A Cost-Effective Intelligent Edge-Based Driver Monitoring System for Real-Time Detection of Unsafe Driving Behaviors

Fyaz Nafin Rahman, Joaquin Owono Ntoo, Mostafa Mahmoud Selim, and Ahmed Abdelmoamen Ahmed*
Department of Computer Science, Prairie View A&M University, Prairie View, TX, USA
*Corresponding Author: amahmed@pvamu.edu

***Abstract*—The rapid increase in vehicle numbers across U.S. roadways has intensified concerns regarding traffic safety, as motor vehicle incidents account for the overwhelming majority of transportation-related fatalities and injuries. Human factors– particularly distraction, fatigue, and drowsiness–remain the predominant causes, contributing to approximately 94% of all traffic accidents. Edge computing offers a promising technological avenue to mitigate these challenges by enabling data processing directly on local devices. This decentralized approach reduces latency and operational costs, preserves data privacy, and minimizes dependence on external network infrastructures. This paper proposes a cost-effective driver monitoring system, implemented using affordable edge hardware such as the Raspberry Pi, and designed for seamless integration into a wide range of vehicles. The system performs real-time visual analysis to detect unsafe driving behaviors and generate timely driver alerts. Model development and validation were conducted using a dataset of self-collected photos of real-world driving conditions across diverse environmental and behavioral contexts. Experimental evaluation under real-world driving conditions demonstrated a detection accuracy of approximately 92%, with a mean Average Precision (mAP@0.5) of 0.86 and mAP@0.5-0.95 of 0.68.**

***Index Terms*—IoT; AI; Wireless Communication; Edge Computing; Behavior Monitoring; Safe Driving; Cost-Effective.**

I. Introduction

Vehicle crashes remain the primary cause of transportation-related fatalities and injuries, accounting for over 95% of deaths and nearly all reported injuries in 2022 [1]. Human factors– particularly distraction, fatigue, and drowsiness– are major contributors, with drowsiness alone linked to 16-21% of fatal accidents [2]. Although existing driver-assistance systems have improved safety, their ability to detect behavior-related risks remains limited. As approximately 94% of crashes are attributed to human error, there is a pressing need for more advanced, technology-driven solutions to enhance road safety.

Integrating the Internet of Things (IoT) [3], Artificial Intelligence (AI) [4], and edge computing [5] offers a transformative framework for enhancing road safety through real-time, on-device data processing. By reducing reliance on cloud connectivity, edge computing minimizes latency, bandwidth, and privacy limitations [6]. Imagine a fully edge-based driver monitoring system, implemented on low-cost hardware such as a Raspberry Pi device, coupled with a standard camera module, offers a practical and scalable alternative to traditional cloud-dependent architectures. This design eliminates the need for costly infrastructure and continuous internet connectivity, broadening accessibility across a broad spectrum of users– from individual drivers to commercial fleet operators [7].

This study introduces a real-time driver monitoring system leveraging edge computing to detect unsafe driving behaviors, including distraction, drowsiness, and seatbelt non-compliance. By performing local on-device computation, the framework preserves data privacy, ensures real-time responsiveness, and aligns with edge computing principles for decentralized intelligence [8], offering a cost-effective solution for improving driver safety in varied operational contexts.

A dataset of 1,133 images was collected to train and evaluate the proposed system, capturing diverse drivers, illumination conditions, and camera perspectives to enhance generalization. Additional real-time imagery from live driving sessions improved posture, motion, and background variations robustness. The dataset comprised seven classes (see Figure 1) with manual annotations to ensure accurate labeling for reliable ML-based detection of safe and unsafe driving behaviors.

The developed system was evaluated under diverse real-world driving conditions, achieving approximately 92% accuracy in detecting unsafe behaviors such as drowsiness and distraction. At a confidence threshold of 0.5, the model attained a mean Average Precision (mAP@0.5) of 0.766, demonstrating reliable object localization and behavioral classification on resource-constrained hardware. These results validate the effectiveness of the edge-based driver monitoring system, with further improvements expected through dataset expansion and iterative refinement to capture reckless behaviors, establishing a scalable framework for real-time driver safety interventions.

