E-ISSN: 1115 – 8492 P-ISSN: 3027 – 0693
MPRI-JNSAR| INTERNATIONAL JOURNAL OF NATURE AND SCIENCE ADVANCE RES.
VOL. 13 NO. 9 JULY-2026

A LIGHTWEIGHT CNN-BASED ANDROID VISION SYSTEM FOR REAL-TIME DRIVER DROWSINESS DETECTION USING EYE-STATE ANALYSIS

***BENISEMENI ESTHER ZAKKA; *FABUNMI ESTHER OMOWUNMI; **GLORIA NGOZI JOLA; & ***SANUSI ABDULHAMID DANTATA**

*Department of Computer Science, Federal Polytechnic Bauchi. **Department of Electrical and Electronics Engineering, Federal Polytechnic Bauchi. ***Department of Cyber Security, Federal Polytechnic Bauchi.

DOI: https://doi.org/10.70382/mejnsar.v13i9.0105

Keywords: Driver drowsiness detection, Convolutional Neural Network (CNN), Android application, Computer vision, Eye-state analysis, Deep learning.

Abstract

Driver drowsiness remains one of the leading causes of road traffic accidents worldwide, as fatigue significantly impairs a driver's alertness, reaction time, and decision-making ability. Existing drowsiness detection approaches often require specialized hardware or computationally intensive models that limit their deployment on resource-constrained mobile devices. This study presents a lightweight CNN-based Android vision system for real-time driver drowsiness detection using eye-state analysis. The proposed system combines Haar Cascade classifiers for real-time face and eye localization with a custom Convolutional Neural Network (CNN) for classifying eye states as open or closed. A score-based decision mechanism continuously monitors eye closure across successive video frames and activates an audible alarm when prolonged eye closure indicates possible driver fatigue. The CNN model was trained using labeled eye-image datasets in Python with the Keras deep learning framework and subsequently integrated into an Android application for real-time inference using a smartphone camera. Experimental evaluation demonstrated

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

that the developed system accurately classified eye states and generated timely drowsiness alerts under real-time operating conditions while maintaining computational efficiency suitable for mobile deployment. The findings demonstrate that lightweight deep learning models can be effectively deployed on Android devices to provide an accessible, low-cost, and practical solution for continuous driver drowsiness monitoring and improved road safety.

Introduction

Driver drowsiness is a significant contributor to road traffic accidents worldwide, posing a major challenge to transportation safety. Fatigue impairs a driver's alertness, judgment, reaction time, and decision-making ability, thereby increasing the likelihood of collisions that may result in injuries, fatalities, and substantial economic losses. According to the World Health Organization (WHO), road traffic accidents remain one of the leading causes of injury and death globally, with human factors such as fatigue contributing substantially to crash occurrence (World Health Organization, 2018). Studies have further reported that driver drowsiness contributes to approximately 20–50% of road crashes worldwide, highlighting the need for reliable mechanisms capable of detecting fatigue before it results in an accident (Abbas, 2020; May & Baldwin, 2009). Consequently, the development of intelligent driver monitoring systems has become an active area of research aimed at improving road safety through early detection and timely intervention.

Various approaches have been proposed for detecting driver drowsiness, broadly categorized into vehicle-based, physiological-based, and behavioral-based methods (Al-Khateeb et al., 2023). Vehicle-based approaches estimate the driver's level of alertness by monitoring driving patterns such as steering wheel movements, lane deviation, and vehicle speed variations. Although these techniques are non-intrusive and can be implemented without directly observing the driver, their performance is often influenced by external factors including road conditions, traffic density, weather conditions, and individual driving habits, which may result in inaccurate detection of fatigue (Abbas, 2020; Chen & Lee, 2020). Consequently, their reliability may decrease under complex real-world driving environments.

Physiological-based methods provide a more direct assessment of a driver's alertness by monitoring biological signals such as electroencephalograms (EEG), electrocardiograms (ECG), and heart rate variability (HRV). These physiological signals exhibit measurable changes as fatigue increases and have therefore been widely employed in driver monitoring research (Ahmed et al., 2021; Patel et al., 2020). Despite their high detection

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

accuracy, physiological approaches require wearable sensors or electrodes attached to the driver's body, making them intrusive, relatively expensive, and less suitable for routine driving applications (Eoh et al., 2005; Santos et al., 2021). Furthermore, the use of specialized sensing equipment reduces user convenience and limits widespread adoption in consumer vehicles.

Behavioral-based approaches have gained considerable attention because they offer a non-intrusive means of monitoring driver alertness using computer vision techniques. These systems analyze facial characteristics such as eye closure, blink frequency, head movement, facial expressions, yawning, and other observable indicators of fatigue from images or video streams (Rahman et al., 2022). Among these indicators, eye-state monitoring has consistently been identified as one of the most reliable measures of driver drowsiness because prolonged eye closure is strongly associated with fatigue-related impairment (Smith et al., 2021). However, behavioral approaches remain susceptible to environmental challenges such as varying illumination, facial occlusions caused by spectacles or face masks, and changes in head orientation, all of which may reduce detection accuracy under real-world driving conditions (Liu et al., 2019; Ahmed & Khan, 2022).

Recent advances in deep learning and computer vision have significantly improved the performance of vision-based driver monitoring systems. In particular, Convolutional Neural Networks (CNNs) have demonstrated remarkable capabilities in automatic feature extraction and image classification, enabling accurate recognition of facial features associated with driver fatigue (Géron, 2022). Recent studies have shown that CNN-based models consistently outperform traditional handcrafted feature-based approaches while maintaining high detection accuracy across diverse datasets (Jazzar et al., 2021; Rundo et al., 2023). At the same time, the rapid advancement of smartphone technology has created new opportunities for deploying intelligent vision systems on mobile devices. Modern smartphones are equipped with high-resolution cameras and increasingly powerful processors, making them suitable platforms for implementing lightweight deep learning applications without requiring dedicated in-vehicle hardware (Qu et al., 2023). Consequently, mobile-based driver monitoring systems provide an affordable, portable, and accessible alternative to conventional embedded solutions.

