International Journal of Artificial Intelligence and Digital Innovation
https://journals.hu.edu.so/index.php/ijaidi

Embedded Eye-State Classification and Fleet Event Logging for a Driver Drowsiness Alert Prototype

Abdisamed Mohamed Nor[1]; Mohamed Osman Hassan[1]; Mohamed Sadik Ibrahim[1]; Sudeis Mohamed Botan[1], Mohamed Abdullahi Ali*[2]
[1]Department of Software Engineering, Faculty of Computer Science and Information Technology, Hormuud University, Mogadishu, Somalia
[2]Faculty of Computer Science and Information Technology, Jazeera University, Mogadishu, Somalia

*Corresponding author's email: Ileysyare@gmail.com

Abstract
Camera-based drowsiness alerting requires reliable visual measurements and an operational process for event recording. This study developed a preliminary Raspberry Pi prototype that combined a compact MobileNetV2 eye-state classifier, consecutive-frame alert logic, and fleet event dashboard. A balanced subset of 20,000 MRL eye images was allocated at the image level to the training, validation, and test partitions in a 75/15/10 ratio. The classifier used 64 × 64 grayscale inputs, a width multiplier of 0.35, two-stage transfer learning, and TensorFlow Lite deployment. The compact classifier contained approximately 492,000 parameters and was trained for 23 epochs. On 2,000 test eye images, it correctly classified 1,957 images, achieving 97.85% accuracy, 98.40% closed-eye sensitivity, 97.30% open-eye specificity, and a receiver operating characteristic area under the curve of 0.9977. A Raspberry Pi 5 desk-based session processed 2,824 frames in 202.4 seconds, with a reported average of 13.95 frames per second and 1.24 ms per eye-classification call. The FastAPI and SQLite platform supported the event history, evidence images, and company, vehicle, and device management. Fifteen event records were observed, of which three lacked resolved vehicle associations. These findings support the preliminary technical feasibility of the reported tests. This study did not establish subject-independent performance, drowsiness diagnosis, or operation in a moving vehicle.

Keywords: Driver drowsiness, Eye-state classification, MobileNetV2, Embedded inference, Fleet monitoring

1 Introduction

Road safety is a practical motivation for driver monitoring. The World Health Organization [1] estimated 1.19 million road traffic deaths annually in its global status report. This total covers all road traffic causes and should not be interpreted as a drowsiness-specific estimate. The present study addresses a narrower engineering problem: detecting sustained eye closure from a camera stream and making the resulting alerts available for a later fleet review.

Eye closure is observable without attaching physiological sensors to the driver, but it is only one behavioral indicator of drowsiness. Reviews distinguish visual measures from physiological and vehicle-based approaches and emphasize that drowsiness ground truth and

temporal interpretation remain separate methodological questions [2]–[4]. A closed-eye image may represent an ordinary blink, and a drowsy driver need not have their eyes closed in every frame. Florez et al. [5] investigated CNN-based eye-state identification for driver monitoring, while Florez et al. [6] combined eye and mouth analysis in an embedded system.

The problem considered in this study has two parts. First, a resource-constrained device must locate the eye regions, classify them, and accumulate evidence across frames without interrupting the video stream. Second, alerts must remain interpretable after the immediate warning has ended. Fleet staff require an event timestamp, evidence image, and reliable association with the relevant device and vehicle. A classifier score alone cannot provide an operational record.

Consequently, this study integrates a lightweight eye-state model with a local detector, audible warning, screenshot capture, and web dashboard. Its contribution is an implemented monitoring workflow with a separate classifier, runtime, and functional evidence. This study adapts an established neural network architecture and evaluates a preliminary prototype. The system deliberately excludes automatic driver identification and records the monitored subject as a Detected Person.

The general objective was to implement and evaluate an embedded eye-state monitoring prototype linked to fleet-event records. The specific objectives were as follows: (1) to train a compact open- and closed-eye classifier; (2) to integrate the classifier with camera localization and consecutive-frame warning logic; (3) to record alerts with image evidence and fleet context; and (4) to assess image-level classification, Raspberry Pi runtime, and dashboard operation as separate outcomes.

The primary outcome was binary eye state classification within the documented MRL image split. The secondary outcome was the execution of the processing loop during one Raspberry Pi 5 session. The third outcome is a functional inspection of event storage and fleet contexts. The eye state is used as an input to a warning rule and is not treated as a physiological confirmation of drowsiness. This study did not evaluate collision reduction, diagnostic validity, subject-independent generalization, or performance during real driving.