II. Related Work

Recent innovations in intelligent transportation systems demonstrate the growing impact of distributed and edge computing on road safety. IoT-enabled traffic signals dynamically adjust timing to reduce congestion [9], while AI-driven edge dashcams and localized IoT traffic platforms enhance driver awareness and adaptive traffic management. These technologies underscore the importance of low-latency, edge-based computation in building safer transportation networks.

Lightweight neural network architectures [10] are widely employed to ensure efficient execution on resource-limited

(a) Wearing a seatbelt, not looking forward, and eyes closed. (b) Wearing a seatbelt, not looking forward, and using a phone. (c) Wearing a seatbelt, using a phone, and holding an object. (d) Not wearing a seatbelt, closed eyes, and holding an object. (e) Not wearing a seatbelt and not looking forward. (f) Wearing a seatbelt and closed eyes.

Fig. 1. Representative samples of unsafe driving behaviors from the dataset. The images illustrate varying degrees of non-compliance with safe driving practices, ranging from partial compliance–such as drivers wearing seatbelts while simultaneously using mobile phones–to complete disregard for safety measures, where no preventive actions (e.g., seatbelt use or visual attention to the road) are observed.

devices. CNN models such as MobileNet, ResNet, and VGGNet [11], along with SVM-based methods utilizing Eye and Mouth Aspect Ratios [12], represent common strategies for driver state detection. Enhanced robustness has been achieved through Kalman filters, Haar Cascades, and LSTM networks for temporal modeling. Additionally, edge-optimized frameworks like TensorFlow Lite enable deployment on embedded platforms such as Raspberry Pi and Jetson Nano [13], [14], [15], [16], [17], [18], [9]. Collectively, these developments highlight the synergy between algorithmic efficiency and hardware-aware optimization for real-time driver monitoring.

Oviedo et al. [14] implemented a real-time driver monitoring system on a Raspberry Pi 5 using MediaPipe for detecting drowsiness indicators such as prolonged eye closure and yawning, following a similar approach to Akrout et al. [15]. A Kalman filter was employed to reduce false positives, and the system additionally monitored seat belt usage. Achieving approximately 92% accuracy for drowsiness detection and over 95% for seat belt compliance, the fully edge-based solution demonstrated efficient real-time performance and low-cost hardware integration, underscoring the viability of embedded AI for practical driver safety applications.

Several studies have demonstrated the feasibility of deploying advanced AI models on low-power hardware such as the Raspberry Pi [5]. Vadlamudi and Ahmadinia [19] integrated InceptionV3 and LSTM networks with Haar Cascade preprocessing to achieve real-time drowsiness detection with approximately 95% accuracy using TensorFlow Lite. Similarly, Moredo et al. [17] employed an LSTM-based framework leveraging EAR, MAR, and head pose features, attaining 95.23% accuracy and real-time performance on Raspberry Pi hardware. These works illustrate the effectiveness of optimized deep learning pipelines for edge-based driver monitoring.

III. System Design and Implementation

A. System Architecture

The system architecture, depicted in Figure 2, comprises four primary stages: data collection and preprocessing, edge computing, AI models, and a user interface. These components operate sequentially to capture driver activity, preprocess and

