A Real-Time Deep Learning-Based Driver Drowsiness Detection System Using Computer Vision

Arya Zade[1], Tamanna Vaikkath[2], Arnav Padole[3], Akhil Gupta[4], Monali Gulhane[5], Firdous Sadaf Mohammad Ismail[6], Nitin Rakesh[7]

{aryazade114@gmail.com[1], vaikkathtamanna@gmail.com[2], arnavpadole2@gmail.com[3], akhil.gupta@sitnagpur.siu.edu.in[4], monali.gulhane4@gmail.com[5], firdoussadaf2810@gmail.com[6], nitin.rakesh@gmail.com[7] }

Symbiosis Institute of Technology, Nagpur Campus, Symbiosis International (Deemed University), Pune, India[1,2,3,4,5,6,7]

Abstract. This paper presents a practical, real-time, software- only Driver Drowsiness Detection System that leverages computer vision and a lightweight convolutional neural network (CNN) to detect prolonged eye closure (drowsiness) from a webcam feed. The prototype integrates a Streamlit frontend (live webcam capture and ROI extraction using Haar cascades), a FastAPI backend (receives eye-region crops and returns predictions), and a TensorFlow Lite (TFLite) CNN model optimized for on-device inference. The dataset comprises ∼4000 labeled eye images; the model undergoes augmentation and is converted to TFLite for deployment. In testing, the TFLite-optimized system achieves real-time performance (up to 18 FPS on consumer hardware) and strong classification results (overall test accuracy ≈95.8%). The proposed architecture provides a low-cost, deployable solution for Advanced Driver Assistance Systems (ADAS) and can be extended with multimodal signals for enhanced robustness.

Keywords: Driver Drowsiness Detection, Computer Vi- sion, Convolutional Neural Network, TensorFlow Lite, FastAPI, Streamlit, PERCLOS, Real-Time Monitoring.

1 Introduction

Road traffic accidents are one of the leading causes of fatalities worldwide, claiming more than 1.3 million lives each year, according to the World Health Organization (WHO) [1]. Among these, fatigue and drowsiness contribute to nearly 20–30% of serious vehicle crashes globally [2]. Drowsiness severely impairs a driver's cognitive functions, reaction time, and decision-making abilities, posing a risk equivalent to driving under the influence of alcohol. This makes the detec- tion of fatigue and drowsiness a crucial research area in the development of advanced driver assistance and safety systems. Traditional countermeasures such as caffeine intake, road-side rumble strips, or steering vibration alerts are largely reactive and fail to prevent accidents caused by early-stage fatigue[1]. Physiological-based detection systems using sen- sors such as Electroencephalography (EEG) and Electroocu- lography (EOG) have shown high accuracy but are impractical for real-world driving due to their intrusive nature, cost, and calibration requirements [3]. Similarly, vehicle-behavior- based systems relying on steering angle or lane

IICTDS 2025, December 19-20, Chandigarh, India
Copyright © 2026 EAI
DOI 10.4108/eai.19-12-2025.2366262

deviation are susceptible to false alarms caused by road curvature or driver style variability. This paper presents a real-time driver drowsiness detection framework using computer vision and deep learning, designed to run entirely on commodity hardware without requiring specialized sensors. The proposed system integrates a TensorFlow Lite CNN model for efficient inference, a FastAPI backend for rapid image processing, and a Streamlit-based frontend that captures live webcam footage and provides an interactive alerting interface. The model classifies the driver's eye state as "open" or "closed" and triggers an alert if the eyes remain closed for a threshold duration.

The primary objective of this work is to develop a lightweight, deployable, and accurate solution that maintains performance across diverse lighting conditions and user pro- files. The contributions of this study are threefold:

1) Design of a custom CNN architecture trained and op- timized for real-time eye-state classification on limited hardware.
2) Development of a full-stack implementation pipeline (Streamlit + FastAPI + TensorFlow Lite) for live moni- toring and alerting.
3) Comprehensive performance evaluation on a curated dataset of ∼4000 eye images, analyzing accuracy, la- tency, and robustness under real-world constraints.