2 Literature Review

2.1 Visual indicators and temporal interpretation

Visual driver monitoring uses observable facial behavior to support alert decisions. Eye-state classification estimates whether a supplied eye crop is open or closed. A complete system must also identify suitable regions in the camera frames and determine whether the sequence warrants an alert. Florez et al. [5] provide a relevant example of using convolutional classification for this purpose. Their work motivates the learning of eye appearance instead of relying exclusively on manually chosen geometric features, while leaving the temporal decision as a separate design question.

Temporal interpretation is required because frame-level eye labels do not directly represent a drowsiness episode. Shiau et al. [7] used eye-state sequences to construct temporal measures, whereas Kolus [4] systematically reviewed eye-activity measures across driver monitoring studies. Additional facial cues may also provide information when the visibility of the eyes is poor. The embedded approach of Florez et al. [6] combined convolutional eye analysis with the mouth aspect ratio. The present system retains only eye-state information, which limits its coverage of fatigue-related behavior but reduces the number of visual cues processed by the prototype.

Recent reviews have described the continuing variation in sensors, ground truth, datasets, validation protocols, and operational metrics [8], [9]. Consequently, headline accuracy values from different studies are not directly comparable unless the subjects, data

partitions, endpoint definitions, and testing conditions are aligned properly. This issue is especially relevant when repeated images from the same participants may occur in more than one partition of the dataset.

2.2 Compact classification and deployment evidence

MobileNetV2 uses inverted residual blocks and linear bottlenecks to construct an efficient convolutional representation of an image [10]. This study adopts the established architecture with a reduced width multiplier and small classification head. Earlier embedded work also showed that the model size and classifier speed must be considered together with the complete processing pipeline [11]. Therefore, this study reports the per-crop inference time and complete-loop frame rate as different measurements.

The MRL Eye Dataset provides 84,898 infrared eye images from 37 people and includes labels for eye state, glasses, reflections, illumination, and sensor type [12]. Although it is suitable for developing an eye-state classifier, it does not establish performance on visible-light webcam images or continuous driver videos. Subject overlap between partitions also affects the interpretation of the held-out image results. These dataset properties are relevant to the present study because the prototype uses an RGB webcam, and the documented split does not establish participant separation.

2.3 Study focus and research gap

The literature establishes eye-state analysis, temporal measures, and embedded warning systems as existing research directions [3], [13]. However, technical reports frequently emphasize classifier accuracy while providing less evidence of complete-loop edge performance and the integrity of persistent fleet event records. This study addresses this implementation gap by examining three layers of one prototype: image classification, embedded execution, and stored event context. This is a system integration contribution rather than a claim that eye state alarms, dashboards, or MobileNetV2 are new technologies.

This positioning also determines the comparison strategies. External accuracy values are not ranked against the present result because the datasets, split policies, sensors, and endpoints differ. The earlier internal model was not used as a quantitative baseline because its recorded confusion matrix and stated support disagreed. The final model was assessed based on its own internally consistent counts, with the missing comparative and deployment evidence retained as limitations.

3 Data and Method

3.1 Study design and data preparation

The project followed an applied system development design comprising model training, software integration, and prototype evaluation. The development was conducted between March and July 2026. Three units of analysis were maintained: an individual eye image for classification, a camera frame for runtime measurement, and a stored alert for functional inspection. No survey-based acceptance score or formal user study was conducted.

A class-balanced subset of 20,000 images was selected from the MRL dataset. The preparation workflow decoded candidate files using OpenCV, skipped duplicate paths, rejected unreadable files, and sampled equal numbers of open-eye and closed-eye images. Duplicate-path checking does not remove visually duplicate or temporally adjacent images. The numbers rejected during cleaning were not recorded in the audit log.

The MRL filenames encoded closed eyes as 0 and open eyes as 1. The model output represents the closed-eye class as positive. Therefore, preparation must map the source annotation to the training target consistently rather than treating the two encodings as

interchangeable. Table 1 shows the documented balanced partitions. The counts for training and validation follow directly from the reported 75/15/10 division of 10,000 images per class, and the test support is also confirmed by the final confusion matrix.