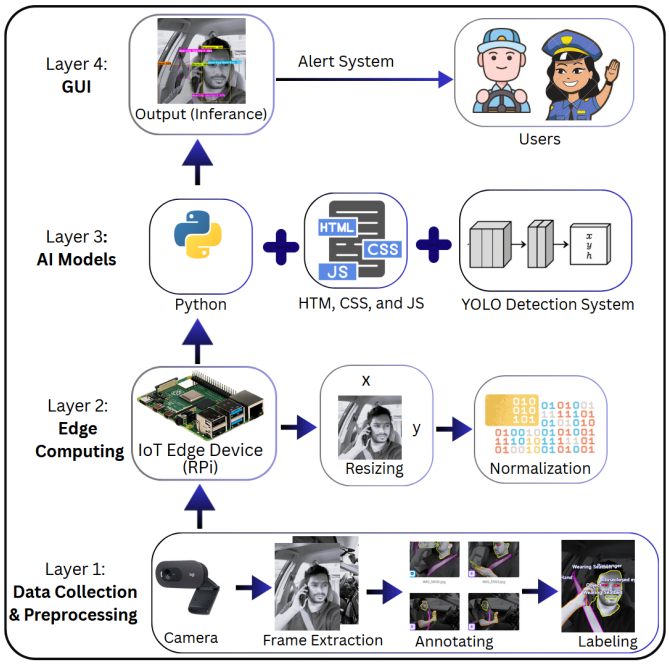


Fig. 2. System Architecture

structure the data for analysis, identify safety-critical behaviors, and deliver real-time feedback to the driver.

The system is initiated using a Raspberry Pi 5 interfaced with a camera module positioned to capture the driver's frontal view. The camera continuously acquires video streams, which are decomposed into individual frames. This conversion from continuous video to discrete images facilitates frame-by-frame analysis, ensuring that each driver activity is captured and evaluated for precise and timely behavioral assessment.

Upon capture, each frame undergoes a series of preprocessing operations on the Raspberry Pi to render it suitable for inference. These operations include resizing the image to conform to the input dimensions required by the YOLOv8n [20] model and normalizing pixel values to the [0, 1] range, as formalized in Equation 1. Such standardization ensures consistency with the model's training data, which is essen-

tial for preserving detection accuracy across varying lighting conditions and camera perspectives.

$$I_{\text{norm}} = \frac{I_{\text{pixel}} - \mu}{\sigma} \tag{1}$$

where I_{pixel} represents the original intensity value of the image pixel, μ is the mean pixel value of the dataset, σ is the standard deviation of the pixel values, and I_{norm} is the resulting normalized pixel intensity used as input to the detection model.

The preprocessed frame is subsequently input to the YOLOv8n (nano) model, exported from Roboflow and trained on a custom-annotated dataset. YOLO (You Only Look Once) [21] is a real-time object detection framework that partitions an image into a grid, with each cell responsible for simultaneously predicting bounding boxes and associated class probabilities. This unified, single-pass methodology enables YOLO to achieve a balance of high processing speed and robust detection accuracy. The model identifies driver-related safety indicators within the proposed system, including eye status (open or closed), yawning, seatbelt usage, hand position, the presence of objects, and passenger detection. The model produces a bounding box, a corresponding class label, and a confidence score for each identified instance.

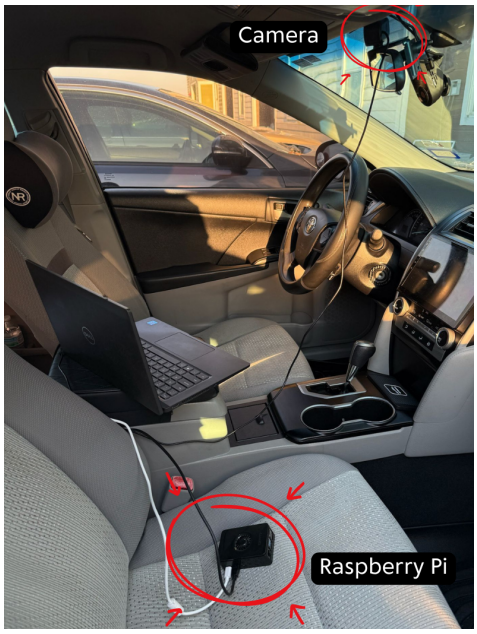
In the final stage, the system renders real-time detection outputs by overlaying bounding boxes and class labels onto the live camera feed. The system generates an immediate alert upon identifying an unsafe condition–such as prolonged eye closure, yawning, or the absence of seatbelt use. Depending on the driver's configuration, this alert may be conveyed visually on a display, audibly via a buzzer, or through synthesized speech. This real-time feedback mechanism facilitates prompt driver awareness and intervention, mitigating safety risks.

