Journal of Computer and Communications, 2025, 13(9), 104-120
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

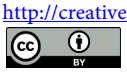
Driver Drowsiness Detection Based on Eye Blinking Using Convolutional Neural Network Approach

Mithun Kumar, Mahmudur Rahman, Jakir Khan, Abu Sale Mohammad Mostafizur Rahman

Department of Computer Science and Engineering, Bangladesh Army University of Engineering & Technology, Natore, Bangladesh
Email: mithunraj601@gmail.com, mahmudur19204027@gmail.com, jakirkhannisan9@gmail.com, himelmrj@gmail.com

How to cite this paper: Kumar, M., Rahman, M., Khan, J. and Rahman, A.S.M.M. (2025) Driver Drowsiness Detection Based on Eye Blinking Using Convolutional Neural Network Approach. *Journal of Computer and Communications*, **13**, 104-120.
https://doi.org/10.4236/jcc.2025.139006

Received: August 24, 2025
Accepted: September 23, 2025
Published: September 26, 2025



Abstract

This research presents a Driver Drowsiness Detection System (DDDS) that uses a Convolutional Neural Network (CNN) to improve road safety. The system uses a vast dataset of 97,860 images from the internet and 55,900 images created, achieving a combined accuracy of 99.66%. The individual datasets show exceptional performance, with the internet source dataset achieving 99.71% accuracy and the created dataset achieving 99.94%. The methodology prioritizes safety and reliability, resulting in a stable driver frame that meets functional requirements. The research approach uses advanced computer vision techniques, including edge detection, grayscale conversion, and dilation, to preprocess images from the MRL eye dataset. The methodology employs an OpenCV program for the precise detection of critical areas, specifically eyes and faces, using the Haar Cascade program. This work introduces a state-of-the-art solution to driver sleepiness detection and emphasizes the importance of a diverse and comprehensive dataset for robust model training.

Keywords

Artificial Intelligence, Driver Drowsiness Detection System, Deep Learning, Haar Cascade Algorithm, Machine Learning

1. Introduction

Sleep is essential for human functioning, and lack of it leads to fatigue, poor reflexes, loss of focus, and impaired decision-making—critical factors for safe driving. Fatigue and intoxication are among the leading causes of traffic accidents, particularly during night driving or long, uninterrupted trips, often resulting in

severe injuries and fatalities. While ongoing research seeks effective detection methods, challenges remain, as not all signs of negligence directly indicate drowsiness. Despite being overlooked by many, adherence to traffic rules is vital for safety. According to the Pan American Health Organization (PAHO), road accidents cause 1.35 million deaths and millions of injuries annually, with 90% of fatalities occurring in low- and middle-income countries, accounting for nearly 3% of their GDP [1].

In Peru, for example, 1853 persons lost their lives in traffic accidents during the first seven months of 2022, accounting for an average of 265 casualties per month. Over 47,600 traffic incidents were documented. Alarmingly, 30,032 persons died because of more than 74,620 traffic accidents in 2021 [2]. In 2022, drivers will be responsible for 27,396 (93.9%) of all road accidents; of these, the human component is one of the primary causes [3]. Drunk drivers are a major contributing factor in a lot of car accidents.

According to L. Fletcher *et al.*, drowsiness has been cited as a contributing factor in 30% of all road accidents [4]. It has been shown that growing tiredness impairs driving performance, with the consequence being crashes, which account for almost 20% of all vehicle accidents [5]. Using eye blink detection, Ursulescu *et al.* assessed the intervals between consecutive eye blinks to identify tiredness in the eyes [6]. Tabrizi *et al.* performed open or closed-eye analysis to detect tiredness using image processing techniques by saturation or the S-channel of the color model [7]. To detect eye blinks, Dehnavi *et al.* also employed image processing techniques [8]. To ascertain whether an eye is open or closed, Rahman *et al.* presented an architecture that takes advantage of the feature points of the eyes [9]. Viola-Jones proposed the use of cascade classifiers, which included characteristics for eye identification, for object detection [10]. Lu *et al.* performed eye recognition using texture characteristics based on rectangle and pixel-pattern features [11]. Rahul Atul Bhone's study uses facial characteristics to identify sleepiness. It calculates the eye aspect ratio (EAR) for each frame and uses the Viola-Jones-based classifier to detect eye activity. If the EAR is below a threshold, it indicates a closed-eye condition [12].

This research investigates the transformative potential of Driver Drowsiness Detection using Convolutional Neural Networks (CNNs) to analyze eye-blinking patterns and reduce fatigue-related accidents [13]. Trained on diverse datasets, the system evaluates driver alertness through real-time video analysis and issues timely alerts such as audible or visual warnings. The objectives include data collection and preprocessing, feature selection, and comparative evaluation against other models. Through iterative refinement, cross-validation, and fine-tuning, the study aims to develop a highly accurate and reliable drowsiness detection system.

The motivation for developing a real-time drowsiness detection system stems from the high number of accidents caused by driver fatigue. A proactive system can alert drivers when signs of drowsiness are detected, prompting corrective actions such as taking breaks or pulling over [7]. This study proposes a Driver

Drowsiness Detection System (DDDS) using Convolutional Neural Networks (CNNs) to analyze eye-blinking patterns, aiming to overcome the limitations of existing methods by improving both accuracy and real-time responsiveness [14].

The proposed CNN-based method for DDDS resonates deeply within the domains of road safety and intelligent transportation systems. By leveraging Convolutional Neural Networks, it contributes to preventing individual accidents and laying the groundwork for a future where intelligent vehicles collaboratively enhance overall road safety standards [5] [8]. The system can proactively prevent accidents caused by drowsy driving, reduce road fatalities and injuries, enhance driver assistance systems, influence road safety policies and regulations, and foster a culture of responsible driving habits.

The research aims to push the boundaries of CNN models by fine-tuning their architecture to achieve heightened accuracy in real-time drowsiness detection. The model is trained to detect minor variations related to drowsiness using a dataset carefully selected for its relevance to driver behavior and drowsiness indications [15]. Comparative analyses with other machine learning paradigms, such as Recurrent Neural Networks (RNN), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM), help evaluate the efficacy of the CNN model [16].

