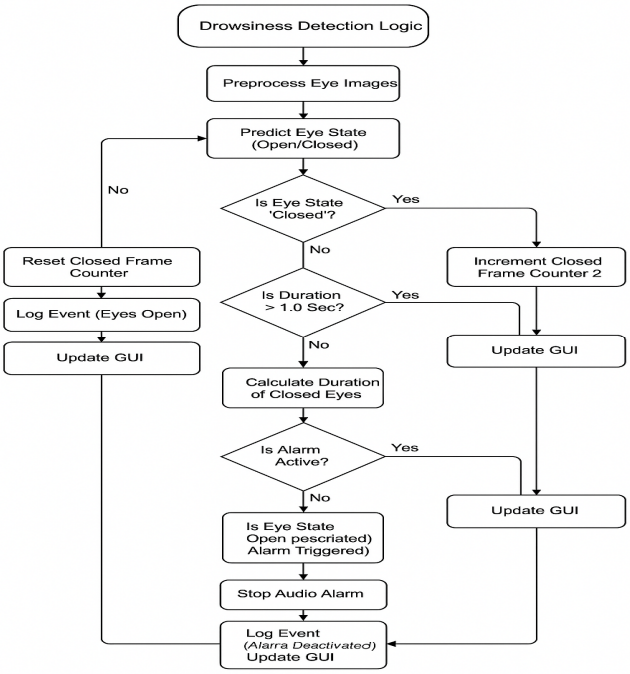



Fig. 4 Drowsiness Detection Logic Flow diagram

### *E. Graphical User Interface (GUI)*

A user-friendly graphical interface, developed using Python's Tkinter library, serves as the central control panel for the system. The GUI enables users to intuitively manage the system and receive real-time feedback.

The GUI provides direct control functions, allowing users to "Start Monitoring" to initiate the webcam feed and drowsiness detection process, and "Stop Monitoring" to halt the detection and close the webcam. A "View Logs" button offers convenient access to the detection_logs.txt file, providing a historical record of events. A real-time status label dynamically updates with the current operational state, displaying messages such as "Monitoring Active," "Eyes Closing...", or "DROWSY! ALARM!". In addition, the GUI has camera fault handling, which notifies the user if the webcam is unavailable.

The GUI runs in the main thread of the application, while the computationally intensive computer vision and deep learning tasks are executed in a separate daemon thread (detect_drowsiness), GUI maintains its responsiveness and offers a seamless user experience. A queue.Queue object is employed as a thread-safe mechanism for reliable and asynchronous communication of status updates from the detection thread to the GUI thread.

## IV. IMPLEMENTATION AND TECHNOLOGIES

The Driver Vigilance Monitoring System is developed entirely in Python, leveraging several key libraries and frameworks that collectively enable its real-time functionality and robust performance.

The core technologies utilized include:

1) TensorFlow/Keras: This framework is fundamental for building, training, and deploying the deep learning model, specifically the Convolutional Neural Network (CNN) used for eye state classification.
2) MediaPipe: Crucial for robust and efficient facial and eye landmark detection, MediaPipe's optimized pre-trained models provide high accuracy in real-time processing.
3) OpenCV (cv2): This library is indispensable for webcam access, various image processing operations (such as resizing, grayscale conversion, and drawing bounding boxes), and displaying the live video feed.
4) NumPy: Essential for numerical operations and efficient array manipulation, particularly for handling image data.
5) Pygame: Specifically, its pygame.mixer module is utilized for loading and playing the distinct audible alarm sound (alarm.wav) when drowsiness is detected.
6) Tkinter: As Python's standard GUI library, Tkinter is used to create the interactive and user-friendly desktop application interface.
7) threading and queue modules: These Python standard library modules are vital for managing the concurrent execution of the drowsiness detection logic and GUI updates, thereby ensuring a smooth and responsive user experience.
8) os and subprocess modules: These modules from Python's standard library are used for path management and facilitating cross-platform opening of the detection log file (detection_logs.txt).