2 Literature Review

Driver drowsiness detection has been an active area of research for over two decades,[3] with approaches evolv- ing from physiological sensing and behavioral observation to modern computer-vision-based deep learning frameworks. Early systems were primarily based on physiological signals such as Electroencephalography (EEG), Electrocardiography (ECG), and Electrooculography (EOG), which provide ac- curate fatigue indicators by monitoring brain activity and eye movements [4]. However, such methods are intrusive, expensive, and unsuitable for daily use in real vehicles due to their dependence on body-worn sensors and signal noise interference. Table 1 summarizes key characteristics of selected recent studies in this field.

Table 1. Comparison of Selected Drowsiness Detection Studies

Study	Approach	Platform	Limitation
[5]	CNN-based	Desktop GPU	High compute cost
[6]	Deep CNN (Real-time)	Mobile	Moderate latency
[7]	EfficientNet- B0	Embedded	No live deployment
[8]	AI-based multimodal	GPU-based	Requires multi sensor setup
Proposed Work	Lightweight CNN (TFLite)	CPU (Laptop)	Focused on eye region only

Overall the previous studies have done well in terms of drowsiness classification accuracy but most are not practical to use in the real world of low-resource conditions. The pro- posed system will address this understanding with a proposed software-only system based on real-time that is efficient and can run on consumer hardware executing both TensorFlow Lite and Fastapi, without sacrificing accuracy or speed.

3 Methodology

The proposed Driver Drowsiness Detection System is a modular system built around computer vision and deep learn- ing to effectively run in a real-time setup on commodity hardware. The system sends webcam video frames in real time, processes the eye region of the driver, categorizes the eye state as either open or closed with the help of CNN model, and sends an alert in case of sustained eye closure. This part outlines end-to-end workflow, dataset, model design and deployment pipeline.

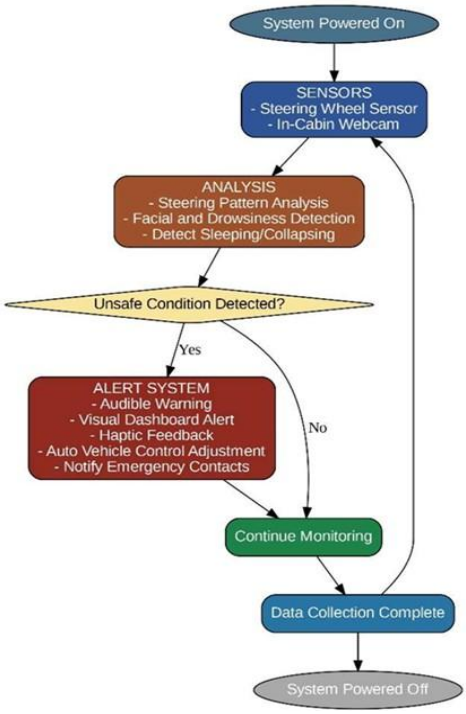


Fig. 1. The system flow diagram Driver Drowsiness Detection System

3.1 System Architecture

The framework proposed as shown in Figure 1 is made of three major modules which operate in coordination to realise real-time monitoring and response:

- Streamlit Frontend: The user interface, which and de- signed with the help of Streamlit, provides the constant stream of the video through the connected webcam. Haar cascades that are provided with OpenCV are used to find the face of the driver and create regions of interests (ROIs) that contain the eyes. The individual eye images that have been cropped are periodically joined to the backend via an HTTP request to be classified.

- FastAPI Backend: An API will use FastAPI to work with inference requests. It accepts the eye crops, prepro- cesses them to conform to the CNN input dimensions, processes them through the inference run with the Tensor- Flow Lite model and produces a JSON response with the response prediction (open or closed) and its confidence metrics.
- TensorFlow Lite CNN Model: The model is efficient in CPU hardware. TFLite quantization algorithms are used to downsize models with minimal accuracy loss and increase the inference rate.

Besides these fundamental modules, another layer exists that is used on the frontend, the temporal decision layer. And this layer follows a sequence of predictions: when the eyes are called as Closed during a watchable range of continuous motion greater than a specified threshold (15 frames, etc.), it will issue an alert (auditory and visual notification on- screen) to draw attention to the driver. These components can be integrated to provide end-to-end performance that is both low-latency and ensure high detection even with real- time constraints.