This research lies at the intersection of advanced CNN model refinement, comprehensive comparative analysis, and the strategic integration of diverse datasets. Its goal is to enhance road safety by developing a compact and real-time responsive model for drowsiness detection.

2. Algorithms

Drowsy drivers pose a significant threat to road safety by increasing the likelihood of accidents and exhibiting slower reaction times. To mitigate this risk, advanced technologies leveraging artificial intelligence and machine learning play a crucial role in developing proactive drowsiness detection systems [15].

2.1. HAAR Feature Selection

Features are selected for object detection based on positive and negative training images. Haar-like features include Line, Edge, and Rectangular Features. Features are determined by subtracting pixel sums under white and black rectangles. Hence, the goal is to analyze useful features and reduce processing time.

2.2. Integral Images and Adaboost

Intermediate representation facilitating fast rectangular region calculations. Resolves the repetitive computation issue in Haar feature selection. Allows for efficient computation of Haar features of different sizes and locations. Essential stage applying each feature to find the best threshold for accurate object classification. Features with minimum error rates are selected to form a strong classifier. Adaboost training reduces the number of required features and enhances accuracy.

2.3. Cascade Classifiers

The Haar Cascade efficiently reduces computation by focusing on object regions and using a cascade of weak classifiers. Detection is performed with a sliding window across the input image, making it robust and suitable for real-time tasks such as face and pedestrian detection. This study highlights its applications in computer vision, efficiency in real-time processing, and the importance of training classifiers for specific objects, while noting the availability of pre-trained models in libraries like OpenCV.

3. Methodology

The paper presents a thorough drowsiness detection system that is intended to closely monitor the physical behavior, emotions, and environment of the driver. Key aspects of the drowsiness detection system include the following.

3.1. Adaptability and Machine Learning

The technology adapts to different driving situations and individual variations. Machine learning algorithms continually learn and adapt to diverse driving styles, environmental conditions, and individual traits.

3.2. Real-Time Alerts and Responses

The system generates real-time alerts, such as tactile or audio signals, when drowsiness exceeds a predefined threshold, prompting drivers to take corrective action or rest. Integration with vehicle control systems further enables the activation of safety features.

3.3. Data Collection Procedure

A diverse dataset is gathered to represent different driving scenarios, with balanced alert and drowsy states, and annotations obtained through crowd-sourcing.

In **Figure 1**, we researched a variety of images from the 97,860 images in the downloaded datasets from Kaggle [17] that showed people with their eyes open and closed. Based on whether noisy data was present or not, the images were split into two classes. The following steps outline the specific procedures for obtaining a labeled dataset:

1) Image Data Acquisition: Collect a diverse set of images capturing drivers in various real-world driving scenarios. Utilize on-board cameras, dashcams, or collaborate with vehicle manufacturers to obtain images representing different lighting conditions, driver perspectives, and road environments.

2) Alert and Drowsy States: Ensure the dataset includes a balanced representation of both alert and drowsy states of drivers. Capture images depicting drivers with varying levels of fatigue, such as open-eyed alertness and closed-eyed drowsiness.

3) Crowdsourcing for Annotation: Engage with drivers, road safety organizations, or online platforms to collect annotations for the images.

Figure 1. Various eyes images from download and created datasets.

4) Data Cleaning and Preprocessing: Remove irrelevant or duplicate images to streamline the dataset. Standardize image formats, correct for lighting variations, and resize images as needed.

5) Privacy Measures: Adhere to data privacy regulations and obtain explicit consent from individuals in the images. Anonymize or encrypt sensitive facial features to protect the privacy of drivers.

3.4. Processing Model

A flow diagram demonstrates the working model of the system, involving image capture, frame division, and the use of datasets for face and eye detection (Figure 2). Drowsiness detection triggers an alarm if fatigue is identified, outlining a real-time monitoring system to prevent potential accidents.

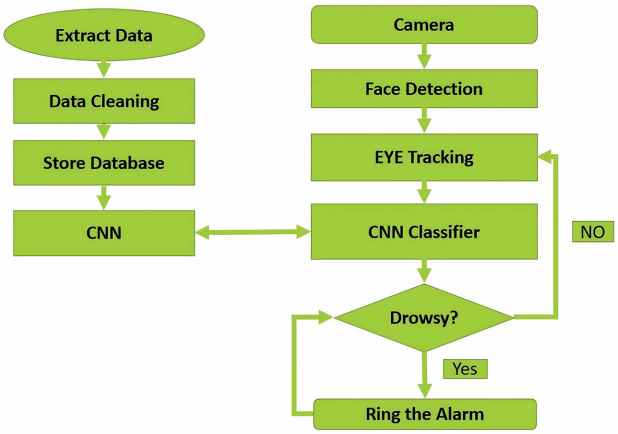


Figure 2. Experimental model.

3.5. System Architecture

The system architecture outlines the design and organization of components, detailing procedures for data pre-processing, extraction, and transformation using CNN (Figure 3). To make our model more powerful, we incorporate non-linear activation functions (such as the popular ReLU—Rectified Linear Unit) after the convolution and pooling layers. This adds a crucial layer of non-linearity, allowing the model to capture intricate and complex relationships. Once several convolu-

tion and pooling layers have done their job, a flattened layer comes into play. The fully connected layers, also known as dense layers, are where traditional neural networks shine. Every neuron in these layers connects to all the neurons in the preceding and succeeding layers. This enables the network to make holistic decisions by leveraging the learned features.