The system architecture assumes a structured directory for its assets: the dataset directory is expected to be in the parent directory of the script, containing "Closed" and "Open" subfolders with respective eye images. The trained model (eye_state_model.h5) and the alarm sound (alarm.wav) are also located in a structured directory relative to the main scripts.

For operation, the system requires a webcam connected to the computer, with the default camera (index 0) being utilized. The chosen image size of 24×24 pixels for the eye images represents a critical balance between retaining sufficient detail for accurate classification and minimizing computational load to ensure real-time performance.

## V. RESULTS AND DISCUSSION

The performance of the Driver Vigilance Monitoring System was comprehensively evaluated in two primary phases: an analysis of the model training performance and an assessment of the real-time detection efficacy.

### *A. Model Training Performance*

The Convolutional Neural Network (CNN) model was trained and validated on a diverse dataset of 'Open' and 'Closed' eye images. The training process leveraged comprehensive data augmentation techniques and an early stopping mechanism to achieve optimal generalization and prevent overfitting. The key evaluation metrics obtained from the train_model.py script are summarized below:

1) Validation Loss: A low validation loss was achieved, indicating a good model fit and effective generalization to unseen data.

2) Validation Accuracy: The model demonstrated high accuracy on the validation set, confirming its robust ability to correctly classify eye states.
3) Precision (Open Class): A high precision of 0.9763 for the 'Open' class indicates a low rate of false positives, meaning the system rarely incorrectly classifies open eyes as closed.
4) Recall (Open Class): A recall of 0.9154 for the 'Open' class suggests the model is effective at identifying true open eyes.
5) F1 Score (Open Class): An F1 Score of 0.9448 for the 'Open' class provides a balanced measure of precision and recall, reflecting the model's overall accuracy, which is particularly important in potentially imbalanced datasets.
6) False Positive Rate (FPR): The FPR of 0.0228 signifies that only approximately 2.28% of actual 'Open' eye instances were incorrectly predicted as 'Closed' or drowsy.
7) False Negative Rate (FNR): The FNR of 0.0846 indicates that about 8.46% of actual 'Closed' eye instances were incorrectly classified as 'Open' or alert, highlighting an area for potential future refinement.
8) Area Under the Receiver Operating Characteristic Curve (AUC): An exceptionally high AUC value of 0.9894 was obtained. An AUC closer to 1.0 indicates excellent discriminative power, confirming the model's strong ability to distinguish between 'Open' and 'Closed' eye states across various classification thresholds. Important markers of model performance and training stability are the accuracy and loss curves for training and validation.

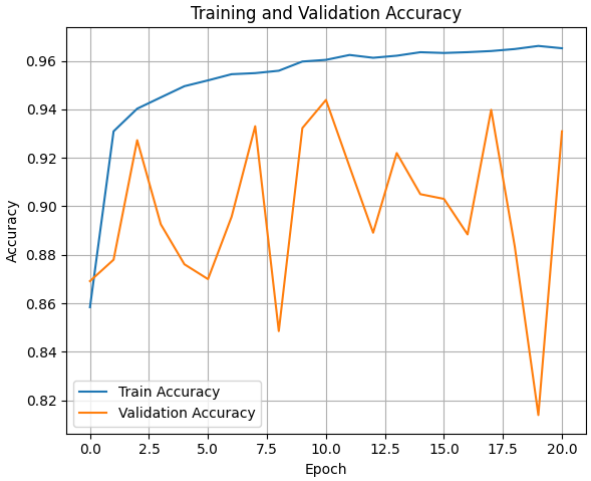



Fig. 5 Training and Validation Accuracy over Epochs.

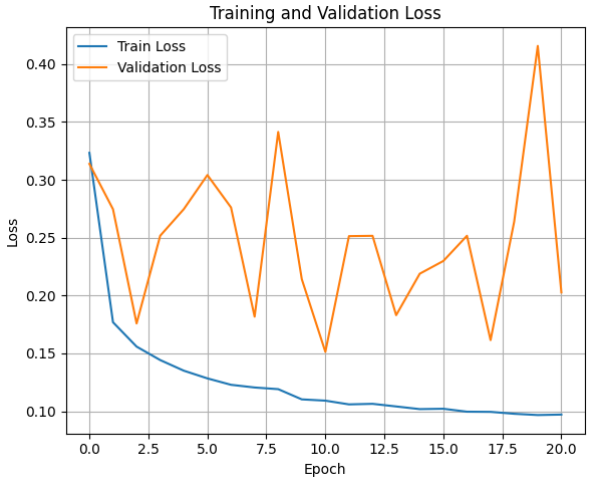



Fig. 6 Training and Validation Loss over Epochs.