B. Dataset Collection

Figure 3(b) depicts the camera placement within the vehicle cabin, configured to capture real-time images of the driver. The camera may be positioned directly in front of the driver or at an oblique angle, providing an unobstructed view of the driver's face and upper body. The dataset consists of 1,133 images acquired by the authors, partitioned into 993 for training, 90 for validation, and 50 for testing. During annotation, we ensured balanced class representation, minimizing potential bias and enhancing the robustness of the training process.

The system was implemented on a Raspberry Pi 5 (4GB RAM, 2.4GHz quad-core Cortex-A76 processor), selected for its optimal balance between computational performance and energy efficiency, as illustrated in Figure 3(a). A camera was interfaced with the Raspberry Pi and mounted directly in front of the driver to continuously capture images. The device operated on Raspberry Pi OS (64-bit) with Python 3.9, utilizing OpenCV for image acquisition and preprocessing, and NumPy for numerical computations. Inference was performed using TensorFlow Lite. The object detection model, developed and exported from Roboflow, was trained on a custom-labeled dataset encompassing seven behavioral classes. To optimize performance on the resource-constrained hardware, the trained model was subsequently quantized and converted to TensorFlow Lite format, reducing its memory footprint and improving inference efficiency.

(a) Experimental Hardware Configuration



(b) System Setup in a Vehicle

Fig. 3. System setup and experimental configuration. The figure depicts: (a) the Raspberry Pi connected to its peripheral components and the configuration deployed on a laboratory bench during preliminary testing and calibration; and (b) the fully assembled system mounted within a vehicle to emulate real-world driving conditions.

C. Dataset Preprocessing and Labeling

The annotation and training workflow was performed using Roboflow 3.0, which manages the end-to-end computer vision pipeline. Each image $I_i \in \mathcal{D}$ in the dataset $\mathcal{D}$ was annotated with bounding boxes and labels corresponding to seven classes, $\mathcal{C} = \{\textit{open eye}, \textit{closed eye}, \textit{yawning}, \textit{seatbelt}, \textit{hand}, \textit{object}, \textit{passenger}\}$, as summarized in Table I and shown in Figure 1.

TABLE I
DISTRIBUTION OF OBJECT IMAGES ACROSS TRAINING, TESTING, AND VALIDATION DATASETS. THE DATASET IS PARTITIONED INTO 87.64% FOR TRAINING, 4.41% FOR TESTING, AND 7.95% FOR VALIDATION.

Class	Training	Testing	Validation	Total
Closed Eye	967	49	88	1104
Hand	1159	58	105	1322
Object	993	50	90	1133
Open Eye	1237	62	112	1412
Passenger	898	45	82	1025
Wearing Seatbelt	1152	58	105	1315
Yawning	463	23	42	528
Total	6869	345	624	7838

For each image, I_i, the annotation is defined as:

$$A_i = \{(b_{ij}, c_{ij}) \mid j = 1, 2, \ldots, n_i\}, \tag{2}$$

where $b_{ij} = (x_{ij}, y_{ij}, w_{ij}, h_{ij})$ denotes the bounding box coordinates and $c_{ij} \in \mathcal{C}$ represents the class label.