Table 1. Experimental image partitions

Partition	Open eyes	Closed eyes	Total
Training	7500	7500	15000
Validation	1500	1500	3000
Test	1000	1000	2000
Total	**10000**	**10000**	**20000**

The split was described as a class-stratified image allocation. A subject-disjoint partition was not demonstrated; therefore, the results should be read as an image-level evaluation rather than a confirmed generalization to unseen people. The source dataset included 33 men and four women. No performance breakdown by participant, glasses, illumination, or sensor was reported, and the demographic composition of the selected subset was not provided.

3.2 Model architecture and training

The classifier accepted a 64 × 64 × 1 grayscale eye crop. The channel was repeated three times to provide the backbone input, followed by the preprocessing required for MobileNetV2. The model uses ImageNet initialization, a width multiplier of 0.35, and no original ImageNet classification heads. Global average pooling feeds a dropout layer with a rate of 0.30, a 64-unit dense ReLU layer, dropout with a rate of 0.20, and a single sigmoid output. Figure 1 shows the architecture diagram.

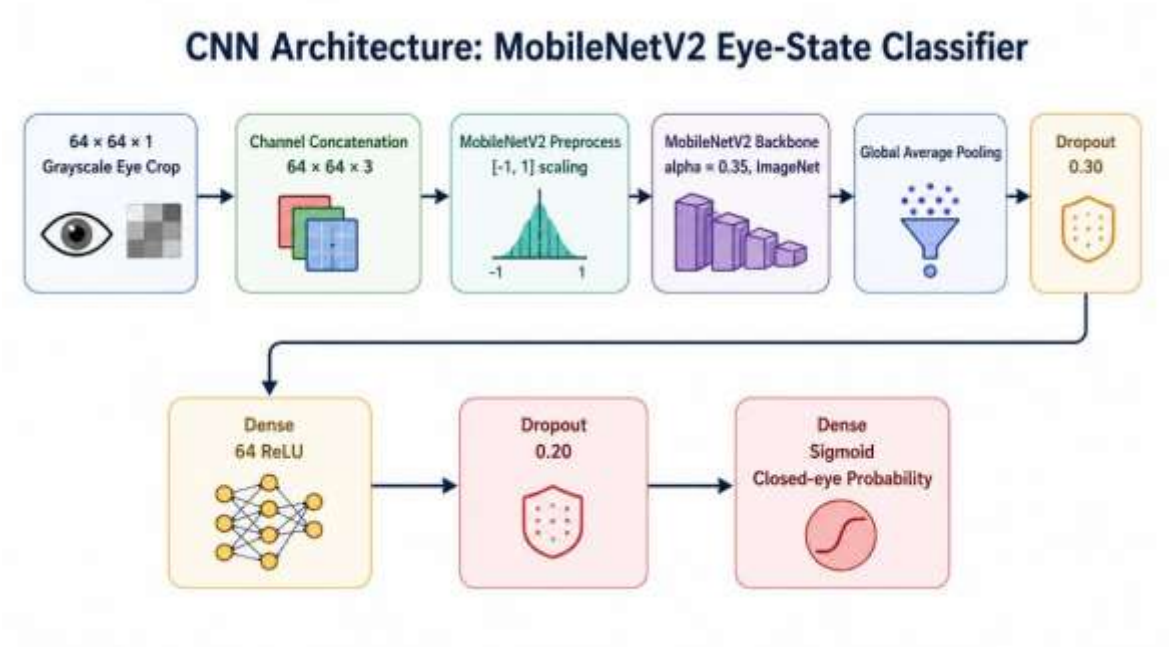


Figure 1. MobileNetV2 eye-state classifier architecture.

The architecture contained 492,257 parameters: 410,208 in the backbone, 81,984 in the dense layer, and 65 in the output layer. The saved fine-tuned configuration reported 392,689 trainable and 99,568 non-trainable parameters. The parameter totals were internally consistent. They described the saved model and did not imply that every parameter was updated during both training stages.

The training used binary cross-entropy and Adam. In the first phase, the pretrained backbone was frozen, whereas the classification head was trained. The second phase unfroze the upper backbone layers from index 110 onward at a lower learning rate, whereas the batch-normalization layers remained frozen. The training data were augmented, cached, shuffled, and prefetched using TensorFlow and Keras, whereas the validation and test data were not augmented. The recorded history spanned 23 epochs. CPU training on a 13th-generation Intel Core i7 laptop with approximately 15.7 GB RAM took approximately two hours and 14 min. The training record did not fully specify the numerical learning rates, batch size, augmentation magnitudes, seed value, or checkpoint identifier. These omissions limit the exact reproduction. The preprocessing description also mentions both scaling to [0,1] and MobileNetV2 scaling to [-1,1]. Reproduction requires checking the actual transformation between these stages, as applying a preprocessing function intended for [0,255] directly to [0,1] inputs would produce a different range. No particular implementation of this intermediate conversion was assumed.