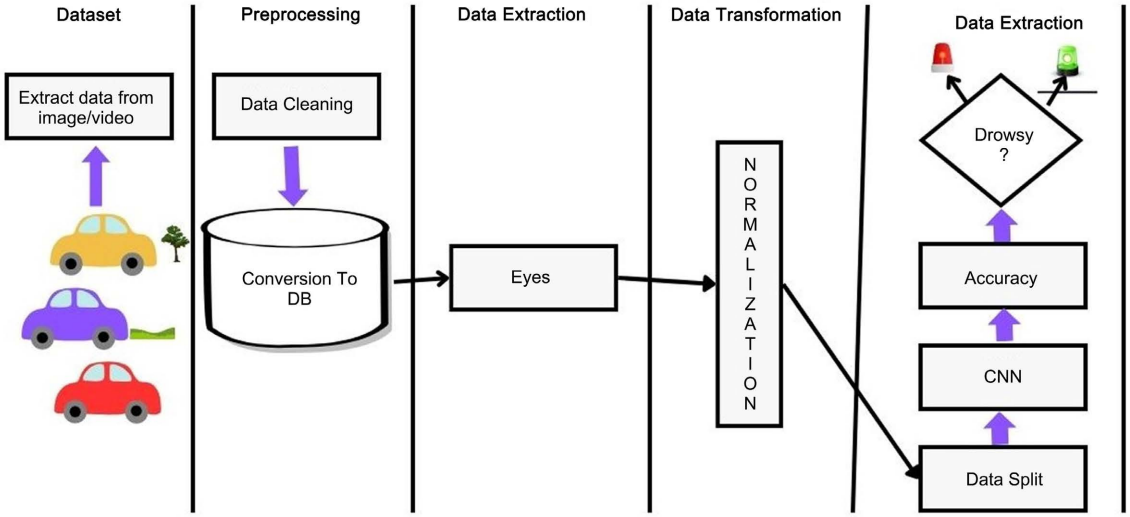


Figure 3. System architecture.

The key components are the convolution layers, pooling layers, dropout layers, and the dense layers. Input Data: Let Xtrain be the input data, where Xtrain Rm × 24 × 24 × 1, where m is the number of samples, and the images are (24 × 24) pixels with a single channel (grayscale).

First Convolution Layer: Input shape: (24, 24, 1) Number of filters: 32 Kernel size: (3, 3) and Output shape: (22, 22, 32). The output at each spatial position is calculated by convolving the 3 × 3 kernel with the input at that position, followed by the ReLU activation. First MaxPooling Layer: Pool size: (1, 1) and Output shape: (22, 22, 32) (no change in size due to pool size)

Second Convolution Layer: Number of filters: 32 Kernel size: (3, 3) and Output shape: (20, 20, 32). Second MaxPooling Layer: Pool size: (1, 1) and Output shape: (20, 20, 32).

Third Convolutional Layer: Number of filters: 64 Kernel size: (3, 3) and Output shape: (18, 18, 64) Third MaxPooling Layer: Pool size: (1, 1) and Output shape: (18, 18, 64) (no change in size due to pool size). Dropout Layer: Dropout with a rate of 0.25 is applied after the third max-pooling layer. This helps prevent overfitting by randomly setting a fraction of input units to zero during training.

Flatten Layer: The Flatten layer is used to flatten the output from the previous layer into a one-dimensional array. Output shape: (18, 18, 64). First Dense Layer: Number of units: 128, Activation function: ReLU-Output shape: (128).

Dropout Layer: Dropout with a rate of 0.5 is applied after the first dense layer for further regularization. Output Layer: Number of units: 2 (assuming binary classification). Model Training: The model is trained using the "fitgenerator" method

or 15 epochs with a batch size of 32.

4. Experimental Results

Developing a driver drowsiness detection system requires integrating key components such as a user interface, image preprocessing, and a CNN-based detection model. The design incorporates real-time video monitoring, results presentation, database integration, and emphasizes scalability and performance for optimal user experience. Experimental results reflect careful integration and optimization across all modules. A user-friendly interface enhances accessibility, while fine-tuned image training and preprocessing ensure robust feature extraction for improved model precision.

The CNN-based model enables detection of subtle drowsiness cues, supported by real-time video monitoring for continuous assessment. Results are presented through clear, interpretable outputs, with database integration ensuring efficient data management. Scalability and performance allow the system to adapt to varying conditions and deliver consistent outcomes. Performance metrics demonstrate high accuracy, efficiency, and responsiveness, validating the effectiveness of this integrated approach in mitigating driver drowsiness and enhancing road safety. Implementing a Real-Time Driver Drowsiness Detection System based on Eye Blinking using a Convolutional Neural Network (CNN) approach involves several key requirements to ensure a successful deployment.

4.1. Functional and Non-Functional Requirements

The functions, features, and capabilities that a system, software program, or product must have to satisfy the intended customers or stakeholders are outlined in functional requirements, which are specifications. Here are key characteristics of functional requirements: OpenCV, TensorFlow, Keras, Pygame. A webcam is required as a prerequisite for this Python-based process. The entire system design is divided into 6 parts—face recognition, eye detector, face tracking, eye tracking and distraction detection. We may create a reliable and efficient real-time sleepiness recognition system using a Convolutional Neural Network technique by attending to these implementation criteria.

4.2. Experimental Setup

The experimental setup is designed to control variables, observe phenomena, and measure outcomes in a systematic and reproducible manner. The setup process typically includes the key activities such as Operating System, Programming Languages, AI and Machine Learning, Development Tools and Version Control.

4.3. Result Analysis

Experimental result analysis involves the examination and interpretation of data obtained from experiments to draw meaningful conclusions and insights. It is a critical step in the scientific method where researchers assess the outcomes of their experiments to answer research questions, test hypotheses, or gain a deeper un-

derstanding of a phenomenon.

Figure 4 shows the steps to process data and build a model in machine learning or data science. It starts with importing libraries, which are like tools or resources needed for the task. Then it moves to converting data into a usable format. The dataset is then split into two parts: one for training the model and another for testing it. The model is built and generated using the training data, then it's tested with the test data to see how well it performs. The accuracy of the model is checked, and finally, there's an output which could be results or decisions made based on that model. We want to build a model that can recognize different types of eye pictures. We would start by importing libraries that can help you with this task. We would then split the data into two parts: one for training the model and another for testing it. We would use the training data to build the model, and then test it with the test data to see how well it performs. If the model performs well, you could use it to recognize different types of fruits. If not, you would need to adjust the model until it works well.