Despite these advancements, several challenges remain. Many existing driver drowsiness detection systems rely on computationally intensive models or specialized hardware that increase implementation cost and reduce portability. Other approaches require additional sensors that are inconvenient for everyday use or perform poorly under unconstrained driving environments. Furthermore, lightweight mobile implementations often experience performance degradation due to limited computational resources and varying

environmental conditions (Liu et al., 2024). These limitations create the need for efficient computer vision systems capable of providing reliable real-time performance while operating on commonly available Android devices.

This study addresses these challenges by developing a lightweight CNN-based Android vision system for real-time driver drowsiness detection using eye-state analysis. The proposed system employs Haar Cascade classifiers to detect facial and eye regions from live camera images, while a custom Convolutional Neural Network classifies each detected eye as either open or closed. A score-based decision mechanism continuously evaluates eye closure across successive frames and activates an audible alert whenever prolonged eye closure indicates possible driver drowsiness. The trained CNN model was integrated into an Android application using TensorFlow Lite, enabling real-time inference on a smartphone without requiring specialized monitoring equipment.

The major contribution of this study is the development and evaluation of a practical, low-cost, and non-intrusive driver drowsiness detection system that combines lightweight computer vision techniques with deep learning for deployment on Android mobile devices. By integrating efficient face and eye detection with CNN-based eye-state classification and a real-time alert mechanism, the proposed system demonstrates the feasibility of using smartphone technology to enhance driver safety through continuous fatigue monitoring.

Related Work

Driver drowsiness detection has attracted significant research interest over the past two decades because fatigue remains one of the major contributors to road traffic accidents worldwide. Numerous approaches have been proposed to monitor driver alertness using vehicle dynamics, physiological signals, computer vision, and hybrid intelligent systems. Each category offers unique advantages while presenting limitations that influence its effectiveness in practical driving environments (Al-Khateeb et al., 2023; May & Baldwin, 2009).

Vehicle-Based Approaches

Vehicle-based drowsiness detection methods estimate driver alertness by analyzing driving behavior, including steering wheel movement, lane deviation, braking patterns, vehicle speed, and acceleration. These techniques are attractive because they are non-intrusive and rely primarily on sensors already integrated into modern vehicles (Abbas, 2020). Changes in steering behavior and lane-keeping performance have been shown to correlate with increasing driver fatigue, making them useful indicators for fatigue detection.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Despite these advantages, vehicle-based approaches suffer from several limitations. Driving behavior is strongly influenced by external factors such as road geometry, weather conditions, traffic congestion, and individual driving styles, making it difficult to distinguish fatigue-related behavior from normal driving variations (Chen & Lee, 2020). Consequently, these systems may generate false alarms or fail to detect fatigue under complex real-world conditions. Because of these limitations, vehicle-based methods are increasingly being combined with driver-centered monitoring techniques to improve reliability.

Physiological-Based Approaches

Physiological approaches monitor biological signals that change as fatigue develops. Commonly used indicators include electroencephalography (EEG), electrocardiography (ECG), electromyography (EMG), and heart rate variability (HRV). Since these signals directly reflect the driver's physiological and cognitive state, they generally provide higher detection accuracy than indirect observation methods (Ahmed et al., 2021; Patel et al., 2020).

Several studies have demonstrated that EEG-based systems can detect fatigue at an early stage before noticeable behavioral changes occur (Eoh et al., 2005; Santos et al., 2021). However, these approaches require wearable sensors or electrodes attached to the driver's body, increasing system cost and reducing user comfort during prolonged driving. In addition, physiological monitoring systems often require periodic calibration and specialized hardware, limiting their suitability for widespread deployment in consumer vehicles.

Behavioral and Vision-Based Approaches

Behavioral approaches detect driver fatigue by analyzing observable facial characteristics using computer vision techniques. Typical indicators include eye closure, blink frequency, yawning, gaze direction, head pose, and facial expressions (Rahman et al., 2022). Among these indicators, eye-state monitoring has consistently been recognized as one of the most reliable measures of driver drowsiness because prolonged eye closure is highly correlated with fatigue (Smith et al., 2021).

Compared with physiological methods, vision-based systems are non-intrusive and require only a camera for continuous monitoring. However, their performance remains sensitive to illumination changes, facial occlusions, camera positioning, and head orientation (Ahmed & Khan, 2022; Liu et al., 2019). Traditional image-processing methods based on handcrafted features often struggle under these challenging conditions, motivating the adoption of deep learning techniques for improved robustness.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Deep Learning-Based Driver Drowsiness Detection

Recent advances in deep learning have transformed driver monitoring by enabling automatic extraction of discriminative visual features directly from facial images. Convolutional Neural Networks (CNNs) have demonstrated superior performance over conventional machine learning algorithms in image classification tasks because they learn hierarchical feature representations without manual feature engineering (Géron, 2022).

Several studies have successfully applied CNN architectures to eye-state classification and driver monitoring, reporting substantial improvements in detection accuracy (Jazzar et al., 2021; Zhang et al., 2022). More recent approaches combine CNNs with Long Short-Term Memory (LSTM) or Recurrent Neural Networks (RNNs) to capture temporal dependencies in driver behavior and further improve fatigue detection performance (Rundo et al., 2023). Although these hybrid models often achieve excellent accuracy, they require considerable computational resources, larger training datasets, and longer inference times, making them less suitable for deployment on resource-constrained mobile devices (Liu et al., 2024). Consequently, there is growing interest in lightweight CNN architectures that balance classification accuracy with computational efficiency, enabling real-time inference on embedded and mobile platforms.

Mobile-Based Driver Monitoring Systems

The rapid advancement of smartphone hardware has encouraged researchers to investigate mobile-based driver monitoring systems capable of performing real-time computer vision tasks. Modern Android devices provide high-resolution cameras, multi-core processors, and optimized inference engines such as TensorFlow Lite, making them suitable platforms for lightweight deep learning applications (Qu et al., 2023).

Compared with dedicated embedded systems, smartphone-based implementations offer several practical advantages, including lower deployment cost, portability, ease of installation, and wider accessibility. Nevertheless, mobile deployment introduces significant computational constraints related to processor speed, memory availability, battery consumption, and inference latency. Therefore, models intended for Android deployment must achieve an appropriate trade-off between computational efficiency and classification accuracy.

Problem Formulation

Driver drowsiness detection is formulated as a supervised binary image classification problem in which the objective is to determine whether a driver's eyes are open or closed using facial images captured by a smartphone camera. The eye state serves as an important indicator of driver alertness, and prolonged eye closure is considered a strong

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

symptom of drowsiness. Therefore, the task of the proposed system is to accurately classify each detected eye image into one of two predefined classes and generate an alert whenever sustained eye closure is observed.