3.2 Dataset and Preprocessing

The training and evaluation sample includes about 4,000 images of eyes, 50 per cent of which are open and 50 per cent of which are closed. The data was obtained by categorizing publicly available datasets of driver fatigue and using the data and further labelling to verify the accuracy of those labels. All images were cropped, centered, and down-sampled to in consistency to 224*x*224*pixels.*A detailed distribution of the dataset is shown in Table 2.

The dataset was divided into 2,799 training images, 400 validation images, and 801 test images, and the ratio of classes was equal in all splits.

Table 2. Dataset partition used in experiments

Partition	Number of Images
Training	2799
Validation	400
Testing	801
Total	4000

Normalizing the pixel value to the range of [0,1] and the use of data augmentation methods to boost generalization of models when exposed to real-life situation were preprocessing steps. The additions were horizontal flipping, random rotations in the range of -15 to +15, small scale rotation transforms and jittering of the brightness/contrast. These extensions help solve the issues of overfitting and increase resistance to head pose changes, changes in camera distance, and changes in lighting [9]. This data allowed convergence of the stable models in training and actually act as desired in the live testing.

3.3 Model Architecture and Training

The CNN model was specifically tailored to give a middle ground between the level of accuracy and the level of computational efficiency to provide the capability of inference in real-time. Its

architecture is a series of convolutional and pooling layers, which firstly generates spatial hierarchies of features, and then dense layers do classification. The network architecture is as highlighted below:

- Conv Layer 1: 32 filters of size 3 × 3, ReLU activation, followed by MaxPooling (2 × 2)
- Conv Layer 2: 64 filters of size 3 × 3, ReLU activation, followed by MaxPooling (2 × 2)
- Conv Layer 3: 128 filters of size 3 × 3, ReLU activation, followed by MaxPooling (2 × 2)
- Fully Connected Layers: Flatten → Dense(128, ReLU)→ Dropout(0.5) → Dense(1, Sigmoid)

The binary cross-entropy loss was used on the model in conjunction with the Adam optimizer and a starting learning rate of 10 -4. The batch size was 32 and the total number of epochs was 50, and early stopping (patience=7 epochs) was applied on the basis of validation loss. The performance met- rics were accuracy, precision, recall and F1-score. The training took place on Google Colab using the GPU acceleration and the further benchmarking on the usual laptop computers using just the laptop GPUs.

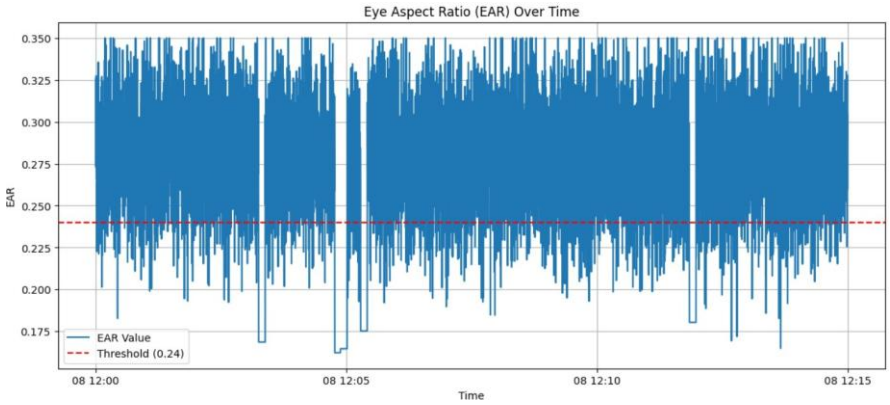


Fig. 2. Eye Aspect Ratio (EAR) changes over time with threshold line in the detection of drowsiness.