Both the training and validation curves show convergence, as shown in Figures 5 and 6, indicating that the model is learning efficiently and not significantly overfitting. This stability is attributed to the implementation of data augmentation, batch normalization, and dropout layers during the training process.

The confusion matrix provides a detailed breakdown of correct and incorrect classifications, visually summarizing the model's performance on the validation dataset (Figure 7).

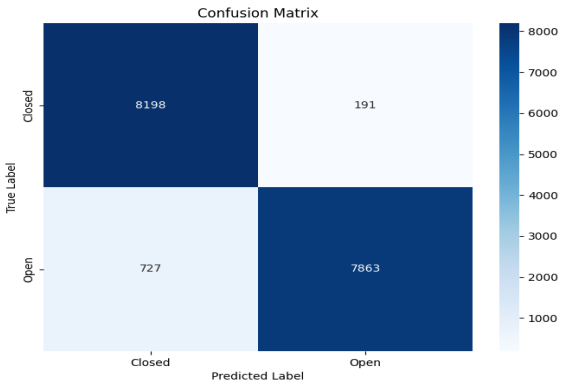



Fig. 7 Confusion Matrix for Eye State Classification on the Validation Set

The Receiver Operating Characteristic (ROC) curve further illustrates the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) at various threshold settings (Figure 8). The high AUC value of 0.9894 visually confirms the model's strong ability to differentiate between 'Open' and 'Closed' eye states effectively.

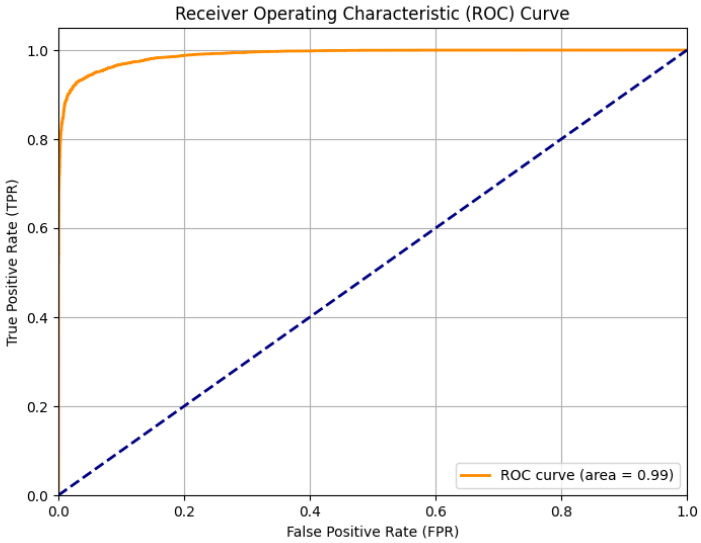



Fig. 8 Receiver Operating Characteristic (ROC) Curve.

*B. Real-time Detection Efficacy and User Interface*

The system's real-time performance was evaluated by directly interacting with the developed GUI and observing its behavior under simulated drowsiness scenarios.

The system's main interface is a Tkinter-based GUI titled "Driver Vigilance Monitoring System". Upon launching, it presents a clean layout with a "Status: Ready" message and interactive buttons: "Start Monitoring," "Stop Monitoring," "View Logs," and "Exit". This initial state confirms the GUI loads correctly and is ready for user interaction (Figure 9).