Let the labelled eye-image dataset be represented by $Equa$tion (1).

$$D = \{(x_i, y_i)\}, i = 1,2,..,N$$

where D denotes the labelled dataset containing N eye images, x_i represents the *i*th input eye image, and y_i denotes the corresponding class label. In this study, the class labels consist of two categories:

- Open Eye
- Closed Eye

The objective of the learning process is to train a Convolutional Neural Network (CNN) capable of learning the relationship between an input eye image and its corresponding eye-state class. During real-time operation, each eye region extracted from the live video stream is passed to the trained CNN model, which predicts whether the driver's eye is open or closed.

Unlike conventional rule-based approaches that depend on manually selected thresholds or handcrafted visual features, the proposed CNN automatically learns discriminative features directly from the training data. This enables the model to adapt to variations in illumination, facial appearance, eye shape, and viewing angle, thereby improving its robustness under real-world driving conditions.

To improve the reliability of drowsiness detection, the proposed system does not trigger an alarm based on a single prediction. Instead, consecutive eye-state predictions are analysed over time using a cumulative drowsiness score. An audible warning is generated only when the accumulated score exceeds a predefined threshold, thereby reducing false alarms caused by normal blinking while ensuring timely detection of prolonged eye closure associated with driver fatigue.

Proposed System Framework

The proposed driver drowsiness detection system integrates computer vision, deep learning, and mobile computing into a unified framework for real-time driver monitoring. The system is designed to operate entirely on an Android smartphone, thereby eliminating the need for specialized hardware such as infrared cameras, wearable sensors, or vehicle-mounted monitoring devices. The framework consists of two major

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

phases: offline model development and online real-time deployment, as illustrated in Figure 1.

During the offline phase, a labelled eye-image dataset is prepared and preprocessed to ensure consistency in image quality and dimensions. The processed images are then used to train a lightweight Convolutional Neural Network (CNN) capable of classifying eye states into two categories: Open Eye and Closed Eye. Following training and validation, the model is converted into a mobile-compatible format and integrated into an Android application for real-time inference.

The online deployment phase begins when the smartphone camera continuously captures video frames of the driver's face. Each frame is processed using a Haar Cascade classifier to detect the face and localize the eye regions. The detected eye images are preprocessed using the same procedures applied during model training before being passed to the trained CNN for classification.

Rather than relying on a single prediction to determine driver drowsiness, the proposed system continuously evaluates consecutive eye-state predictions over time. A cumulative drowsiness score is maintained to distinguish normal blinking from prolonged eye closure. When the score exceeds a predefined threshold, the application immediately activates an audible warning alarm to alert the driver and encourage corrective action.

The overall processing sequence of the proposed framework is summarized as follows: Image Acquisition → Face Detection → Eye Detection → Image Preprocessing → CNN-Based Eye State Classification → Drowsiness Assessment → Alarm Generation

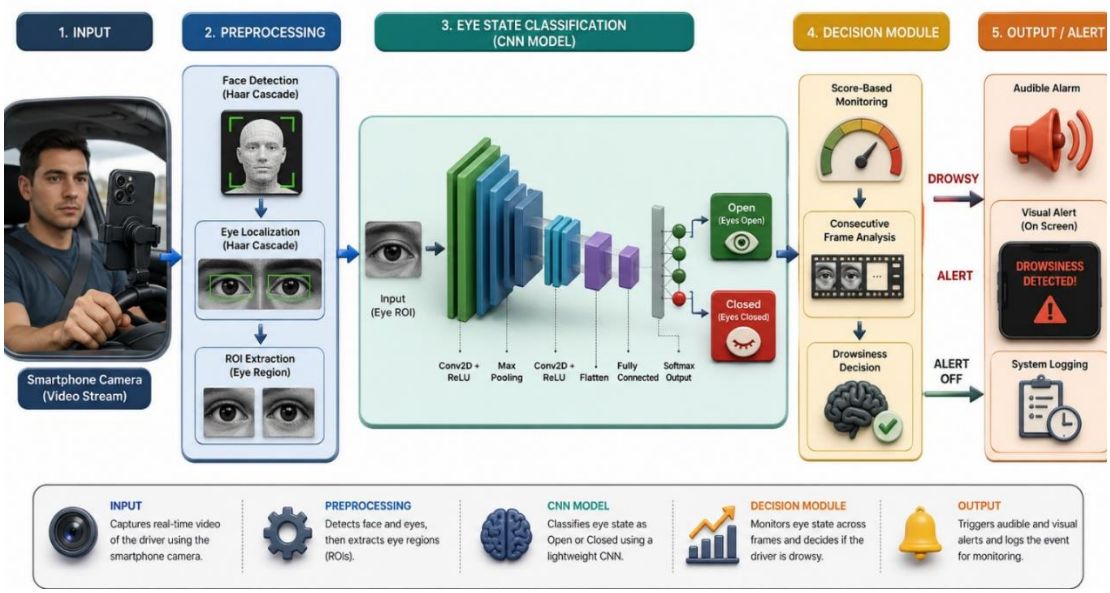

Figure 1 Proposed Mobile Drowsiness Framework

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

This framework enables continuous, real-time driver monitoring while maintaining low computational overhead, making it suitable for deployment on resource-constrained Android smartphones. The proposed framework was designed to balance classification accuracy with computational efficiency by combining lightweight face detection and CNN-based eye-state classification, thereby enabling reliable real-time inference on Android smartphones without dependence on cloud computing or dedicated hardware.

Image Acquisition and Dataset Preparation

The performance of the proposed driver drowsiness detection system depends largely on the quality and diversity of the training data. Consequently, a labelled image dataset containing facial images representing different driver conditions was prepared for model development. The dataset included images corresponding to eye-state categories used for drowsiness detection and was organized into separate class directories to facilitate supervised learning using the TensorFlow/Keras framework.

Before training, all images were resized to 145 × 145 pixels with three colour channels (RGB) to ensure a uniform input size for the convolutional neural network. Standardizing the image dimensions reduced computational complexity and ensured consistency during feature extraction and classification.