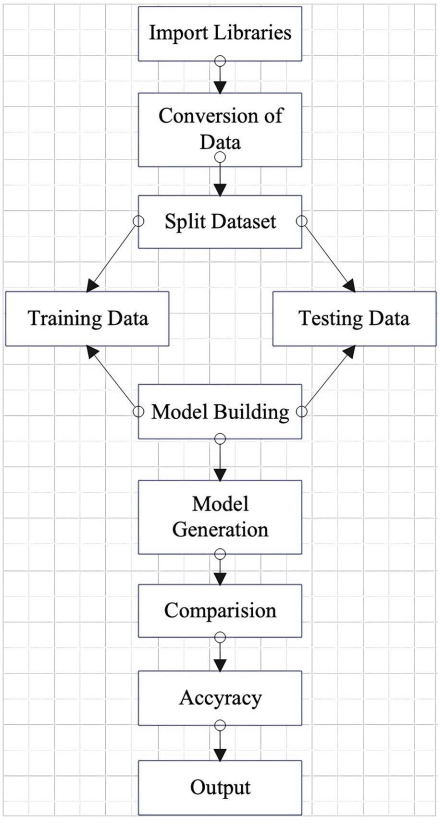


Figure 4. Train CNN flowchart.

A CNN model's training accuracy, validation accuracy, validation loss, and training accuracy across several epochs are displayed in Figure 5. Training accuracy starts high and remains stable, indicating good performance on training data. Validation accuracy starts high but slightly decreases as epochs increase, indicating overfitting. Training loss starts low and remains stable, while validation loss

starts low but increases slightly with more epochs, indicating overfitting. Monitoring both losses is crucial to ensure the model generalizes well to unseen data. When a model performs well on training data but badly on validation data due to its excessive complexity, this is known as overfitting.

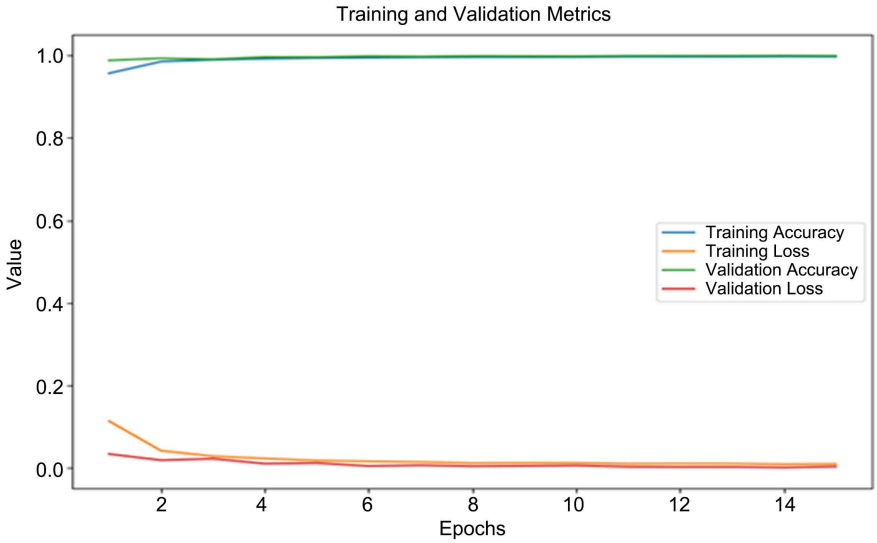


Figure 5. Training and validation metrics (download dataset).

A confusion matrix is a graphical depiction of a classification model's performance, as seen in **Figure 6.** It displays the total number of predictions the model made for a binary classification job where the classifications are "open" and "close," including true positive, true negative, false positive, and false negative. The projected class is represented by the columns, while the actual class is represented by the rows. The values in each cell (e.g., 12,241 true positives for "close", 13,842 false negatives for "close") show the number of occurrences of each combination.

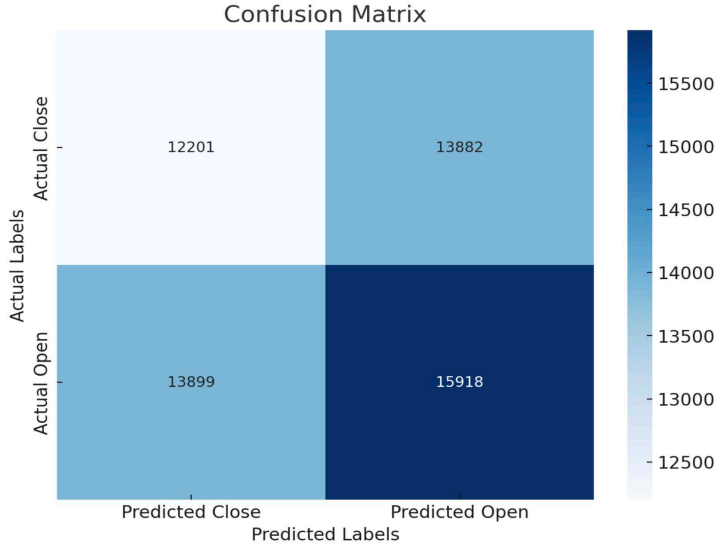


Figure 6. Confusion matrix (download dataset).

The improvement of training accuracy, training loss, validation accuracy, and validation loss across several epochs during the CNN model training is depicted by the graph in Figure 7. The y-axis values vary from 0 to 1, while the x-axis indicates the number of epochs or iterations across the full dataset. The blue line represents Training Accuracy, which begins at a high value and stays largely constant throughout epochs. This suggests that the performance of the training data is consistent. Conversely, the Validation Accuracy (green line) shows a little decrease with increasing epochs after starting at a high position. This implies that a possible overfitting signal is somewhat diminished by the model's performance on unseen (validation) data as it becomes more adapted to the training set. The Training Loss (red line) starts at a low value and maintains stability, indicating minimal errors or losses during training. In contrast, the Validation Loss (orange line) begins at a low point but experiences a slight increase with more epochs. A rise in validation loss while training loss remains low serves as an indicator of overfitting.