3.3 Camera and alert integration

The runtime captures webcam frames at a resolution of 640 × 480, converts them to grayscale, locates a primary face with a Haar cascade, and searches the upper face region for eyes. The detected eye crops were resized and passed to the TensorFlow Lite model. The export used default optimization with float16 supported types, producing an approximately 0.96 MB file. This export setting does not ensure that all operations are executed in float16 on the device.
A closed-eye score of at least 0.40 was used to produce a closed-eye classification. The documented temporal policy changes the state to WARNING after 20 consecutive closed-eye frames and to DROWSY after 40 consecutive closed-eye frames. The DROWSY state activates an audible alarm that is subject to a cooldown. Screenshot evidence also shows an alert after 18 closed-eye frames, indicating that not all captured evidence establishes the same configuration as the model. Accordingly, the 20/40-frame values are reported as the documented policy, not as thresholds independently confirmed for every screenshot or runtime session.

At a constant rate of 13.95 frames per second, 20 and 40 frames corresponded to approximately 1.43 and 2.87 s, respectively. These are arithmetic conversions and not measured alert latencies. The actual elapsed time depends on frame processing, visibility, and handling of frames without valid eyes. The two-eye fusion rule, counter-reset behavior during tracking loss, and cooldown duration were not formally specified in this prototype. These details require explicit specification before the temporal behavior can be reliably reproduced.

3.4 Fleet event records and dashboard

The detector and dashboard are separate processes that share the SQLite storage. The events included a timestamp, event type, frame count, screenshot path, and available company, device, and vehicle context. FastAPI provides administrative views for managing fleet entities, viewing event history and analytics, and accessing report export. Figure 2 shows the processing sequence. It summarizes the data flow rather than a cloud or multi-device synchronization architecture.

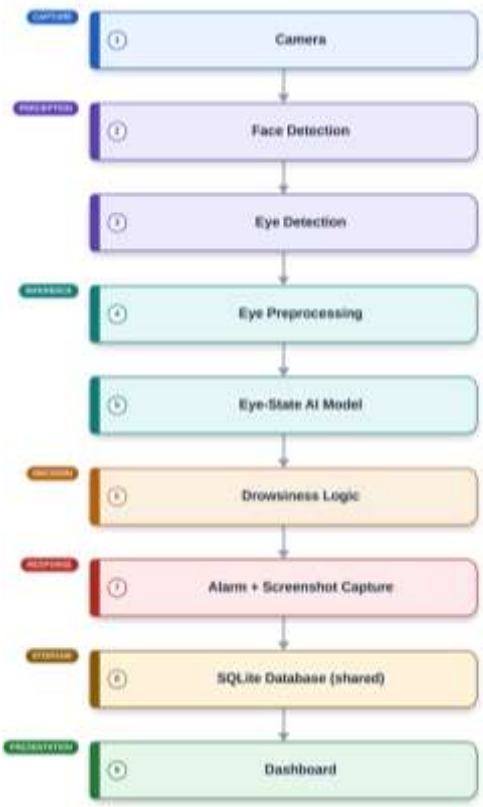


Figure 2. Camera processing and event-record workflow.

The application includes a Super Admin interface and a company-facing portal. The observed company session demonstrated navigation and access to displayed records. Query-level isolation across companies was not audited independently. Automatic facial identity recognition was removed from the system, and records used the Detected Person rather than a verified driver's identity. This design reduces identity claims but does not anonymize screenshots that contain recognizable faces.

3.5 Evaluation protocol

The final classifier was evaluated on 2,000 eye images, with 1,000 examples per class and a decision threshold of 0.40. Closed eyes were considered positive. Accuracy, class-wise precision, recall, F1, and specificity were calculated from the reported confusion matrix. The ROC-AUC was computed from the classifier's continuous output scores.