To improve the generalization capability of the model, the dataset was randomly partitioned into training and testing subsets using a 70:30 ratio, where 70% of the images were used for model training and the remaining 30% were reserved for performance evaluation. This separation ensured that the trained model was evaluated using previously unseen images, thereby providing an unbiased estimate of its predictive performance.

Data augmentation techniques were applied to the training images to increase dataset diversity and reduce the likelihood of overfitting. Specifically, the images were rescaled by a factor of 1/255, randomly rotated by up to 30°, horizontally flipped, and subjected to a 20% zoom transformation. These augmentation operations enabled the model to learn robust visual features under varying illumination conditions, facial orientations, and camera perspectives commonly encountered during real-world driving.

Following preprocessing and augmentation, the prepared dataset was supplied to the CNN for supervised training and validation. The dataset preparation workflow adopted in this study contributed significantly to improving the robustness and generalization capability of the developed driver drowsiness detection model. Table 1 summarizes the preprocessing parameters used during dataset preparation.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Table 5 Summary of preprocessing parameters used during dataset preparation

Parameter	Value
Image Size	145 × 145 × 3
Training–Testing Split	70% : 30%
Image Rescaling	1/255
Rotation Range	30°
Horizontal Flip	Enabled
Zoom Range	0.2

Image Preprocessing

Image preprocessing is a critical stage in deep learning because it enhances data quality and ensures that all input images conform to the requirements of the convolutional neural network (CNN). In this study, preprocessing was performed prior to model training and was consistently applied during real-time deployment to maintain uniformity between the training and inference phases.

Initially, all input images were resized to 145 × 145 pixels to provide a fixed input dimension for the CNN. Standardizing image dimensions ensures consistent feature extraction while reducing computational complexity during model training and inference. To improve numerical stability during optimization, pixel intensity values were normalized by rescaling each pixel to the range of 0–1 using Equation (2).

Equation (2)

$$I_normalized = \frac{1}{255}$$

where I represents the original pixel intensity and I_normalized denotes the normalized pixel value. This normalization prevents large pixel values from dominating the learning process and contributes to faster model convergence.

To improve the robustness of the trained model, several data augmentation techniques were applied to the training images. These included random image rotation of up to 30°, horizontal flipping, and random zooming with a zoom factor of 0.2. Data augmentation artificially increases the diversity of the training data, enabling the model to generalize better to variations in head orientation, facial appearance, and camera positioning encountered during real-world driving.

During real-time operation, video frames captured by the smartphone camera undergo the same preprocessing sequence before classification. The driver's face is first detected using the Haar Cascade algorithm, after which the eye regions are extracted as Regions

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

of Interest (ROIs). The extracted eye images are resized and normalized before being supplied to the trained CNN model for eye-state classification. Applying identical preprocessing procedures during both training and deployment minimizes data distribution differences and improves prediction consistency. The preprocessing pipeline adopted in this study is illustrated in Figure 2.

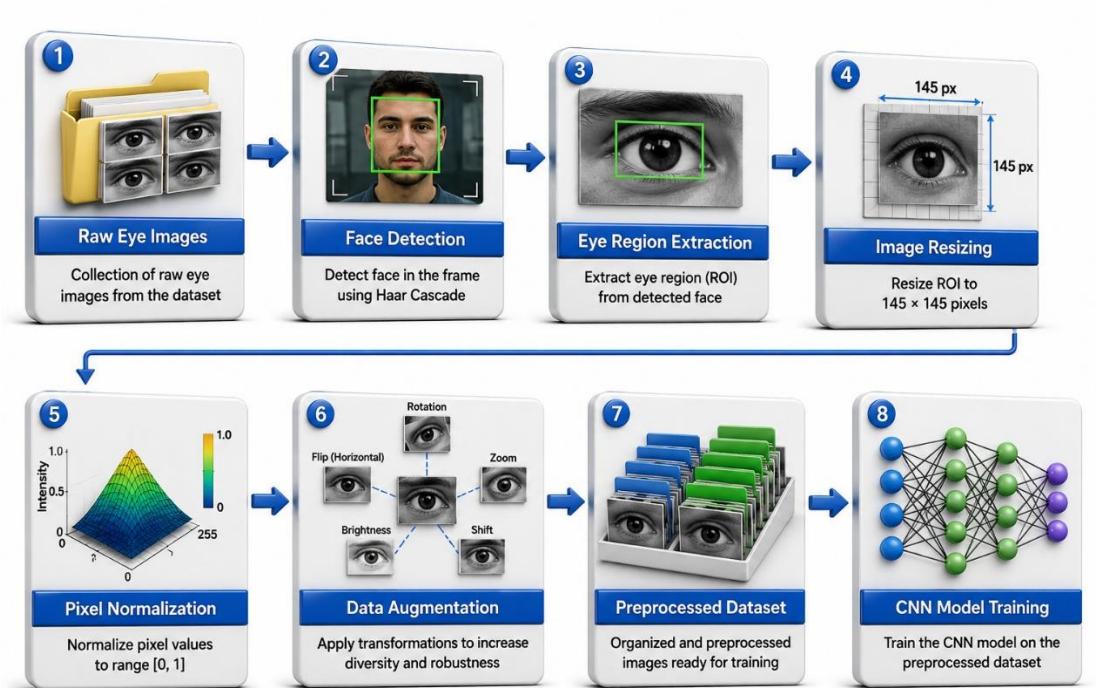

Figure 2 Image Preprocessing Pipeline

Convolutional Neural Network Architecture

A Convolutional Neural Network (CNN) was adopted as the classification model due to its ability to automatically learn hierarchical visual features from images. Unlike traditional machine learning algorithms that rely on manually engineered features, CNNs extract relevant spatial patterns directly from the input images through successive convolution and pooling operations. This capability makes CNNs particularly suitable for image classification tasks such as eye-state recognition for driver drowsiness detection.

The proposed CNN was implemented using the TensorFlow and Keras deep learning frameworks. The network accepts RGB images of size 145 × 145 × 3 as input and processes them through a sequence of convolutional and pooling layers to learn discriminative image features. Four convolutional layers with 256, 128, 64, and 32 filters, respectively, were employed. Each convolutional layer was followed by a max-pooling layer, which progressively reduced the spatial dimensions of the feature maps while preserving the

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

most important features. This combination of convolution and pooling enabled the network to capture both low-level features, such as edges and textures, and higher-level facial characteristics relevant to eye-state recognition.