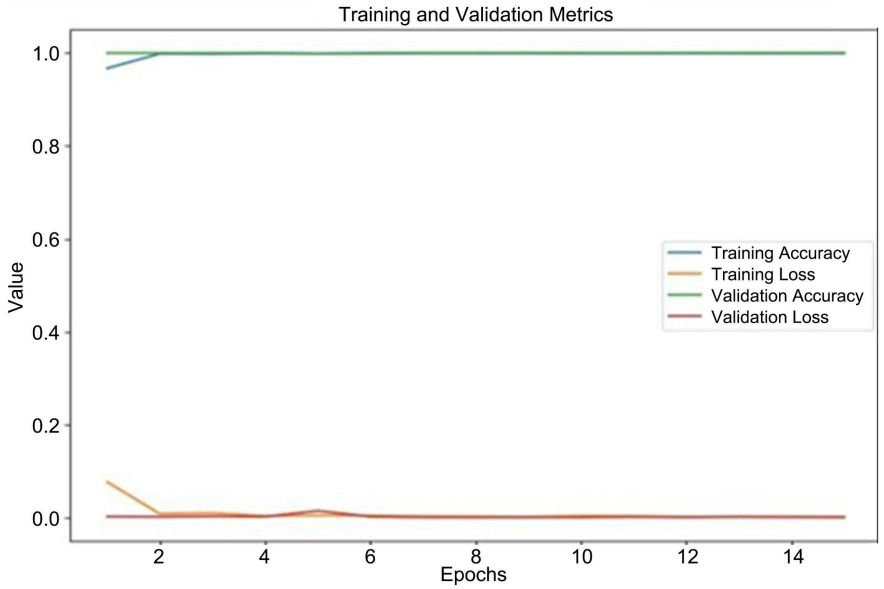


Figure 7. Training and validation metrics (created dataset).

A confusion matrix is applied in Figure 8 to show a classification model's performance. The real and expected classifications of "close" and "open" are displayed. Errors are shown by off-diagonal cells, but the diagonal cells (top left and bottom right) show accurate predictions. There are 12,044 true positives where "close" was correctly predicted, and 15,761 true negatives where "open" was correctly predicted. However, there are 14,039 false positives where "close" was incorrectly predicted as "open", and 14,056 false negatives where "open" was incorrectly predicted as "close".

A CNN model's training accuracy, training loss, validation accuracy, and validation loss are displayed throughout several epochs in Figure 9. Good performance on training data is indicated by training accuracy that starts high and stays constant. Overfitting is indicated by a high initial validation accuracy that gradu-

ally declines as epochs rise. Validation loss begins low and grows significantly with further epochs, while training loss starts low and stays constant. It is essential to keep an eye on both losses to make sure the model applies effectively to new data. When a model is very intricate and excels on training data but struggles with validation data, it is said to be overfitted.

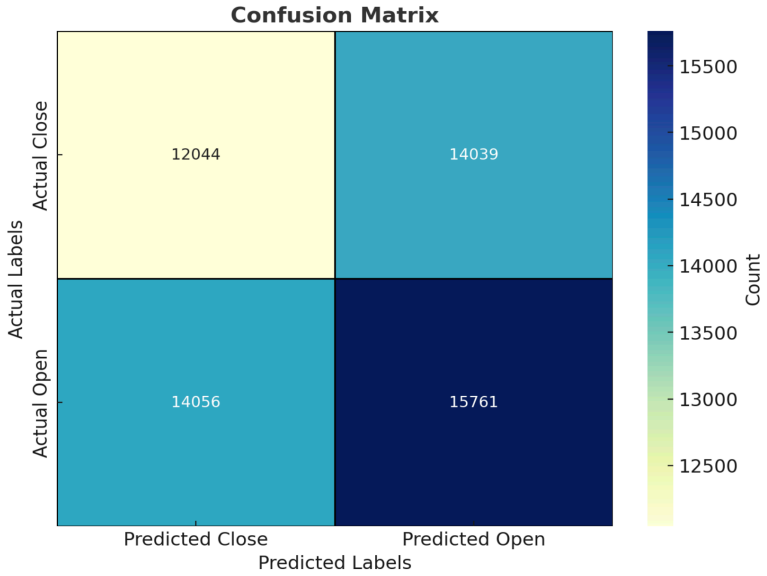


Figure 8. Confusion matrix (created dataset).

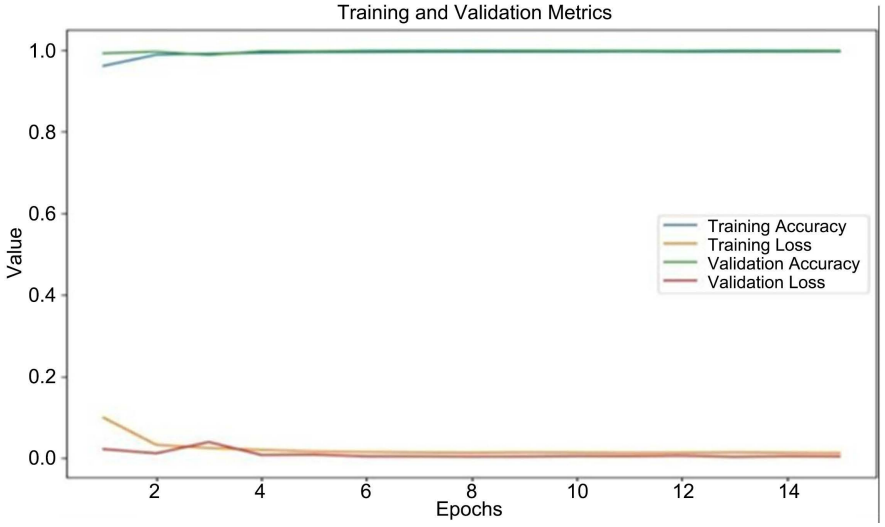


Figure 9. Training and validation metrics (combine dataset).