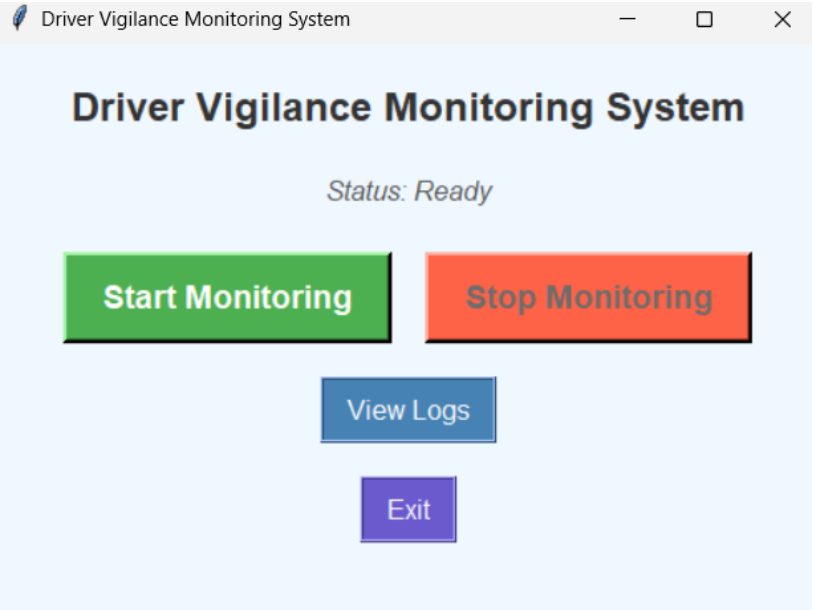



Fig. 9 GUI Initial State

Upon clicking "Start Monitoring," an OpenCV window titled "Driver Drowsiness Detector" opens, displaying the live webcam feed. The system actively tracks the driver's eyes, drawing green bounding boxes around them.

- Eyes Open State: When the driver's eyes are open, the status displayed in the OpenCV window correctly indicates "Eyes: OPEN". This confirms the model's ability to accurately classify open eyes in real-time (Figure 10).\

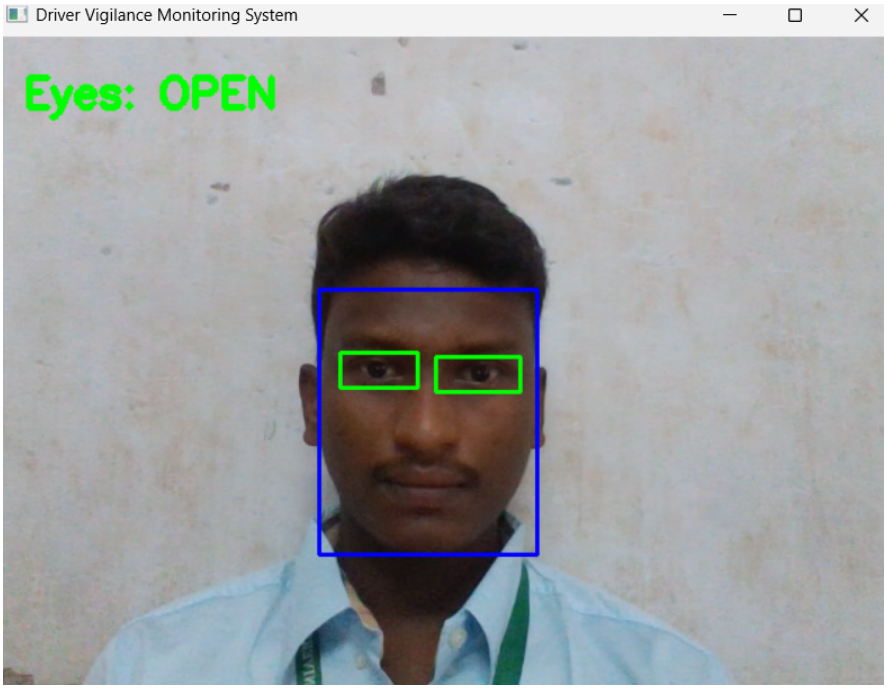



Fig. 10 OpenCV Window: Eyes Open State with Bounding Boxes

- Brief Eye Closure (Blinking): For short periods of eye closure, simulating natural blinking, the system updates the status to "Eyes Closed (X.Xs)", showing the duration of the closure. Importantly, it does not immediately trigger an alarm if the duration is below the set drowsiness threshold. This demonstrates the system's ability to differentiate between normal blinks and potential drowsiness (Figure 11).