Following feature extraction, the resulting feature maps were flattened into a one-dimensional vector and passed through a fully connected layer consisting of 64 neurons. A dropout layer with a dropout rate of 0.5 was incorporated before the fully connected layer to reduce overfitting by randomly deactivating neurons during training. The final output layer employed the Softmax activation function to produce class probabilities for the target categories.

The Rectified Linear Unit (ReLU) activation function was used in the hidden layers because of its computational efficiency and its ability to mitigate the vanishing gradient problem commonly encountered in deep neural networks. The Softmax activation function was used in the output layer to generate normalized probability values corresponding to each output class.

The convolution operation performed by the CNN is expressed by Equation (3).

Equation (3)

$$Feature\ Map = Input\ Image * Kernel + Bias$$

where the symbol (*) denotes the convolution operation, Kernel represents the learnable filter, and Bias is the bias term associated with each filter. During training, the convolution kernels are automatically updated to learn image features that are most useful for distinguishing between different eye states. Table 2 summarizes the architecture of the developed CNN model.

Table 6 Architecture of the developed CNN model.

Layer	Configuration
Input Layer	145 × 145 × 3 RGB Image
Convolution Layer 1	256 Filters
Max-Pooling Layer 1	2 × 2
Convolution Layer 2	128 Filters
Max-Pooling Layer 2	2 × 2
Convolution Layer 3	64 Filters
Max-Pooling Layer 3	2 × 2
Convolution Layer 4	32 Filters
Max-Pooling Layer 4	2 × 2
Flatten Layer	Feature Vector
Dropout Layer	0.5
Fully Connected Layer	64 Neurons
Output Layer	Softmax

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

The adopted architecture provides a balance between classification performance and computational efficiency, making it suitable for deployment on Android smartphones where memory, processing power, and battery capacity are limited. By progressively extracting discriminative visual features while maintaining a relatively lightweight network structure, the proposed CNN supports real-time eye-state classification without requiring cloud-based computation or specialized hardware.

Model Training

The proposed Convolutional Neural Network (CNN) was trained using the TensorFlow and Keras deep learning frameworks within the Python programming environment. Prior to training, the prepared dataset was divided into training and testing subsets using a 70:30 ratio, with 70% of the images used for learning the network parameters and the remaining 30% reserved for model evaluation.

During training, the Adam optimization algorithm was employed to update the network weights because of its adaptive learning capability and efficient convergence properties. The model was trained for 50 epochs, allowing the network to iteratively learn discriminative visual features from the eye-image dataset while minimizing the classification error. The Categorical Cross-Entropy loss function was adopted because the model performs multi-class classification. The loss function measures the difference between the predicted class probabilities and the true class labels, thereby guiding the optimization process during training. It is expressed as:

Equation (4)

$$L = -\sum y_i \log \hat{y}_i$$

where:

- L is the classification loss;
- y_i is the actual class label; and
- $\hat{y}_i$ is the predicted probability for the corresponding class.

During each training epoch, the model parameters were updated automatically through backpropagation to minimize the loss function. The training process continued until all epochs were completed, after which the trained model was saved for deployment within the Android application.

To reduce overfitting and improve the model's ability to generalize to unseen data, a dropout rate of 0.5 was applied before the fully connected layer. In addition, data augmentation techniques, including image rotation, horizontal flipping, zooming, and pixel normalization, were incorporated into the training pipeline. These techniques

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

increased the diversity of the training data and improved the robustness of the model under varying environmental conditions.

The performance of the trained model was monitored throughout the training process using both training and validation accuracy and loss. These metrics provided an indication of the learning progress and helped verify that the model was converging appropriately without significant overfitting. Table 3 summarizes the training parameters used in this study.

Table 7 Training Parameters

Parameter	Value
Deep Learning Framework	TensorFlow/Keras
Programming Language	Python
Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Number of Epochs	50
Image Size	145 × 145 × 3
Training–Testing Split	70% : 30%
Dropout Rate	0.5
Data Augmentation	Rotation, Horizontal Flip, Zoom, Rescaling

Mobile Application Integration and Real-Time Deployment

To demonstrate the practical applicability of the proposed driver drowsiness detection system, the trained Convolutional Neural Network (CNN) model was integrated into an Android mobile application capable of performing real-time driver monitoring. Deploying the model on a smartphone eliminates the need for dedicated hardware, making the system portable, cost-effective, and suitable for everyday use.

The mobile application was developed using Android Studio, while the deep learning model was trained using the TensorFlow/Keras framework in Python. After the training process was completed, the trained model was converted into a mobile-compatible format and incorporated into the Android application to enable on-device inference. This approach allows all computations to be performed locally on the smartphone without requiring cloud connectivity, thereby reducing latency and preserving user privacy. During operation, the application continuously captures live video frames using the smartphone's front-facing camera. Each frame is processed to detect the driver's face, after which the eye regions are extracted as Regions of Interest (ROIs). The extracted eye images undergo the same preprocessing operations applied during model training before being passed to the CNN for classification.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

The predicted eye-state information is evaluated continuously to determine the driver's level of alertness. Rather than responding to a single prediction, the application analyses consecutive predictions over time to distinguish normal blinking from prolonged eye closure. When the accumulated evidence indicates that the driver is becoming drowsy, the application immediately activates an audible warning alarm to alert the driver and encourage corrective action. The developed application was tested on an Android smartphone under real-time operating conditions. Experimental evaluation demonstrated that the application was capable of processing live camera input, detecting the driver's facial features, classifying eye states, and generating warning alerts with minimal delay. These results confirm the feasibility of deploying lightweight deep learning models on resource-constrained mobile devices for real-time driver assistance applications. The overall deployment architecture of the proposed system is illustrated in Figure 3.



Figure 3 Deployment Framework

Performance Evaluation

The performance of the proposed driver drowsiness detection system was evaluated using standard classification metrics commonly employed in deep learning and computer vision research. The trained CNN model was tested using previously unseen images from the testing dataset to assess its ability to correctly classify eye states.