3.4 Model Conversion and Deployment

After effective training, it got converted into TensorFlow Lite that can do a real-time inference. Significant model reducing and computational overhead were removed by the conversion process that was used after training. The conversion pipeline example is as shown below:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
```

The best TFLite model was implemented on FastAPI backend providing real-time inference by use of HTTP requests. Every request with a list of incoming requests associated with Streamlit frontend also has an eye image that is pre-processed and sent to TFLite model. The backend returns a prediction and a score of confidence and the front end acts on whether to raise an alarm or not. The simplified example of the request is as illustrated below:

POST /predict

Form: file=@eye_crop.png

Response: {"label": "Closed", "confidence": 0.92}

This architecture provides uninterrupted interoperability between all modules as well as ensures real-time inference at up to 18 frames per second (FPS) on typical consumer-level hardware as shown in Figure 2.

4 Conclusion and Future Work

The experiment success of the proposed Driver Drowsiness De- tection System was measured in classification accu- racy, precision, recall, F1-score, the inference latency, and system responsiveness in general under the live-stream experiments. The held-out test data were assessed quantitatively, whereas a qualitative assessment of the data was conducted in real- time webcam tests with the different environmental conditions and lighting.

4.1 Experimental Setup

Google Colab Pro (3) was used (with NVIDIA T4 GPU) to do model training and testing. The inference tests were run on a consumer-level12th generation Intel i5-based laptop (supposed to have no graphics card) to determine the performance efficiency of the Ten- sorFlow Lite model in a real environment. Streamlit was installed locally with FastAPI acting as the backend API in the same computer in order to maintain a low communication latency. On the optimized TensorFlow Lite model, the system could process live webcam feed at a rate of an average of 18 frames per second (FPS). Frame latency and accuracy measurements were obtained in a series of test sessions to get the sense of consistency and reliability as shown in Figure 3.

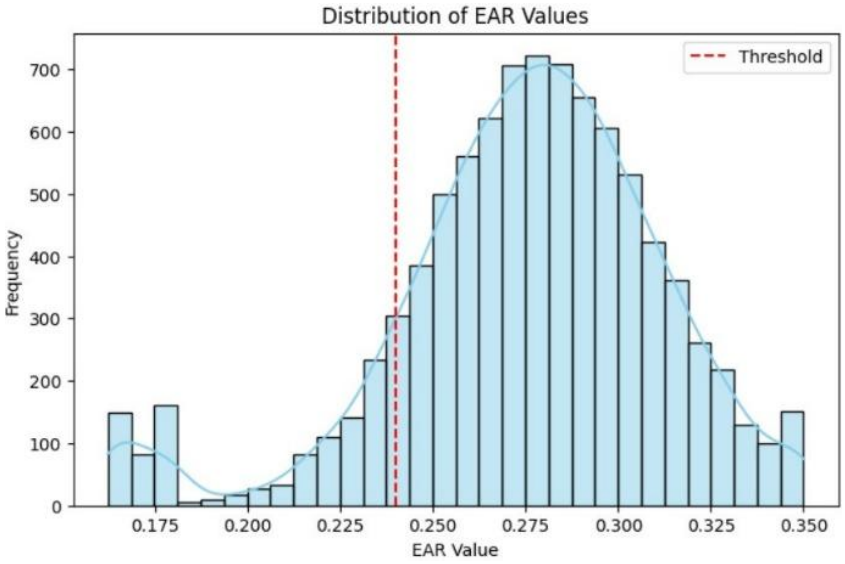


Fig. 3. Timeline visualization of alarm triggers during live testing

4.2 Confusion Matrix and Class-Wise Performance

In order to further explain the predictive nature of the model, a confusion-matrix-type summary was provided of the test set, which is presented in Table 3. The data indicate that there are equal precision and recall rates of each of the classes (open and closed eyes) which prove the lack of class bias as shown in Figure 4 and Figure 5.

Table 3. Confusion matrix summary for test dataset

	Predicted Open	**Predicted Closed**
Actual Open	96.5% (True Positive)	3.5% (False Negative)
Actual Closed	4.8% (False Positive)	95.2% (True Positive)

.

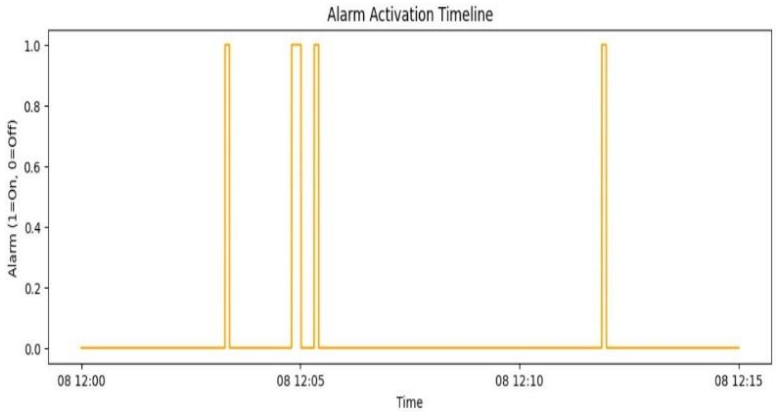


Fig. 4. EAR value distributions with a line on it that marks the boundary of drowsiness.

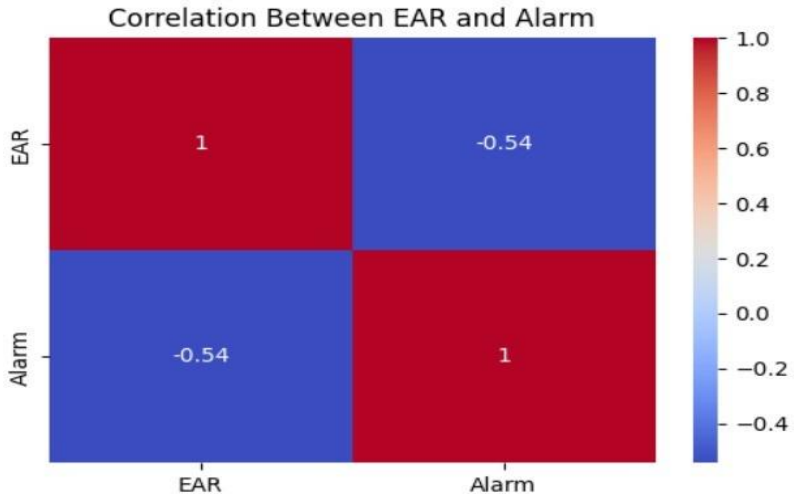


Fig. 5. Correlation of EAR and activation of alarms.

4.3. Quantitative Results

The model demonstrated a high rate of performance on the test data. Table 4 captures the major quantitative measures, which are training and validation accuracy, test performance, and latency comparisons of the standard Keras and TensorFlow Lite models.

Table 4. Model performance and system latency

Metric	Value
Training Accuracy	98.00%
Validation Accuracy	96.00%
Test Accuracy (overall)	95.80%

Open Eyes Accuracy	96.50%