The hardware evaluation comprised one Raspberry Pi 5 session with a Logitech webcam, lasting 202.4 s and processing 2,824 frames. A separate Windows CPU benchmark of the classifier was run over 300 iterations. Functional evidence was reviewed for camera display, alarms, screenshot references, event persistence, navigation, and export. This design assesses prototype operation; however, it does not provide a labelled continuous video test of drowsiness episodes, a controlled driving trial, or a measurement of accident prevention.

4 Results and Discussion

4.1 Eye image classification

Table 2 presents the final confusion matrices. The classifier correctly identified 973 open-eye and 984 closed-eye images, resulting in 1,957 correct predictions out of 2,000. It misclassified 27 open eyes as closed and 16 closed eyes as open eyes. These counts reproduce a reported

accuracy of 97.85%. Table 3 separates the two classes and reports the corresponding threshold-dependent metrics for each class.

Table 2. Confusion matrix at a closed-eye threshold of 0.40

True class	Predicted open	Predicted closed	Support
Open eyes	973	27	1000
Closed eyes	16	984	1000

Table 3. Final classifier results

Measure	Value
Accuracy	0.9785
Open-eye precision / recall / F1	0.9838 / 0.9730 / 0.9784
Closed-eye precision / recall / F1	0.9733 / 0.9840 / 0.9786
Specificity for the closed-eye target	0.9730
Macro F1	0.9785
Closed-eye false-negative rate	0.0160
ROC-AUC	0.9977

Closed-eye sensitivity exceeded open-eye specificity by 1.1 percentage points. The chosen threshold favors the detection of closed eyes, but there was no matched evaluation at 0.50 or a documented threshold-selection curve from which to quantify that trade-off. A false-negative eye image is not equivalent to a missed drowsiness episode. Temporal accumulation may compensate for one erroneous frame, whereas correlated errors or failed eye localization may persist across several frames.

The ROC-AUC of 0.9977 indicates strong score discrimination within the test partition. Extreme sigmoid scores do not establish calibration or reliable confidence in new cameras. The class-balanced test set also differs from a continuous stream dominated by ordinary open-eye frames; therefore, the observed precision should not be transferred directly to the deployed warning precision.

4.2 Training behaviour and comparison limits

Figure 3 shows the training history. The training accuracy generally increased, and the curves showed a lower validation loss later in training than near the beginning. The final portion also shows that the training loss continues to decrease, whereas the validation loss rises from its minimum. This pattern supports validation-based checkpoint selection and cautions against the assumption that the last epoch is necessarily the best.

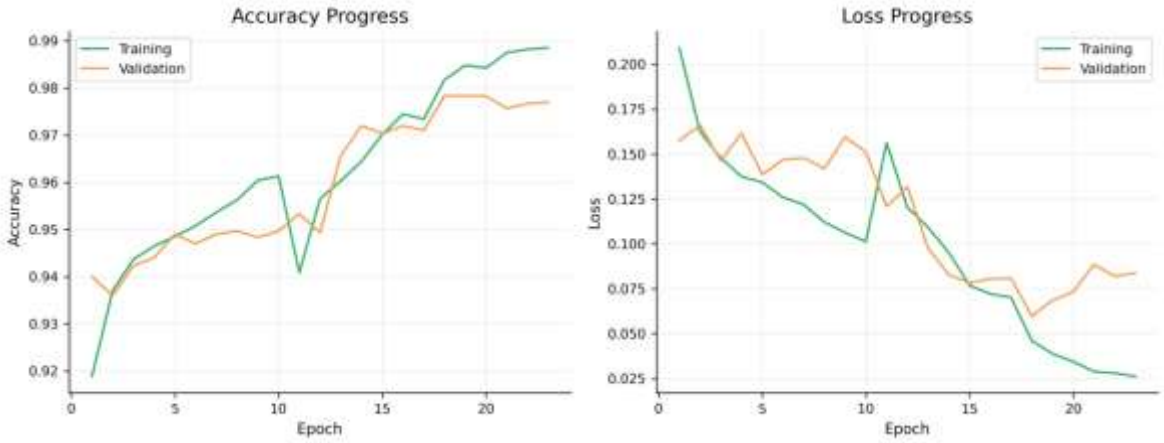


Figure 3. Training and validation curves over 23 recorded epochs.

The earlier model was excluded from the quantitative comparison. Its recorded confusion matrix contained 500, 27, 16, and 473 predictions, summing to 1,016, whereas its classification report stated 2,000 observations and 95.85% accuracy. These records did not establish a coherent matched baseline. Reporting a two-percentage-point improvement over that model would therefore overstate what could be verified. The present results establish the final classifier performance within the stated partition, not an experimentally isolated gain from MobileNetV2 or fine tuning.