A confusion matrix was used to summarize the classification results by comparing the predicted eye-state labels with the corresponding ground-truth labels. Based on the confusion matrix, four evaluation metrics were computed, namely accuracy, precision, recall, and F1-score. These metrics provide complementary information regarding the classification performance of the developed model. Classification

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

accuracy measures the proportion of correctly classified samples relative to the total number of test samples and is computed as

Equation (5)

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

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Precision measures the proportion of correctly predicted positive samples among all positive predictions and is calculated as

Equation (6)

$$Precision = {}^{TP}/_{(TP + FP)}$$

Recall measures the ability of the model to correctly identify positive samples and is given by

Equation (7)

$$Recall = {}^{TP}/_{(TP + FN)}$$

The harmonic mean of Precision and Recall is represented by the F1-score and is calculated as

Equation (8)

$$F1\text{-}score = 2 \times (Precision \times Recall)/(Precision + Recall)$$

In addition to these quantitative metrics, the trained model was deployed within an Android application and evaluated under real-time operating conditions. Live testing was conducted using the smartphone's front-facing camera to verify the ability of the application to continuously detect facial features, classify eye states, and generate warning alerts whenever prolonged eye closure was detected. This practical evaluation complements the quantitative performance metrics by demonstrating the feasibility of implementing the proposed system on a resource-constrained mobile device.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Experimental Setup

The experiments were conducted to evaluate the effectiveness of the proposed CNN-based driver drowsiness detection system under both offline and real-time operating conditions. Model development and training were carried out using the Python programming language with the TensorFlow and Keras deep learning libraries. The trained model was subsequently integrated into an Android application for real-time deployment and evaluation.

The image dataset was preprocessed and partitioned into training and testing subsets using a 70:30 ratio. Data augmentation techniques, including image rescaling, rotation, horizontal flipping, and zooming, were applied during training to improve model generalization. The CNN was trained for 50 epochs using the Adam optimization algorithm with the Categorical Cross-Entropy loss function.

After training, the model was exported and integrated into the Android application. Real-time experiments were performed using the smartphone's front-facing camera to evaluate the system's ability to continuously detect facial features, classify eye states, and generate warning alarms whenever signs of driver drowsiness were observed. Table 4 summarizes the experimental environment used in this study.

Table 8 The Experimental Environment

Component	Specification
Programming Language	Python
Deep Learning Framework	TensorFlow / Keras
Development IDE	Jupyter Notebook
Mobile Development IDE	Android Studio
Model Optimizer	Adam
Loss Function	Categorical Cross-Entropy
Training Epochs	50
Image Size	145 × 145 × 3
Data Split	70% Training / 30% Testing
Mobile Platform	Android Smartphone

Model Training Performance

The performance of the proposed Convolutional Neural Network (CNN) was monitored throughout the training process to evaluate its learning behaviour and convergence characteristics. During training, both the training accuracy and validation accuracy were recorded after each epoch, while the corresponding training and validation loss values were simultaneously monitored. These metrics provide insight into the model's ability to

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

learn meaningful features from the training dataset and generalize to previously unseen samples.

The CNN was trained for 50 epochs using the Adam optimization algorithm and the Categorical Cross-Entropy loss function. Throughout the training process, the optimization algorithm iteratively updated the network weights through backpropagation to minimize the classification error. Data augmentation techniques, including image normalization, rotation, horizontal flipping, and zooming, contributed to improving the model's generalization capability and reducing overfitting.

Figure 4 presents the training and validation accuracy curves obtained during model training. The figure illustrates the progressive improvement in classification accuracy as the number of training epochs increased. The convergence of the training and validation curves indicates that the network successfully learned representative visual features while maintaining stable generalization performance.

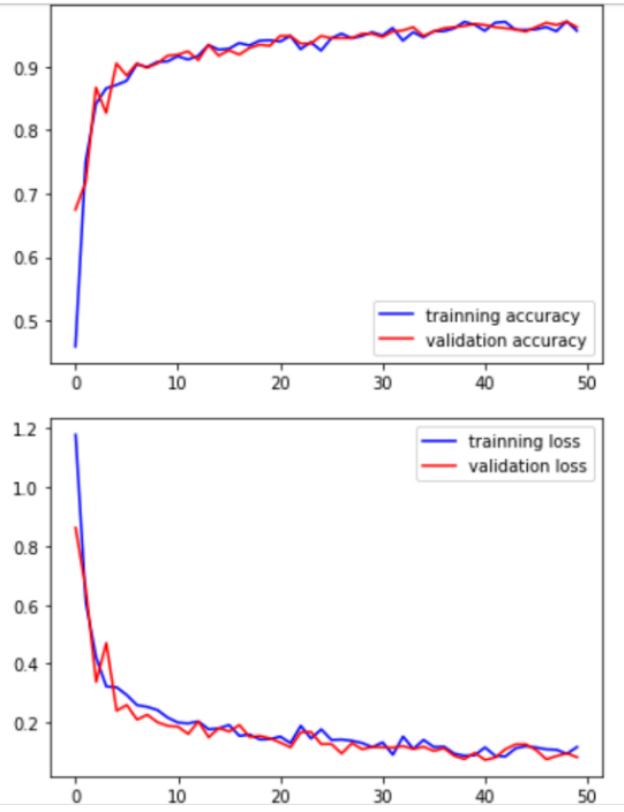

Figure 4 Training and Validation Accuracy Curves

Conclusion and Future Work

This study presented a hybrid CNN–GRU framework for crop yield prediction using agricultural and climatic data. The proposed approach integrated convolutional neural networks for feature extraction with gated recurrent units for learning complex relationships among the input variables. A comprehensive preprocessing pipeline involving label encoding, feature standardization, and data partitioning was implemented to prepare the dataset for model training.

Experimental evaluation demonstrated that the proposed model achieved a coefficient of determination (R2) of 0.8919, indicating that approximately 89.19% of the variation in crop yield was successfully explained by the model. The corresponding Mean Squared

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Error (MSE) further confirmed the effectiveness of the proposed framework in producing reliable crop yield predictions. Analysis of the training and validation curves showed stable convergence without significant evidence of overfitting, demonstrating good generalization performance.

To demonstrate the practical applicability of the developed model, the trained CNN–GRU network was deployed as an interactive Streamlit-based web application capable of providing real-time crop yield predictions from user-supplied agricultural and climatic inputs. This deployment illustrates the potential of the proposed framework as a practical decision-support tool for farmers, agricultural planners, and other stakeholders.