The YOLOv8n model optimizes the total detection loss as:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}}, \tag{3}$$

where λ_{box}, λ_{cls}, and λ_{obj} are balancing coefficients for the corresponding loss components. These are defined as:

$$\mathcal{L}_{\text{box}} = 1 - \text{IoU}(b_{\text{pred}}, b_{\text{gt}}), \quad \mathcal{L}_{\text{cls}} = -\sum_{k=1}^{|\mathcal{C}|} y_k \log(\hat{y}_k), \tag{4}$$

$$\mathcal{L}_{\text{obj}} = (p_{\text{obj}} - \hat{p}_{\text{obj}})^2. \tag{5}$$

Following model training, floating-point weights W_{float} were quantized to 8-bit integers for TensorFlow Lite deployment as:

$$W_{\text{int8}} = \text{round}\left(\frac{W_{\text{float}}}{s}\right) + z, \tag{6}$$

where s is the scale factor and z is the zero-point offset determined during calibration.

Quantization reduces the model's computational footprint, yielding faster inference. The inference time reduction can be approximated by:

$$T_{\text{TFLite}} \approx \frac{T_{\text{float}}}{r}, \tag{7}$$

where r is the quantization-induced speedup ratio on CPU inference.

Finally, the exported TensorFlow Lite model is represented as:

$$M_{\text{TFLite}} = f_{\text{YOLOv8n}}(I_i; W_{\text{int8}}), \tag{8}$$

which enables real-time detection of driver behaviors on the Raspberry Pi CPU.

IV. EXPERIMENTAL RESULTS

The developed system was systematically evaluated regarding inference accuracy, computational performance, and overall cost efficiency.

Fig. 4. Representative safe driving scenarios. Both drivers maintain appropriate posture, wear seatbelts correctly, and exhibit sustained attention to the roadway while keeping both hands on the steering wheel.

Fig. 5. Representative unsafe driving scenarios. Both drivers exhibit clear indicators of drowsiness, including closed eyes and yawning, reflecting impaired alertness and reduced driving vigilance.

A. Inference Accuracy

Multiple test cases were analyzed under diverse conditions to elucidate the system's inference accuracy, including variations in head pose, facial expression, and driver activity. These visualizations demonstrate the system's capacity to detect and classify driver behaviors in real time. Across all scenarios, the system consistently identified critical regions of interest– such as the eyes, mouth, and hands—by overlaying bounding boxes with corresponding class labels. Figure 4 presents an example of a safe driving scenario where the drivers wear seat belts and maintain forward gaze without engaging in any secondary activities. In these conditions, the system's detection confidence ranged from 70% to 97%.

Figure 5 depicts an example of unsafe driving behavior, wherein both drivers exhibit signs of drowsiness, including yawning and prolonged eye closure. Such behaviors substantially increase the risk of accidents by impairing driver attention. The system successfully identified these behaviors, achieving 73% and 96% detection confidence levels.

Table II summarizes the performance of the unsafe driving detection system across multiple classes, including hand position, object presence, passenger detection, seatbelt status, and eye state. Including a background class is essential, as it enables the model to differentiate relevant targets from non-informative regions. This distinction reduces false positives by correctly classifying areas devoid of target objects as background, thereby enhancing overall detection accuracy.

Figure 6 illustrates the relationship between model precision

TABLE II

CONFUSION MATRIX OF THE DRIVER MONITORING MODEL SHOWING CLASSIFICATION OUTCOMES ACROSS BEHAVIORAL AND CONTEXTUAL CATEGORIES. DIAGONAL ENTRIES REPRESENT CORRECT CLASSIFICATIONS. BKG IS THE BACKGROUND CLASS.

Actual	**Predicted Class**							
	Hand	Object	Passen-ger	Seatbelt	Open Eye	Yawning	Closed Eye	Bkg
Hand	**46**	0	0	0	0	0	0	12
Object	0	**26**	0	0	0	0	0	9
Passenger	0	0	**43**	0	0	0	0	15
Seatbelt	0	0	0	**28**	0	0	0	9
Open Eye	0	0	0	0	**26**	0	0	2
Yawning	0	0	0	0	0	**11**	1	4
Closed Eye	0	0	1	0	0	0	**18**	6
Bkg	5	3	6	7	3	0	3	**0**