4.3 Embedded runtime

Table 4. Runtime measurements and their scope

Measurement	Reported result	Scope
Windows CPU classifier latency	0.64 ms	Local benchmark over 300 runs
Windows CPU reciprocal throughput	1557.4 calls/s	Classifier only, not video frame rate
Raspberry Pi 5 session	2824 frames in 202.4 s	One desk-based camera session
Raspberry Pi 5 frame rate	13.95 frames/s	Complete processing loop
Raspberry Pi 5 frame time	71.68 ms	Average frame processing
Raspberry Pi 5 eye inference	1.24 ms/call	Eye classifier only

As shown in Table 4, the Raspberry Pi session totals (2,824 frames in 202.4 s) correspond to approximately 13.95 frames per second, consistent with the reported frame rate. The reciprocal of this rate was approximately 71.68 ms. The 1.24 ms classifier call is much shorter than the complete frame time, suggesting that optimizing only the neural network may have a limited effect on the total throughput. The available measurements do not attribute the remaining time to individual operations, and there may be more than one eye classifier call in a frame.

These results demonstrate the execution on the target hardware during a short session. However, they do not establish sustained performance under thermal load, moving-vehicle vibration, lighting changes, multiple faces, or background dashboard activity. CPU utilization, RAM consumption, energy use, and end-to-end alarm latency were not evaluated. The Windows throughput value is also a classifier benchmark, not evidence that the application processes more than a thousand camera frames per second.

4.4 Functional operation and record quality

The retained screenshots and test descriptions show the camera capture, face and eye boxes, open/closed labels, alert display, and dashboard records. A physical buzzer was activated during the test. Screenshot persistence was indirectly supported by event references and evidence thumbnails. Fifteen alert rows were present in the observed company context. These observations support the functional feasibility without establishing exhaustive pass rates for all runtime conditions.

Three of the 15 events appeared as UNKNOWN_VEHICLE. This corresponds to 20% of the small observed record set, not the estimated fleet-wide error rate. Nonetheless, this is important because incorrect vehicle attribution can distort the analytics, even when the underlying detection is correct. Missing or incorrect device configurations are plausible causes, but they were not established by a configuration audit. Event creation should validate this association before the records are used for vehicle-level decisions.

The report interface and download controls were demonstrated, but the exported contents were not independently checked against the database rows. Testing of report compilation was not completed. Similarly, visible company-scoped navigation does not establish a secure database isolation. Therefore, functional readiness should distinguish a working interface from verified export integrity, authorization, and complete system testing.

4.5 Interpretation and limitations

The results are consistent with the feasibility of CNN-based eye classification described by Florez et al. [5] and the embedded direction demonstrated by Florez et al. [6] and Reddy et al. [11]. The current implementation connects its detector to an event-review workflow while using fewer behavioral cues than the eye-and-mouth approach. The available evidence cannot determine whether this simplification improves reliability or only reduces computational requirements. A comparison would require matched participants, recordings, thresholds and hardware.

The largest generalization concern is the difference between the infrared training source and RGB webcam. Replicating a grayscale channel does not eliminate the sensor difference. The MRL also contains repeated observations from a small participant group, and the experimental split does not demonstrate subject independence. Reviews of the field identify subject-dependent and within-dataset evaluations as sources of optimistic performance estimates and recommend subject-independent and cross-dataset testing [9]. The reported 97.85% therefore describes the documented image partition and should not be interpreted as the performance of unseen drivers.

The alert logic has an additional limitation: frame counts measure processed observations rather than stable elapsed times. The conflicting 18-frame screenshot and 20/40-frame policies need to be reconciled. Missing eye behavior and two-eye disagreement also require explicit handling. A monitoring system should not silently treat tracking loss as proof of open eyes. Future evaluations should report detection availability alongside alert sensitivity, false alarms per hour, and time to warning.

No labelled driving sequences, external dataset evaluation, repeated-seed analysis, or statistically supported architectural comparisons were available. The limited hardware session cannot establish collision reduction or operational safety. Consent, screenshot access, retention periods, and company-level authorization must also be addressed before deploying the system in the workplace. Removing identity recognition does not eliminate the privacy implications of recording a person in a vehicle.