Future research may investigate the integration of larger and more diverse datasets collected from multiple geographical regions, incorporation of additional environmental variables such as satellite imagery and weather forecasts, and comparative evaluation of the proposed CNN–GRU model with more recent deep learning architectures, including transformer-based models, to further improve prediction accuracy and generalization.

Classification Performance

The classification performance of the proposed Convolutional Neural Network (CNN) was evaluated using the testing dataset after completion of the training process. Model performance was assessed using standard evaluation metrics, including precision, recall, F1-score, and overall classification accuracy. These metrics provide a comprehensive assessment of the model's ability to correctly classify the different driver states. Table 5 presents the classification report obtained from the trained model.

Table 9 Classification Performance of the Proposed CNN

Class	Precision	Recall	F1-Score	Support
Yawn	0.63	0.95	0.76	63
No Yawn	0.93	0.54	0.68	74
Closed Eye	0.94	0.98	0.96	215
Open Eye	0.98	0.94	0.96	226
Overall Accuracy			0.90	578

The proposed CNN achieved an overall classification accuracy of 90% on the testing dataset, demonstrating its capability to reliably distinguish between the different driver states. The high accuracy indicates that the adopted network architecture successfully learned discriminative visual features from the training images and generalized well to previously unseen samples.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Among the four classes, the Closed Eye and Open Eye categories recorded the highest classification performance, both achieving an F1-score of 0.96. These results indicate that the proposed model accurately identified eye states, which constitute the primary indicators used for driver drowsiness detection. The high recall value of 0.98 for the Closed Eye class further demonstrates the model's effectiveness in detecting prolonged eye closure, thereby reducing the likelihood of missed drowsiness events.

The Yawn class achieved a recall of 0.95, indicating that the model successfully detected most yawning instances. However, its precision was comparatively lower (0.63), suggesting that some non-yawning facial expressions were incorrectly classified as yawning. Similarly, the No Yawn class recorded a precision of 0.93 but a relatively lower recall of 0.54, implying that certain no-yawn samples were misclassified as yawning. These observations indicate that yawning-related facial expressions exhibit greater visual similarity than eye-state features, making them more challenging to discriminate accurately.

Overall, the experimental results demonstrate that the proposed CNN provides highly reliable eye-state classification while maintaining satisfactory performance for yawning detection. Since the Android application primarily relies on eye-state information to determine driver alertness, the strong performance obtained for the Open Eye and Closed Eye classes supports the suitability of the developed model for real-time driver drowsiness detection.

Mobile Application Deployment

Following successful training and evaluation of the CNN model, the trained network was integrated into an Android mobile application to enable real-time driver drowsiness detection. The application performs on-device inference using the smartphone's front-facing camera, thereby eliminating the need for specialized hardware or cloud-based processing.

During operation, live video frames are continuously acquired from the camera and processed to detect the driver's face and facial landmarks. The detected facial region is analysed to estimate the driver's eye state, after which the application evaluates the driver's level of alertness. When the accumulated evidence indicates drowsiness, an audible warning alarm is triggered to alert the driver.

Figure 5 illustrates the developed application during real-time operation. The application successfully detects the driver's face, identifies facial landmarks, estimates eye-state probabilities, and provides real-time performance information, including frame rate, detector latency, and alarm count. These outputs demonstrate that the proposed system

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

performs continuous monitoring while maintaining the responsiveness required for practical deployment on an Android smartphone.

The deployment results confirm that the proposed framework is capable of executing real-time inference on a resource-constrained mobile device. By combining efficient facial detection with CNN-based eye-state classification, the application provides timely feedback without relying on external sensors or network connectivity. This demonstrates the feasibility of using lightweight deep learning models to support intelligent driver assistance systems in practical driving environments.

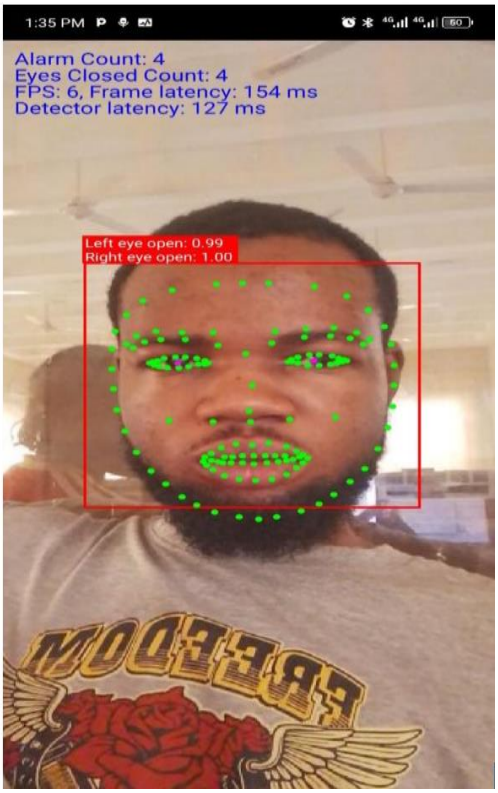

Figure 5 Real-time deployment of the proposed driver drowsiness detection system on an Android smartphone.

Discussion

The experimental results demonstrate that the proposed CNN-based driver drowsiness detection system is capable of accurately recognizing driver facial states while supporting real-time deployment on an Android smartphone. The model achieved an overall classification accuracy of 90%, indicating that the learned visual features were sufficiently discriminative for reliable driver state recognition.

Among the evaluated classes, Closed Eye and Open Eye achieved the highest classification performance, with F1-scores of 0.96 each. These results are particularly significant because eye closure is one of the most reliable physiological indicators of driver drowsiness. The high precision and recall obtained for these classes indicate that the proposed CNN effectively distinguishes between alert and fatigued eye conditions, thereby improving the reliability of the warning system.

Although the model also demonstrated good performance in detecting yawning, the Yawn and No Yawn classes recorded lower F1-scores compared with the eye-state classes. This reduction in performance may be attributed to the greater visual similarity between yawning and non-yawning facial expressions, variations in mouth opening, facial orientation, illumination conditions, and individual differences among subjects.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Nevertheless, the model maintained satisfactory overall performance, demonstrating its ability to capture multiple facial cues associated with driver fatigue.