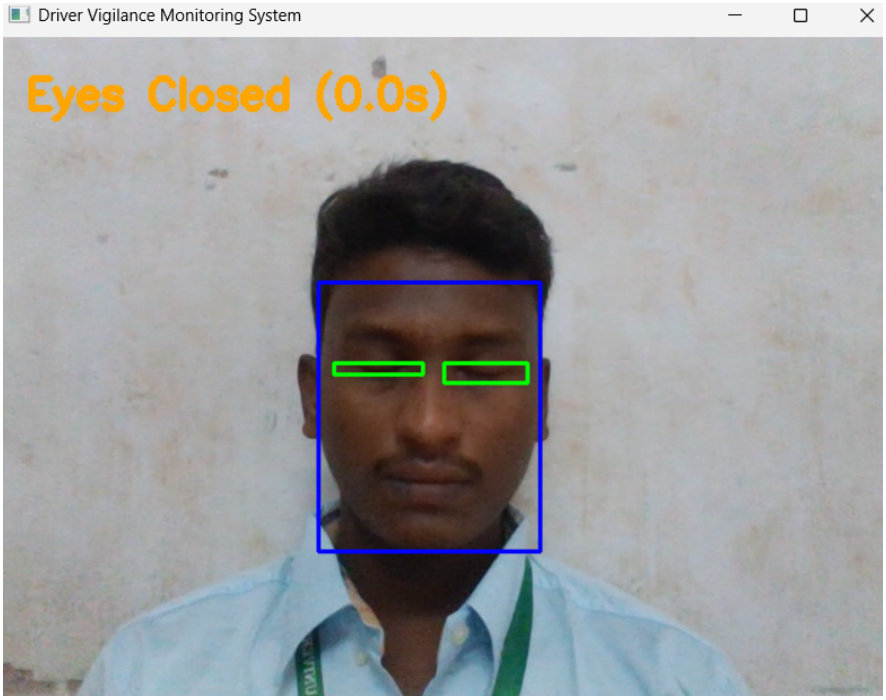



Fig. 11 OpenCV Window: Brief Eye Closure (Blink)

- Drowsiness Detection and Alarm Activation: When eyes remain closed for a duration exceeding the predefined threshold (e.g., 2 seconds, consistent with DROWSINESS_THRESHOLD_SEC), the system correctly identifies a drowsy state and activates an alarm. The OpenCV window prominently displays " DROWSY! Alarm! (X.Xs)" in red, indicating the current duration of eye closure. An audible alarm is simultaneously triggered, confirmed by log entries such as " Alarm function triggered and playing". Log entries further confirm detection and alarm activation, for instance: "[2025-07-20 13:51:45] Drowsiness detected - Eyes closed for 1.11s. Alarm triggered.". Subsequent logs show consistent triggers for longer durations (e.g., 2.07s, 2.12s), aligning with the

  DROWSINESS_THRESHOLD_SEC (Figure 12).