Figure 10. Illustrates a perplexity. A matrix is a visual depiction of a classification model's performance that indicates whether an object is "open" or "close." The genuine labels are shown by the y-axis, while the anticipated labels are represented by the x-axis. The count of occurrences for each set of actual and anticipated labels is indicated by the numbers in each cell. As an illustration, there are 12,201 cases in which the real and predicted labels are both "close," and 15,918

cases in which they are both "open." Predictions have also been off in 13,882 and 13,899 cases, respectively, where it was expected to be "open" but it was instead "close" and "close" respectively.

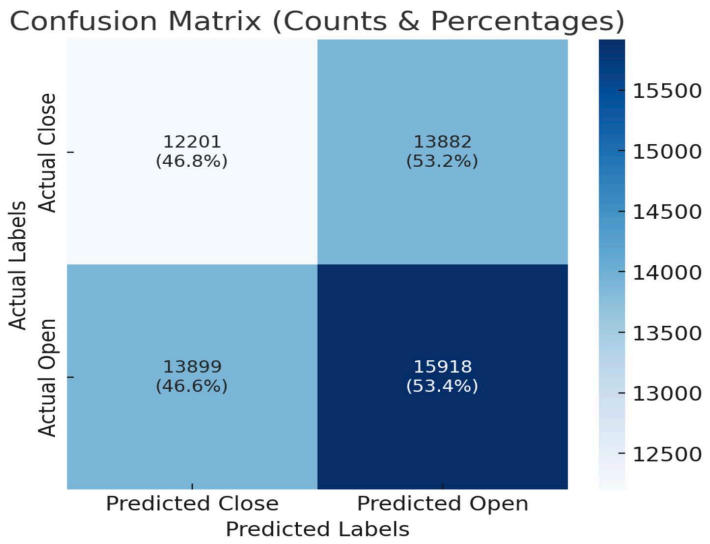


Figure 10. Confusion matrix (combine dataset).

4.4. Comparisons among Models

Accuracy in machine learning refers to the model's performance on the dataset. The model's ability to make predictions improves with increasing accuracy. One specific kind of neural network that performs image classification tasks is the CNN model. One kind of neural network that succeeds in processing sequential data, like text or voice, is the RNN model. One kind of non-parametric technique utilized for regression and classification problems is the KNN model. One kind of algorithm used for problems involving regression and classification is the SVM model. Among the three datasets, CNN model accuracy is the greatest. Of the three datasets, the SVM model's accuracy is the lowest. The KNN and RNN models have similar accuracies across all three datasets.

Figure 11 shows the accuracy (Download Dataset) of four different machine learning models: CNN, SVM, RNN, and KNN. The CNN model has the highest accuracy at 99.71%, followed by KNN at 95.85%, RNN at 98.43%, and SVM has the lowest accuracy at 65.14%.

Figure 12 shows the accuracy (Created Dataset) of four different machine learning models: CNN, SVM, RNN, and KNN. The CNN model has the highest accuracy at 99.94%, followed by KNN at 99.56%, RNN at 98.43%, and SVM has the lowest accuracy at 68.09%.

Figure 13 shows the accuracy (Combine Dataset) of four different machine learning models: CNN, SVM, RNN, and KNN. The CNN model has the highest accuracy at 99.66%, followed by RNN at 98.18%, KNN at 97.29%, and SVM has the lowest accuracy at 59.35%.

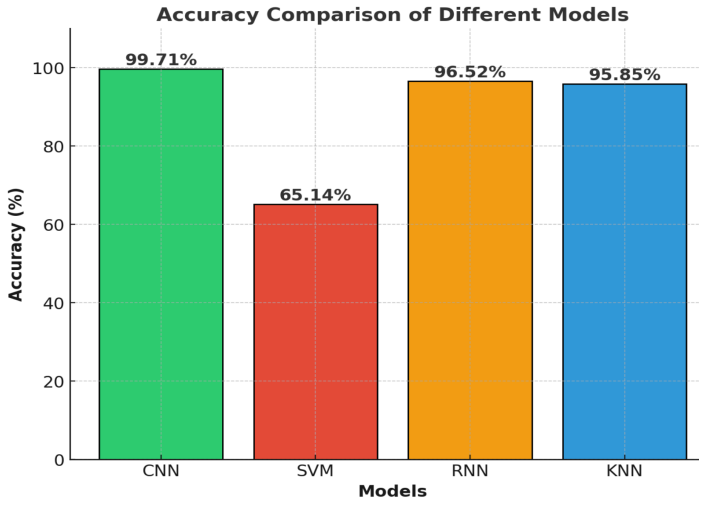


Figure 11. Compared model (download dataset).

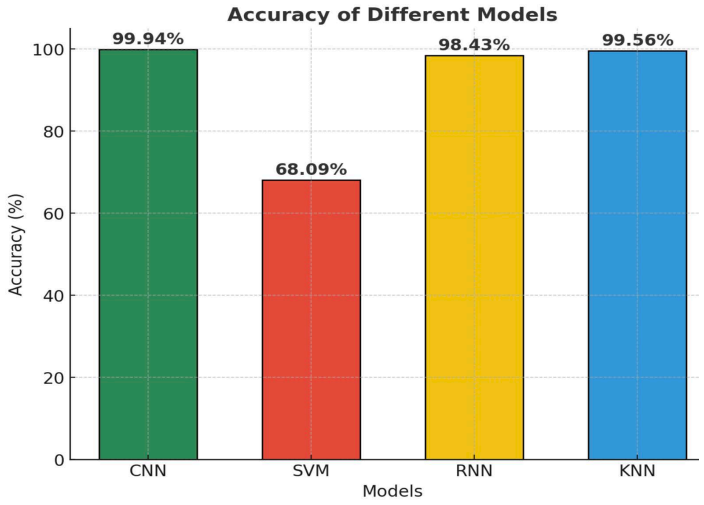


Figure 12. Compared model (created dataset).