A major contribution of this study lies in the successful deployment of the trained CNN model on an Android smartphone. Unlike many existing driver monitoring systems that rely on high-performance computers, infrared cameras, wearable sensors, or cloud-based processing, the proposed system performs inference directly on the mobile device. Real-time testing confirmed that the application was capable of continuously detecting facial landmarks, estimating eye states, and generating warning alerts with acceptable processing latency. This demonstrates the feasibility of implementing lightweight deep learning models on resource-constrained mobile devices for intelligent transportation applications.

Compared with traditional rule-based approaches, which depend on manually selected thresholds for eye closure duration, the proposed CNN automatically learns discriminative image features from training data. This data-driven approach improves adaptability to variations in facial appearance and environmental conditions while reducing the need for handcrafted feature engineering.

Despite these encouraging results, the study has certain limitations. The model was trained and evaluated using a limited image dataset and was tested on a single Android platform. Environmental factors such as poor illumination, extreme head rotations, occlusions caused by sunglasses, and camera positioning may affect detection accuracy under real-world driving conditions. Future research should therefore investigate larger and more diverse datasets, optimize the network architecture for higher inference speed, and evaluate the application across multiple smartphone models and driving environments.

Overall, the experimental findings confirm that the proposed system provides an effective, portable, and low-cost solution for real-time driver drowsiness detection. The combination of deep learning and mobile computing demonstrates significant potential for enhancing road safety by providing early warnings that can help reduce fatigue-related road accidents.

Conclusion

This study presented a Convolutional Neural Network (CNN)-based approach for real-time driver drowsiness detection using an Android smartphone. The proposed system integrates computer vision, deep learning, and mobile computing to monitor the driver's facial features and identify signs of drowsiness without requiring specialized hardware or cloud-based processing. By utilizing the smartphone's front-facing camera, the developed

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

application provides a portable and cost-effective solution for continuous driver monitoring.

Experimental evaluation demonstrated the effectiveness of the proposed approach. The developed CNN achieved an overall classification accuracy of 90%, with particularly high performance in recognizing Open Eye and Closed Eye states, which are critical indicators of driver alertness. The trained model was successfully integrated into an Android application and validated under real-time operating conditions, confirming its ability to perform live facial detection, eye-state classification, and timely warning generation on a resource-constrained mobile device.

The findings of this study demonstrate that lightweight deep learning models can be effectively deployed on smartphones to support intelligent driver assistance systems. The proposed solution offers a practical alternative to conventional hardware-dependent driver monitoring systems by providing an affordable, portable, and easily deployable platform for fatigue detection.

Future research should focus on improving the robustness of the system by incorporating larger and more diverse datasets, enhancing detection under challenging environmental conditions such as low illumination and facial occlusions, and optimizing the model for higher processing speed and lower energy consumption. Additional investigations may also explore the integration of other physiological and behavioural indicators, such as head pose estimation and blink-rate analysis, to further improve detection accuracy and reliability.

References

Abbas, M. (2020). Vehicle-based driver fatigue detection systems. *Journal of Intelligent Transportation Systems, 14*(3), 112–121.

Ahmed, M., & Khan, R. (2022). Challenges in vision-based drowsiness detection systems: Lighting and occlusion issues. *International Journal of Computer Vision and Applications, 8*(2), 87–99.

Ahmed, M., Khan, R., & Liu, Y. (2021). Physiological monitoring for driver fatigue detection: A comparative study. *Journal of Advanced Intelligent Systems, 5*(4), 211–228.

Al-Khateeb, H., Rana, O., & Hill, R. (2023). A review of intelligent driver monitoring systems using machine learning techniques. *IEEE Transactions on Intelligent Vehicles, 8*(1), 67–79.

Chen, L., & Lee, K. (2020). Limitations of vehicle-based fatigue detection under variable road conditions. *Transportation Research Review, 19*(1), 56–70.

Eoh, H. J., Chung, M. K., & Kim, S. H. (2005). Electroencephalographic study of drowsiness in simulated driving with sleep deprivation. *International Journal of Industrial Ergonomics, 35*(4), 307–320.

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

Jazzar, M., Al-Saqqar, F., & Abouelmehdi, K. (2021). Comparative analysis of machine learning models for driver drowsiness detection. *Procedia Computer Science, 194*, 123–130.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]

Liu, Y., Wang, L., & Lee, H. (2024). Deep learning-based drowsiness detection models: Overfitting and generalization issues. *Neural Computing and Applications, 36*(1), 108–120.

Liu, Z., Zhang, J., & Li, W. (2019). Yawning detection as a proxy for driver fatigue assessment. *International Journal of Image and Signal Processing, 12*(3), 233–241.

May, J. F., & Baldwin, C. L. (2009). Driver fatigue: The importance of identifying causal factors of drowsiness. *Accident Analysis & Prevention, 41*(5), 869–875.

Patel, V., Kumar, M., & Sharma, A. (2020). ECG and HRV analysis for physiological fatigue detection in drivers. *IEEE Transactions on Biomedical Engineering, 67*(8), 2264–2271.

Qu, X., Zhang, R., & Adewole, S. (2023). Lightweight deep learning for embedded driver fatigue detection using TensorFlow Lite. *Journal of Mobile Vision Systems, 10*(2), 133–146.

Rahman, M. M., Hossain, M. S., & Alhamid, M. F. (2022). Facial landmark detection techniques in driver safety systems. *Multimedia Tools and Applications, 81*(12), 16593–16610.

Rundo, F., Trenta, F., & Di Stallo, A. (2023). Multimodal deep learning framework for driver monitoring and fatigue detection. *IEEE Access, 11*, 20189–20204.

Santos, L., Garcia, E., & Mejia, P. (2021). EEG-based driver fatigue detection systems: Accuracy and discomfort analysis. *IEEE Transactions on Intelligent Transportation Systems, 22*(3), 1501–1512.

Smith, J., Brown, L., & Clarke, H. (2021). PERCLOS for fatigue monitoring: Efficiency and limitations in modern systems. *Journal of Cognitive Systems, 7*(2), 91–103.

World Health Organization. (2018). *Global status report on road safety*. WHO Press.

Zhang, Y., Lin, Z., & Chen, Y. (2022). CNN-based drowsiness detection for embedded smart car systems. *Journal of Computer Vision Research, 9*(2), 99–112.

EDITOR-IN-CHIEF
Prof. E. Mclenna [Australia]