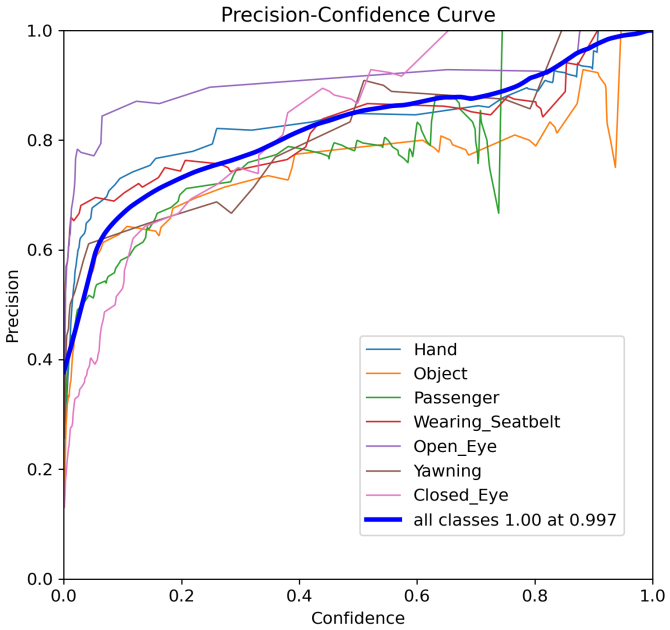


Fig. 6. Precision–Confidence curve illustrating the relationship between the ML model's confidence scores and its classification precision across all classes.

and confidence score, providing an overview of the model's reliability across varying levels of certainty. Precision measures the proportion of positive detections that are correctly identified. The curve demonstrates that overall precision increases with higher confidence thresholds, indicating that predictions made with greater confidence are more likely to be accurate. At a confidence level of 0.997, the model achieves a precision of 1.00, reflecting perfect accuracy for its most confident predictions—an outcome particularly valuable in applications where minimizing false positives is critical. Notably, the "Closed-Eye" class maintains exceptionally high precision across a wide range of confidence levels, underscoring the model's robustness for this behavior category.

A comprehensive class-wise evaluation of Precision, Recall, and F1-Score, in conjunction with the mean Average Precision

TABLE III

PER-CLASS PERFORMANCE METRICS OF THE DRIVER MONITORING MODEL. THE TABLE REPORTS PRECISION, RECALL, AND F1-SCORE FOR EACH BEHAVIORAL CATEGORY.

Class	**Precision**	**Recall**	**F1-Score**
Hand	0.83	0.88	0.86
Object	0.78	0.83	0.80
Passenger	0.78	0.78	0.78
Wearing Seatbelt	0.81	0.74	0.77
Open Eye	0.91	0.90	0.90
Yawning	0.80	0.91	0.85
Closed Eye	0.88	0.77	0.82
Macro Average	**0.83**	**0.83**	**0.82**

at an IoU threshold of 0.5 (mAP@50) and across the 0.5-0.95 range (mAP@50-95), highlights the model's classification strengths and limitations. The ML model performs strongly on visually distinct classes, such as Hand and Yawning, achieving high precision and recall, reflecting accurate and consistent detection. This indicates that the system effectively recognizes features with prominent spatial or motion characteristics. In contrast, performance is comparatively lower for more challenging classes, such as Open Eye, which display considerable intraclass variability due to variations in eye morphology, partial occlusions, and differing illumination conditions.

Table III presents the average values of three key performance metrics–Precision, Recall, and F1-Score–offering a comprehensive summary of the trained model's effectiveness. The model demonstrates powerful performance in detecting hands and critical facial features. The "Open Eye" class achieves the highest F1-Score of 0.90, indicating high accuracy and reliability. Similarly, "Hand" (0.86) and "Yawning" (0.85) exhibit excellent performance, reflecting consistent detection capabilities. Other classes, including "Closed Eye" (0.82), "Object" (0.80), and "Passenger" (0.78), also maintain robust F1-Scores, demonstrating competent performance across additional categories. For "Wearing Seatbelt," the model achieves a balanced performance with an F1-Score of 0.77, Precision of 0.81, and Recall of 0.74. Collectively, these results provide strong evidence that the model is well-suited for real-time driver monitoring, offering reliable and well-rounded performance across a range of critical behavioral classifications.