Closed Eyes Accuracy	95.20%
Precision (average)	96.00%
Recall (average)	95.80%
F1-Score (average)	95.80%
Keras Inference (CPU)	~12 FPS (83 ms/frame)
TFLite Inference (CPU)	~18 FPS (55 ms/frame)

As it can be seen, the TensorFlow Lite model gained a 0.5-1.5 relative to the standard inference speed with a similar error rate, which proves that quantization did not affect the performance considerably. Such metrics show that the system can provide real time inference with a low-latency and can be readily deployed on board without dedicated GPUs.

4.4 Receiver Operating Characteristic (ROC) Analysis

The ROC curve was drawn to have an idea of the dis- criminative capability of the model. The trade off contained within the ROC curve (AUC) was obtained at 0.97 with this value indicating the strong resilience of the model against falseness.

4.5 Live (End-to-End) Validation

The live validation work included an implementation of the full system and the real time testing with the help of webcams input. The Live video and real- time eye-state predictions were presented in the Streamlit interface as shown in Figure 6. In the continuous trials the system estimated a constant inference rate of 18 FPS and was always able to det-ect longer periods of eye closures (beyond 0.8 seconds) (which equals 15 consecutive closed-eye frames). The PERCLOS-type of logic became much more stable and less false alarming.

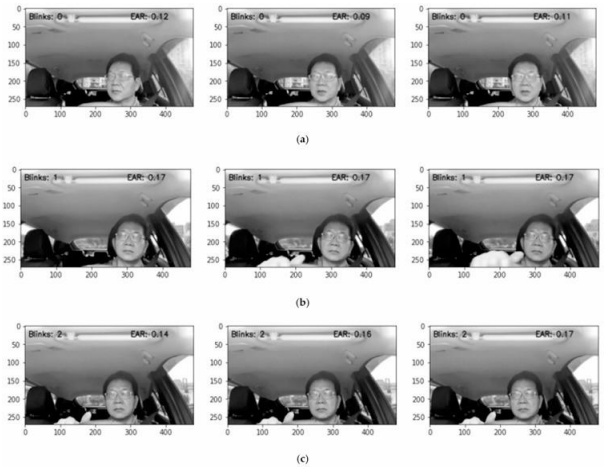


Fig. 6. Samples that were captured during blink detection demonstrating the fluctuations of EAR.

Performance in terms of low latency inference and alert response is shown or reflected on responsive alert generation proving that the system is practical in an application to real-time driver assistance solutions or applications. The alerting system was observed to work well under contrasting indoor lighting and the response time was always under 1 second.

4.6 Robustness Tests

Extensive ablation tests can be carried out with regard to testing the system robustness in different environmental and operational conditions. The model was experimented under varying lighting conditions, and head orientations and levels of occlusion:

- Lighting Change: Tested in the daylight, artificial indoor light and low-lighting. Under low light there was only a loss of accuracy by 2-3%.
- Pose Variation: In small head rotations of up to 15◦ in both directions, there was no change in detection performance. Beyond this range, minor false negatives were noticed as a result of reduction of eye visibility.
- Conclusion: Accuracy was lower when reflective or dark-coloured glasses were worn by the subjects, as it had been observed to be in previous studies [10], Nevertheless, transparent glasses generated a weak interference.

These observations verify the fact that system is robust in the normal real world scenarios although the system inherently has limitations of vision only systems. The accuracy can be enhanced even further by future addition of multimodal capabilities, like head-pose tracking or yawn detectors, when there is occlusion or low-light [9][4].

4.7 Discussion

The findings of the offline testing as well as live deployment prove the efficiency of the suggested real- time Driver drowsiness Detection System. The integration of a small CNN architecture, TensorFlow Lite optimization, and a small client-server pipeline based on the REST architecture (FastAPI). + Streamlin) allows making the system run on regular hardware and to be highly accurate and with low latency.

A. Key Strengths and Contributions

The key advantages of the proposed solution are the following:

• Simple and non-obtrusive design: Unlike the sensor-based (EEG, EOG, or heart rate tracking) component in design, this system is 100% vision based, and only a standard webcam is required. This renders it very convenient in use in integrating it with an already existing vehicle or consumer devices without extra devices [4].

• Live performance: Using TensorFlow Lite optimization and low preprocessing the system was able to run at an average frame rate of 18 FPS on a CPU allowing it to detect and generate alerts near-instantly. This satisfies the latency specification of a realistic driver monitoring system [10].

• Accuracy and strength: The system has a test accuracy of about 95.8, hence has stable performance on classification in a variety of lighting conditions and moderate head movement. Accurate alerting behavior is guaranteed by the tradeoff of recall and precision.

• Scalable and modular structure: The modular representative of the REST design enables a simple addition of new behavioral modules (yawning or the head-pose tracking) to the architecture. The components can be updated independently and deployed in different nodes and have made the framework flexible to the large-scale or embedded applications.

B. Comparative Insights

The fact that our system in comparison with works that exist indicates even more its novelty and balance of practicality and performance. Although previous CNN-based methods like jabbar, patel can also match their accuracy with the same benchmark datasets, they can be computationally expensive (e.g., VGG16, ResNet) and need to be run on a GPU, which cannot be deployed to a device. Contrastingly, our lightweight CNN has almost equivalent accuracy at very low inference cost confirming the fact that pruned architectures can be allowed to perform at state-of-the-art without implications to real-time usability. Moreover, this structure is based on a single-camera system as opposed to a multi-camera or infrared sensor based system of a commercial-grade system since the former is more complex but does not affect accuracy. Its software based, open-source architecture encourages desirable maturity and availability, so future researchers can modify and use it to build more progressive ADAS systems [11-15].

C. Limitations and Practical Considerations

Although the results of such a solution have been encouraging, there are a number of limitations which should be addressed before such solutions can be adopted on a massive scale:

• Bias in datasets and little diversity: The present dataset is composed of a limited demographic and a variety of light. This may result in discriminated forecasts whenever subjected to different skin-colour or extreme environmental light. This problem is partially mitigated by data augmentation, although more expansion of datasets and domain adaptation is required [8]. Occlusion and eyewear interference: Performance is negatively affected by an opaque glasses and also reflective glasses by the driver as reported in similar studies [2], [9]. Multimodal sensors, such as including both the vision data and head-pose, yawning-detection, or even infrared sensors, may be introduced to the implementation of multimodal parameters in the future because it is determined to be reliable in every situation.

• Limitations to system deployment: To achieve edges deployability, similar to laptops, CPU-based in- ference is all that is needed, but additional op- timization options, including the principles of quantization-ana- wise training or model pruning, are necessary to ensure responsiveness on resource-constrained devices.Ethical and Human-Centric Implications