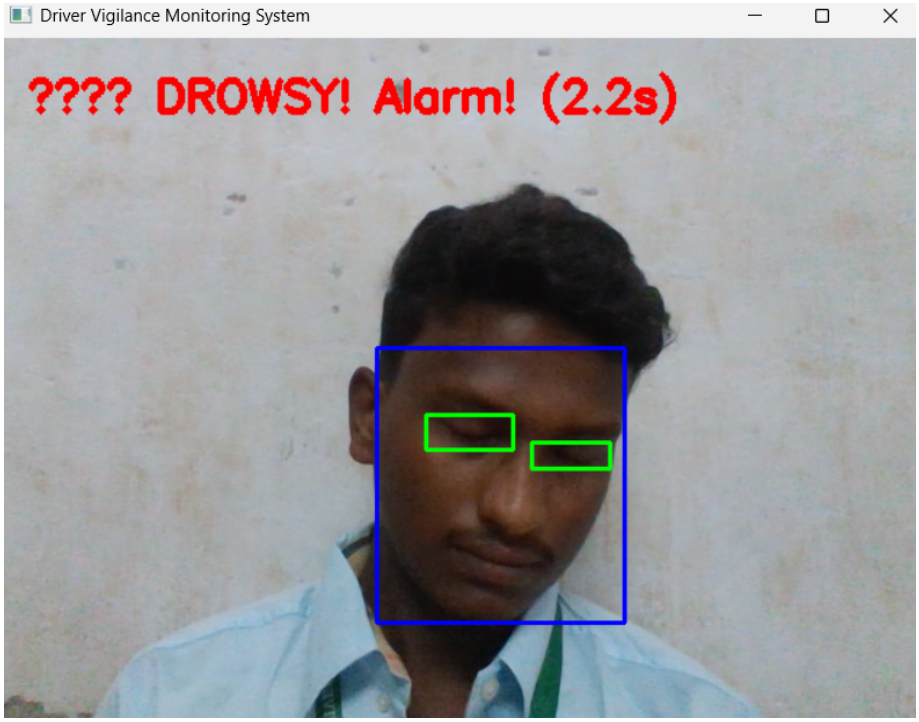



Fig. 11 OpenCV Window: Drowsiness Detected with Alarm Status

- Alarm Deactivation and Log Management: The system effectively deactivates the alarm once the driver's eyes remain open for a sustained period (e.g., 4 seconds, matching SUSTAINED_OPEN_EYE_THRESHOLD_SEC) after a drowsiness event. This is logged with messages such as " Eyes open for >4.0s. Stopping alarm." or " Alarm stopped due to external event (eyes open or app exit).". The detection_logs.txt file accurately records all significant events, including monitoring start/stop, drowsiness detections, and alarm activations/deactivations, complete with timestamps (Figure 12). This log file serves as a valuable record of driver alertness during a session.

### *C. System Performance and Challenges*

The overall system demonstrated its capability as a practical and effective driver vigilance monitoring solution, providing real-time alerts to mitigate drowsiness-related accident risks.

- Robust Face and Eye Detection: MediaPipe's Face Mesh proved highly effective in accurately detecting faces and eye landmarks across various head movements and lighting conditions commonly encountered in a vehicle. The dynamic eye ROI extraction with adjusted margins successfully captured the relevant eye features for the CNN.
- Accurate Eye State Classification: The trained CNN model consistently and quickly classified eye states, even with the modest 24×24 input size. The combination of CLAHE and normalization prior to inference was crucial for consistent performance.
- Responsive Drowsiness Alert: The drowsiness detection logic, based on consecutive closed frames and time thresholds, effectively filtered out normal blinks and triggered the alarm promptly when sustained eye closure indicative of drowsiness occurred. The audio alarm provided an immediate and noticeable alert.

- Alarm Management: The threaded alarm system ensured that the audio alert played without interrupting the video feed or other processing. The automatic stopping of the alarm upon sustained eye opening enhanced user experience and prevented unnecessary prolonged alerts.
- User Interface Functionality: The Tkinter GUI provided intuitive controls for starting and stopping the system, along with clear status updates. The ability to view logs proved useful for reviewing past detection events.

While the system performs robustly, some challenges were observed:

- Extreme Lighting Variations: Conditions such as direct sunlight or very dark environments could occasionally impact MediaPipe's initial detection accuracy.
- Glasses: Reflective glasses, in particular, could sometimes interfere with precise eye landmark detection, although MediaPipe's refine_landmarks feature helped to mitigate this to some extent.
- Computational Resources: Although designed for efficiency, continuous real-time processing on older hardware might lead to slight frame rate drops.

The model training successfully produced a high-performing CNN classifier for eye state detection, achieving a validation accuracy of approximately 94.5% and an excellent AUC of 0.9894. These measurements show how well the model can differentiate between eyes that are open and those that are closed.