B. Computational Performance

We evaluated the average processing time, measured in milliseconds (ms), for each of the seven driver behavior classes. The "Open Eye" class exhibited the highest average processing time at 87.590 ms. This was followed by "Wearing Seatbelt" (71.964 ms), Object (72.012 ms), and Yawning (76.633 ms). The remaining classes demonstrated lower processing times: "Closed Eye" averaged 62.369 ms, Hand 56.673 ms, and Passenger 55.483 ms. This analysis provides a clear and informative assessment of the system's computational efficiency across different behavior classes, highlighting both performance and real-time applicability.

TABLE IV
ESTIMATED COST BREAKDOWN FOR THE EDGE-BASED UNSAFE DRIVING DETECTION SYSTEM.

Category	Component Description	Cost (USD)
Hardware	Starter Kit for Raspberry Pi 5, 64GB MicroSD, Fan, Cables, 18W USB-C Power Supply	104.99
	Pi Camera v2, Logitech C270 HD Webcam	15.94 – 21.99
Software	Raspberry Pi OS, Python, OpenCV, NumPy, TensorFlow Lite	Free
	Model Labeling (Roboflow free plan)	Free
	Model Training (Roboflow free plan)	Free
Total Estimated		**120.93 – 126.98**

C. Cost Analysis

The system was developed with a relatively low overall cost, as most software components and tools utilized are open-source and freely available, as summarized in Table IV-C.[1] The principal expenses are hardware-related, including the Raspberry Pi starter kit and a compatible camera for image acquisition. On the software side, core components such as Raspberry Pi OS, Python, OpenCV, NumPy, and TensorFlow Lite incur no additional cost. Moreover, model annotation and training were conducted using Roboflow's free plan, eliminating further expenses. Collectively, these factors render the system cost-effective and feasible for practical deployment in real-world driver monitoring applications.

V. CONCLUSIONS

This study introduces a real-time driver monitoring system that integrates IoT, AI, and edge computing on a Raspberry Pi to detect unsafe driving behaviors. Using a low-cost camera and optimized deep learning models, the system efficiently analyzes driver behavior while maintaining a compact, energy-efficient setup. Leveraging open-source tools such as Python, OpenCV, NumPy, TensorFlow Lite, and Roboflow, the total cost remains under $130, enabling scalable deployment. Experimental results demonstrate approximately 92% accuracy, with low-latency inference, enhanced privacy, and independence from continuous network connectivity, highlighting its practicality for on-road applications. The system has significant potential for improving road safety through further refinement. Future work includes robust operation under diverse lighting conditions, such as nighttime or tunnel driving, potentially via infrared or thermal imaging. The system could also achieve finer-grained classification of unsafe behaviors (e.g., mobile phone use, eating, incorrect seatbelt use) and incorporate additional sensor data, including vehicle speed, steering angle, and braking, to contextualize risk. Expanding the dataset and iterative model retraining would further improve adaptability to diverse drivers, vehicle types, and environmental conditions, ensuring sustained reliability and generalizability.

[1]Cost estimates are based on publicly available 2025 component prices and open-source software. All listed software tools are free to use under permissive licenses.

ACKNOWLEDGMENTS

This research work is supported by NSF under grants # 2011330, 2200377, and 2302469, is appreciated.

REFERENCES

[1] United States Department of Transportation, Bureau of Transportation Statistics, "Transportation statistics annual report 2024," https://doi.org/10.21949/e0kq-gf72, 2024.