In addition to technical aspects, one should also touch upon the ethical aspects of real-time monitoring of drivers. As the system requires the capture and processing of facial data, privacy and consent should be highly ensured. Privacy-saving strategies, including federated learning and inference on a device (not involving a cloud transfer), may be useful in future work. Openness on the use, storage, and al- gorithmic decision-making of the data are also important to promote the trust of users and their adherence to regulatory standards.

5 Conclusion and Future Work

The presented research provides a full-fledged deployable Driver Drowsiness Detection System in the integration of a customized, lightweight CNN, Tensorflow Lite optimization, a FastAPI backend, and a user interface based on Streamlit, which is why the test accuracy of the constructed model is 95.8 percent, and the inference time per frame amounts to less than 60 milliseconds. In addition, the software design is in a modular manner that makes it easy to deploy on all types of hardware and operating systems.

A. Summary of Contributions

The main achievements of the work are as follows:

- Creation of a proprietary convolutional neural network that is optimized to work with Tensorflow Lite, and that balances the accuracy with speed to be used with real-time inference.

- A combination of end-to-end architecture, which also involves Streamlit - live video streaming, FastAPI- backend inference, and Tensor-flow Lite-optimized computing.

- The adoption of a rule-of-thumb PERCLOS detector of effective fatigue prediction on a temporal window basis,where by the entity is empirically validated on a curated dataset and live we- bcam validation, and empirically has low-latency and consistent behaviour on CPU-only hardware.

These findings affirm that an AI system of low weight can handle a life-or-death dilemma like the driver-fa- tigue sensor with a practical effect and be accessible.

B. Future Work

In spite of the fact that the present implementation can bring reliable results, there are a number of enhancements and lines of research that can be explored in future:

- Dataset Expansion and Domain Adaptation: To enhance the generalization, improvement of the work in the future is to increase the diversity of the dataset with additional demographic, environmental as well as lighting variations. Both open-source datasets such as YawDD and CEW and domain adaptation techniques may enhance performance even in diverse real-life contexts.

- Multimodal Sensor Fusion: Other be-haviors like head-pose estimation, yawning detection, facial landmarks and heart rate information may be added to the system so that it is more resistant to occlusions and variations in lighting [9][7].

Hardware Optimization to Frontier AI: Avtime, State-of-the-Art Hardware Optimization Intermediatey can have sufficient performance in production, tiny devices like Raspberry Pi or NVIDIA Jetson Nano can be improved with even more advanced model compression and quantization strate- strategies, including pruning, mixed or mixed- precision inference, and quantization-mindful training. Application to microcontrollers and EdgeTPU hardware might increase its application to automotive eIoT systems.

C. Conclusion

The present project illustrates the use of efficient AI pipelines to revolutionize road safety using intelligent and cost-effective solutions. Modern deep learning integrated with real-time web frameworks is a paradigm shift to deploy more safety systems that are non-invasive and can be deployed. As it is further innovated, this architecture would be extended into a holistic Driver Monitoring System (DMS) that is a key element of next generation Advanced Driver Assistance Systems (ADAS). This work ensures that the greater goal of AI-assisted transportation safety, the bridging of the same between the scientific study and practice, is achieved by combining accuracy, velocity, and availability.