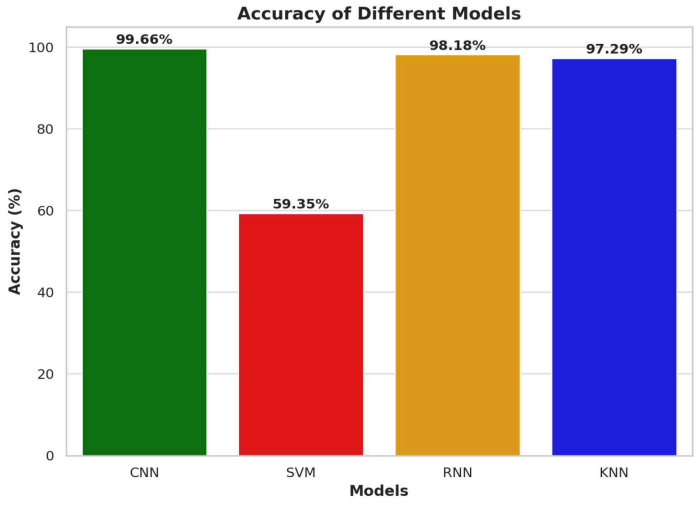


Figure 13. Compared model (combine dataset).

Figure 14 shows accuracy of CNN model based on dataset. There are three bars, each representing the accuracy of a CNN model trained on different datasets. The y-axis is labeled "Accuracy (%)" and ranges from 0% to 100%. The x-axis is labeled "CNN Model" with three categories: "Download Dataset," "Create Dataset," and "Combine Dataset." The first bar, representing the Download Dataset, is purple and shows an accuracy of 99.71%. The second bar, for Create Dataset, is teal and has an accuracy of 99.94%. The third bar, for Combine Dataset, is yellow and indicates an accuracy of 99.66%.

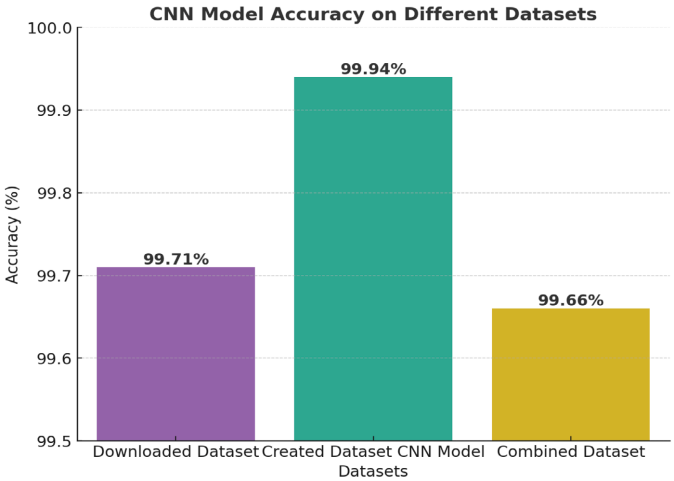


Figure 14. CNN model comparison based on dataset.

4.5. Testing

The testing flowchart shows how to detect driver tiredness in real time using eye blinking. Results from both initial testing and in-the-moment testing are given, indicating how well the system works to warn sleepy drivers.

Figure 15 shows the testing flowchart of a driver drowsiness detection system based on eye blinking using CNN. The process begins with a camera capturing the driver's face, followed by face and eye detection.

Figure 16 presents the results of real-time testing for a set of six images that are possibly expected to demonstrate a CNN based Driver Drowsiness Detection System (DDDS) based on eye blinking. This system can tell if a driver is getting sleepy by watching how often they blink their eyes. If the driver blinks too slowly or their eyes stay closed too long, like when people are sleepy, the system knows the driver might be drowsy and can alert them to wake up or take a break.

Table 1. Testing cases.

Test Cases	Eye Detected	Results
Case 1	No	No result
Case 2	Yes	Alarm Alert

Table 1 shows the testing cases. For the first case (no eye detected), there is no

specific result. For the second case (eye detected), an alarm alert is triggered. This table likely pertains to testing a system or software related to eye detection. When an eye is detected, it responds with an alarm alert.

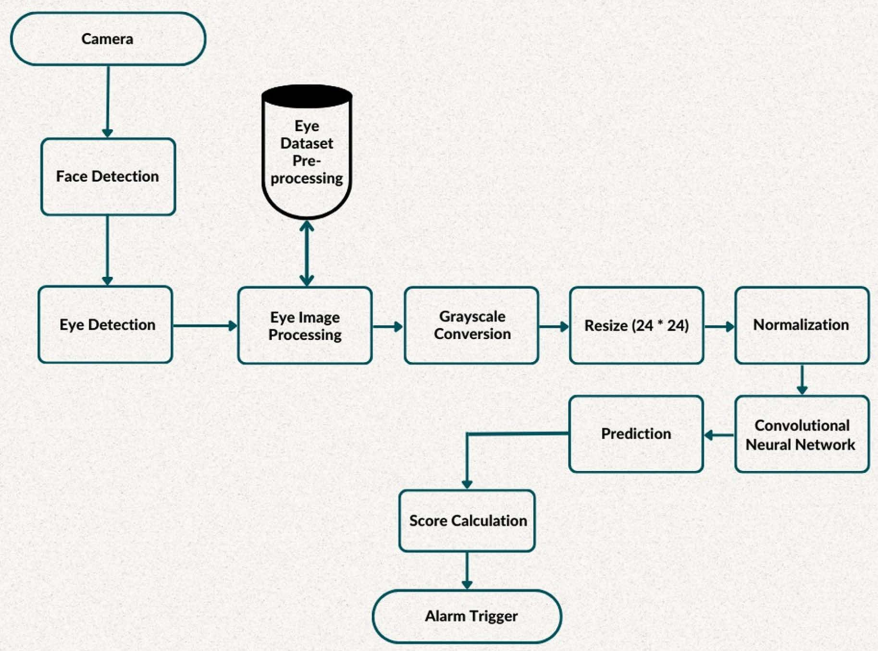


Figure 15. Testing flowchart.

Figure 16. Real-time testing.