[2] B. C. Tefft, "Drowsy driving in fatal crashes, united states, 2017–2021," AAA Foundation for Traffic Safety, Tech. Rep., 2024, https://aaafoundation.org/wp-content/uploads/2024/03/202304-AAAFTS-Drowsy-Driving-Countermeasures.pdf.

[3] A. A. Ahmed, "An actor-based formal model and runtime environment for resource-bounded iot services," *Algorithms*, vol. 15, no. 11, 2022.

[4] A. A. Ahmed and M. Echi, "Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras," *IEEE Access*, vol. 9, pp. 63 283–63 293, 2021.

[5] A. A. Ahmed and V. Bhadani, "Wanderwatch: Lorawan-based monitoring system for wandering risk patients," in *ICNC*, 2025, pp. 468–472.

[6] G. P. Sharma, "Real-time traffic management using iot sensors and edge computing in smart cities," *IJTRD*, vol. 11, 2024.

[7] A. Khan, K. S. Khattak, Z. H. Khan, T. A. Gulliver, and Abdullah, "Edge computing for effective and efficient traffic characterization," *Sensors*, vol. 23, no. 23, p. 9385, 2023.

[8] S. T. Ahmed, A. A. Ahmed, A. Annamalai, and M. F. Chouikha, "A scalable and energy-efficient lorawan-based geofencing system for remote monitoring of vulnerable communities," *IEEE Access*, vol. 12, pp. 48 540–48 554, 2024.

[9] J. R. Krishna, N. Anusha, G. Karthik, E. R. Krishna, and G. Roshith, "Convolutional neural network-based driver drowsiness detection system," *ARPN Journal of Engineering and Applied Sciences*, vol. 20, no. 9, pp. 540–547, 2025.

[10] A. A. Ahmed and S. Ahmed, "A real-time car towing management system using ml-powered automatic number plate recognition," *Algorithms*, vol. 14, no. 11, 2021.

[11] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, pp. 57–99, 2024.

[12] E. Quiles-Cucarella, J. Cano-Bernet, L. Santos-Fernández, and C. Roldán-Blay, "Multi-index driver drowsiness detection method based on driver's facial recognition using haar features and histograms of oriented gradients," *Sensors*, vol. 24, no. 17, p. 5683, 2024.

[13] A. A. Ahmed and B. Nyarko, "Smart-watcher: An ai-powered iot monitoring system for small-medium scale premises," in *ICNC*, 2024, pp. 139–143.

[14] A. Alvarez Oviedo, J. F. Mamani Villanueva *et al.*, "Design of a system for driver drowsiness detection and seat belt monitoring using raspberry pi 4 and arduino nano," *Designs*, vol. 9, no. 1, 2025.

[15] B. Akrout and S. Fakhfakh, "How to prevent drivers before their sleepiness using deep learning-based approach," *Electronics*, vol. 12, no. 4, 2023.

[16] A. A. Moamen and N. Jamali, "Opportunistic sharing of continuous mobile sensing data for energy and power conservation," *IEEE Transactions on Services Computing*, vol. 13, no. 3, pp. 503–514, 2020.

[17] M. J. R. Moredo, J. D. S. Celino, and J. B. G. Ibarra, "Multi-feature long short-term memory facial recognition for real-time automated drowsiness observation of automobile drivers with raspberry pi 4," *Engineering Proceedings*, vol. 92, no. 1, 2025.

[18] A. K. Pathak, A. K. Singh, P. Kumar, V. Bhatia, and O. Krejcar, "Real-time anti-sleep alert algorithm to prevent road accidents to ensure road safety," *Frontiers in Future Transportation*, vol. 6, 2025.

[19] S. Vadlamudi and A. Ahmadinia, "An embedded intelligence engine for driver drowsiness detection," *IET Computers & Digital Techniques*, vol. 16, no. 1, pp. 10–18, 2022.

[20] "Ultralytics – yolov8n model," https://docs.ultralytics.com/models/yolov8/, accessed: 2025-11-16.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640v5*, 2016.