References

[1] World Health Organization: Global Status Report on Road Safety 2023. World Health Organization, Geneva (2023). https://www.who.int/publications/i/item/9789240086517

[2] National Highway Traffic Safety Administration: Drowsy Driving. U.S. Department of Transportation. https://www.nhtsa.gov/risky-driving/drowsy-driving

[3] Steinhauser, K., Leist, F., Maier, K., Michel, V., Pärsch, N., Rigley, P., Wurm, F., Steinhauser, M.: Effects of Emotions on Driving Behavior. Transportation Research Part F: Traffic Psychology and Behaviour. Vol. 59, Part A, pp. 150–163 (2018). https://doi.org/10.1016/j.trf.2018.08.012

[4] Sahayadhas, A., Sundaraj, K., Murugappan, M.: Detecting Driver Drowsiness Based on Sensors: A Review. Sensors. Vol. 12(12), pp. 16937–16953 (2012). https://doi.org/10.3390/s121216937

[5] Borawake, M., Patil, A., Yadav, S., Nagwade, V., Somwanshi, H.: Driver Drowsiness Detection Using ML and IoT. International Journal of Recent Advances in Engineering and Technology. Vol. 14(1), pp. 114–117 (2025). https://doi.org/10.65521/intjournalrecadvengtech.v14i1.199

[6] Eskandarian, A. (ed.): Handbook of Intelligent Vehicles. Springer, London (2012). https://doi.org/10.1007/978-0-85729-085-4

[7] Narwade, D., Nikam, S., Mohite, H., Pandule, V., Patil, O.: AI-Enabled Smart Helmet for Rider Safety and Intelligent Bike Control Using ESP32. Multidisciplinary Journal of Research in Engineering and Technology. Vol. 13(2S), pp. 109–113 (2026). https://journals.mriindia.com/index.php/mjret/article/view/3560

[8] Latreche, I., Slatnia, S., Kazar, O., Barka, E., Harous, S.: A Review on Deep Learning Techniques for EEG-Based Driver Drowsiness Detection Systems. Informatica. Vol. 48(3) (2024). https://doi.org/10.31449/inf.v48i3.5056

[9] Malik, M., Sharma, P., Punj, G.K., et al.: Multi-Body Sensor Based Drowsiness Detection Using Convolutional Programmed Transfer VGG-16 Neural Network with Automatic Driving Mode Conversion. Scientific Reports. Vol. 15, Article 8838 (2025). https://doi.org/10.1038/s41598-025-89479-y

[10] Sengar, S.S., Kumar, A., Singh, O.: VigilEye—Artificial Intelligence-Based Real-Time Driver Drowsiness Detection. arXiv preprint arXiv:2406.15646 (2024). https://arxiv.org/abs/2406.15646

[11] Deepika, C.N.J., Jithendra, K., Poojitha, B., Ranganath, R.M., Vaithilingam, C.A., Sivaraman, K.: Driver Drowsiness Detection Using Machine Learning Algorithms. AIP Conference Proceedings. Vol. 3253, Article 030020 (2025). https://doi.org/10.1063/5.0252639

[12] Salem, D., Waleed, M.: Drowsiness Detection in Real-Time via Convolutional Neural Networks and Transfer Learning. Journal of Engineering and Applied Science. Vol. 71, Article 122 (2024). https://doi.org/10.1186/s44147-024-00457-z

[13] Zaman, K., Zhaoyun, S., Shah, B., Hussain, T., Shah, S.M., Ali, F., Khan, U.S.: A Novel Driver Emotion Recognition System Based on Deep Ensemble Classification. Complex & Intelligent Systems. Vol. 9, pp. 6927–6952 (2023). https://doi.org/10.1007/s40747-023-01100-9

[14] Zaman, F., Xu, Z., Hussain, A., Ullah, A., Zaman, K.: Enhancing Driver Emotion Recognition Through Deep Ensemble Classification. Journal of Intelligent and Connected Vehicles. Vol. 8(2), Article 9210055 (2025). https://doi.org/10.26599/JICV.2025.9210055

[15] Alfaras, M.S., Karan, O.: A Review of Advancements in Driver Emotion Detection: Deep Learning Approaches and Dataset Analysis. In: Proceedings of the 4th International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET), Fez, Morocco. pp. 1–9 (2024). https://doi.org/10.1109/IRASET60544.2024.10549432