## VI. CONCLUSION AND FUTURE ENHANCEMENTS

This paper presented a robust and real-time Driver Vigilance Monitoring System utilizing deep learning and computer vision for detecting driver drowsiness based on eye state analysis. The system successfully integrates MediaPipe for precise facial landmark detection, a tailored CNN model for eye state classification, and a comprehensive drowsiness detection algorithm with an immediate audio alert system. A thorough assessment revealed the eye state categorization model's excellent accuracy as well as the real-time detection and alerting mechanism's usefulness. The Tkinter-based Graphical User Interface (GUI) enhances usability, making the system accessible and manageable.

The key contributions of this work include:

- Development of an efficient CNN model optimized for real-time eye state classification.
- Integration of MediaPipe for robust and accurate eye region extraction, combined with custom margin adjustments for improved cropping.
- Application of Contrast Limited Adaptive Histogram Equalization (CLAHE) in preprocessing for enhanced contrast in eye images.
- Implementation of a multi-threaded system for non-blocking alarm functionality and GUI responsiveness.
- A clear, configurable drowsiness detection logic incorporating consecutive frame counts and time thresholds to differentiate blinks from prolonged eye closure.

For future work, several enhancements can be explored to further advance the system's capabilities:

- Integration of Additional Indicators: Combining eye state analysis with other drowsiness indicators such as yawning detection, head pose estimation, or more explicit PERCLOS calculation could further improve the system's accuracy and robustness.
- Robustness to Lighting Conditions: Further research into advanced image processing techniques or more robust deep learning architectures could enhance performance under extreme or rapidly changing lighting conditions.
- Embedded System Deployment: Optimizing the model for deployment on edge devices or embedded systems (e.g., Raspberry Pi, NVIDIA Jetson) could enable standalone, low-cost in-car solutions. Model quantization or pruning techniques could be explored for this purpose.
- Driver Identification and Personalization: Developing features for driver identification and personalized drowsiness thresholds based on individual blinking patterns or fatigue levels could enhance user experience and accuracy.
- Feedback Mechanisms: Incorporating haptic feedback (e.g., seat vibration) or visual warnings on a dashboard display in addition to audio alerts could provide a more nuanced warning system.
- Dataset Expansion: Training the model on a larger and more diverse dataset, including images with subjects wearing glasses, varying ethnicities, and different lighting conditions, can significantly improve generalization.

By continuously refining these aspects, the Driver Vigilance Monitoring System has the potential to become an even more indispensable tool in promoting road safety and preventing fatigue-related accidents.

## REFERENCES

[1] National Highway Traffic Safety Administration (NHTSA). (Various Years). *Traffic Safety Facts*. U.S. Department of Transportation. Available: https://www.nhtsa.gov/

[2] E. Picton and E. K. St Louis, "Drowsy Driving: A Systematic Review of Behavioral and Physiological Monitoring Technologies," *Sleep Medicine Reviews*, vol. 51, p. 101289, 2020. doi: 10.1016/j.smrv.2020.101289.

[3] W. W. Wierwille, W. F. Wreggit, and R. J. Fairbanks, "Research on Vehicle-Based Driver Fatigue Detection Systems," *Technical Report*, 1994.