5. Discussion

By leveraging AI-based facial and eye-movement analysis, the system can precisely detect signs of drowsiness and issue real-time alerts, such as visual or audible cues, prompting drivers to take corrective action. Beyond detection, the platform supports drivers with safe driving tips, reminders for breaks, and guidance on staying alert during long journeys. The system can provide driving condition

updates, including weather and road information, to aid decision-making and trip planning. From this research, we have shown that the CNN performs better than other algorithms.

6. Conclusion

Driver fatigue is a major cause of traffic accidents, and this research addresses the issue through a real-time sleepiness detection device powered by Convolutional Neural Networks (CNNs). The model demonstrated strong accuracy and reliability through dataset collection, CNN development, and rigorous evaluation. Its key contribution lies in enhancing road safety by offering a robust tool for timely drowsiness detection. In the future, we will integrate technology with insights from psychology and human behavior, and more advanced and personalized solutions can be developed. This interdisciplinary approach aligns with the broader goal of improving driver well-being and preventing fatigue-related accidents.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Road Safety (n.d.) PAHO/WHO | Pan American Health Organization. https://www.paho.org/en/topics/road-safety

[2] Efe, A. (2022) Unas 265 personas murieron cada mes del 2022 en ac-cidentes de tránsito en Perú. Gestión. https://gestion.pe/peru/unas-265-personas-murieron-cada-mes-del-2022-en-accidentes-de-transito-en-peru-noticia/

[3] De Seguridad Vial, O.N. (2022) ONSV—Informe de siniestralidad vial y las acciones para promover la seguridad vial. https://www.onsv.gob.pe/post/informe-de-siniestralidad-vial-y-las-acciones-para-promover-la-seguridad-vial/

[4] Fletcher, L., Petersson, L. and Zelinsky, A. (2004) Driver Assistance Systems Based on Vision in and Out of Vehicles. *IEEE IV*2003 *Intelligent Vehicles Symposium. Proceedings* (*Cat. No.*03 *TH*8683), Columbus, 9-11 June 2003, 322-327. https://doi.org/10.1109/ivs.2003.1212930

[5] Eskandarian, A. and Mortazavi, A. (2007) Evaluation of a Smart Algorithm for Commercial Vehicle Driver Drowsiness Detection. 2007 *IEEE Intelligent Vehicles Symposium*, Istanbul, 13-15 June 2007, 553-559. https://doi.org/10.1109/ivs.2007.4290173

[6] Ursulescu, O., Ilie, B. and Simion, G. (2018) Driver Drowsiness Detection Based on Eye Analysis. 2018 *International Symposium on Electronics and Telecommunications* (*ISETC*), Timisoara, 8-9 November 2018, 1-4. https://doi.org/10.1109/isetc.2018.8583852

[7] Tabrizi, P.R. and Zoroofi, R.A. (2008) Open/Closed Eye Analysis for Drowsiness Detection. 2008 *First Workshops on Image Processing Theory, Tools and Applications*, Sousse, 23-26 November 2008, 1-7. https://doi.org/10.1109/ipta.2008.4743785

[8] Dehnavi, M., Attarzadeh, N. and Eshghi, M. (2011) Real Time Eye State Recognition. 2011 19*th Iranian Conference on Electrical Engineering*, Tehran, 17-19 May 2011, 1-4.

[9] Rahman, A., Sirshar, M. and Khan, A. (2015) Real Time Drowsiness Detection Using Eye Blink Monitoring. 2015 *National Software Engineering Conference* (*NSEC*), Rawalpindi, 17 December 2015, 1-7. https://doi.org/10.1109/nsec.2015.7396336

[10] Viola, P. and Jones, M. (2001) Rapid Object Detection Using a Boosted Cascade of Simple Features. *Proceedings of the* 2001 *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR* 2001, Kauai, 8-14 December 2001, 1. https://doi.org/10.1109/cvpr.2001.990517

[11] Lu, H.C., Zhang, W. and Yang, D.L. (2007) Eye Detection Based on Rectangle Features and Pixel-Pattern-Based Texture Features. 2007 *International Symposium on Intelligent Signal Processing and Communication Systems*, Xiamen, 28 November-December 2007, 746-749. https://doi.org/10.1109/ispacs.2007.4445995

[12] Bhope, R.A. (2019) Computer Vision Based Drowsiness Detection for Motorized Vehicles with Web Push Notifications. 2019 4*th International Conference on Internet of Things*: *Smart Innovation and Usages* (*IoT-SIU*), Ghaziabad, 18-19 April 2019, 1-4. https://doi.org/10.1109/iot-siu.2019.8777652

[13] Chirra, V., ReddyUyyala, S. and KishoreKolli, V. (2019) Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. *Revue d'Intelligence Artificielle*, **33**, 461-466. https://doi.org/10.18280/ria.330609

[14] Salomi Victoria, D.R. and Ratna Mary, D.G. (2021) Driver Drowsiness Monitoring Using Convolutional Neural Networks. 2021 *International Conference on Computing, Communication, and Intelligent Systems* (*ICCCIS*), Greater Noida, 19-20 February 2021, 1055-1059 https://ieeexplore.ieee.org/document/9397070

[15] Assari, M.A. AND Rahmati, M. (2011) Driver Drowsiness Detection Using Face Expression Recognition. 2011 *IEEE International Conference on Signal and Image Processing Applications* (*ICSIPA*), Kuala Lumpur, 16-18 November 2011, 337-341. https://www.semanticscholar.org/paper/Driver-drowsiness-detection-us-ing-face-expression-Assari-Rahmati/6ec7e071c0b6ce3213cf9cdcdafa7ac7885cc32b

[16] Yan, C., Coenen, F., Yue, Y., Yang, X. and Zhang, B. (2016) Video-Based Classification of Driving Behavior Using a Hierarchical Classification System with Multiple Features. *International Journal of Pattern Recognition and Artificial Intelligence*, **30**, Article ID: 1650010. https://doi.org/10.1142/s0218001416500105

[17] (2022) OACE—Open and Close Eyes Dataset. Kaggle. https://www.kaggle.com/datasets/muhammadhananasghar/oace-open-and-close-eyes-dataset