5 Conclusion

This study implemented an embedded eye-state monitoring prototype using a fleet event dashboard. Within the documented image-level test partition, the MobileNetV2 classifier achieved 97.85% accuracy on 2,000 eye images, with closed-eye sensitivity and specificity of 98.40% and 97.30%, respectively. A short Raspberry Pi 5 session demonstrated approximately 13.95 frames per second, and the inspected application records contained 15 events with evidence and fleet context. Three unresolved vehicle associations showed that record integrity requires attention, along with classification performance.

The findings support the feasibility of the prototype, but drowsiness detection in a moving vehicle was not validated. The next steps should be to reconcile the temporal configuration, recover the complete training and partition records, and evaluate unseen participants with the deployed sensor. Continuous video testing should measure missed episodes, false alarms per hour, tracking failures, and warning latency. Longer hardware tests and database authorization checks are required before fleet trials. The earlier model should remain outside quantitative comparisons until its evaluation records have been reconciled.

Data availability

The MRL Eye Dataset is publicly accessible through the dataset website listed in the references. The selected partition manifest, model checkpoint, prediction vectors, configuration files, and runtime logs were not available as public reproducibility packages. Their release, subject to applicable data terms and privacy requirements, would allow for verification beyond the aggregate results reported here.

Acknowledgements

The authors acknowledge the support of the Faculty of Computer Science and Information Technology at Hormuud University, Somalia.

Conflict of Interest

The authors confirm that there is no conflict of interest related to this manuscript.

References

[1] World Health Organization. (2023). Global status report on road safety 2023. https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023

[2] Ramzan, M., Khan, H. U., Awan, S. M., Ismail, A., Ilyas, M., & Mahmood, A. (2019). A survey on state-of-the-art drowsiness detection techniques. IEEE Access, 7, 61904-61919. https://doi.org/10.1109/ACCESS.2019.2914373

[3] Albadawi, Y., Takruri, M., & Awad, M. (2022). A review of recent developments in driver drowsiness detection systems. Sensors, 22(5), 2069. https://doi.org/10.3390/s22052069

[4] Kolus, A. (2024). A systematic review on driver drowsiness detection using eye activity measures. IEEE Access, 12, 97969-97993. https://doi.org/10.1109/ACCESS.2024.3424654

[5] Florez, R., Palomino-Quispe, F., Coaquira-Castillo, R. J., Herrera-Levano, J. C., Paixao, T., & Alvarez, A. B. (2023). A CNN-based approach for driver drowsiness detection by real-time eye state identification. Applied Sciences, 13(13), 7849. https://doi.org/10.3390/app13137849

[6] Florez, R., Palomino-Quispe, F., Alvarez, A. B., Coaquira-Castillo, R. J., & Herrera-Levano, J. C. (2024). A real-time embedded system for driver drowsiness detection based

on visual analysis of the eyes and mouth using convolutional neural network and mouth aspect ratio. Sensors, 24(19), 6261. https://doi.org/10.3390/s24196261

[7] Shiau, J.-Y., Nishiyuki, K., Nagae, S., Yabuuchi, T., Kinoshita, K., & Hasegawa, Y. (2019). Driver drowsiness estimation by parallel linked time-domain CNN with novel temporal measures on eye states. In 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (pp. 937-942). https://doi.org/10.1109/EMBC.2019.8857291

[8] Fonseca, T., & Ferreira, S. (2025). Drowsiness detection in drivers: A systematic review of deep learning-based models. Applied Sciences, 15(16), 9018. https://doi.org/10.3390/app15169018

[9] Hassan, O. F., Ibrahim, A. F., Makhlouf, M. A., Hafiz, B., & Gomaa, A. (2026). Intelligent driver monitoring systems: A survey of drowsiness detection technologies for road safety. Artificial Intelligence Review, 59, 137. https://doi.org/10.1007/s10462-026-11505-w

[10] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510-4520). https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html

[11] Reddy, B., Kim, Y.-H., Yun, S., Seo, C., & Jang, J. (2017). Real-time driver drowsiness detection for embedded system using model compression of deep neural networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 438-445). https://doi.org/10.1109/CVPRW.2017.59

[12] MRL. (n.d.). MRL Eye Dataset. https://mrl.cs.vsb.cz/eyedataset.html

[13] Arakawa, T. (2021). Trends and future prospects of the drowsiness detection and estimation technology. Sensors, 21(23), 7921. https://doi.org/10.3390/s21237921