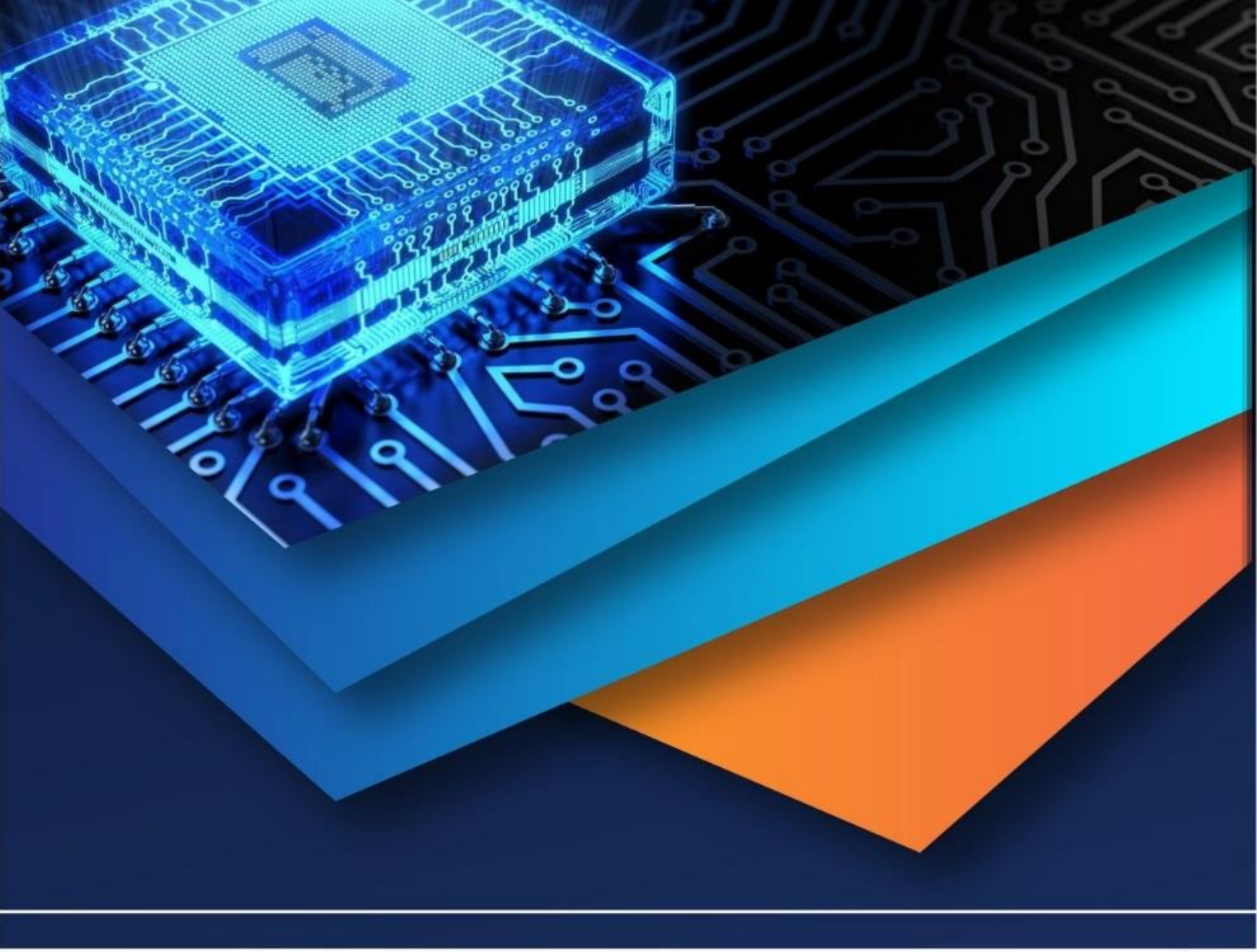
[4] A. Sahayadhas, K. Sundaraj, and S. Murugappan, "Detecting Driver Drowsiness Based on EOG, EEG and ECG Signals: A Review," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1056-1069, 2012. doi: 10.1109/TSMCB.2012.2192736.

[5] H. Summala and H. Mikkola, "Fatal Accidents and Their Drivers in Finland," *Accident Analysis & Prevention*, vol. 26, no. 3, pp. 297-304, 1994. doi: 10.1016/0001-4575(94)90011-8.

[6] A. K. Ranjani and P. Kavitha, "The study of driver fatigue monitor algorithm combined PERCLOS and AECS," in *Proc. 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies (ICACCCT)*, Dec. 2014, pp. 1092-1095. doi: 10.1109/ICACCCT.2014.7032128.

[7] A. Emin, F. G. Sancak, A. Bayar, and F. Cakar, "Deep Learning-Based Driver Assistance System," *ELECTRICA*, vol. 23, no. 3, pp. 607-618, 2023. Available: https://electrica.ieu.edu.tr/en/pub/2023_v23_n3_a17/

[8] C. Yang, Z. Yang, W. Li, and J. See, "An Intelligent Driving Assistance System Based on Lightweight Deep Learning Models," *IEEE Access*, vol. 10, pp. 107062-107071, 2022. doi: 10.1109/ACCESS.2022.3211516.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY