IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

Real-Time Smartphone-Based Driver Drowsiness Detection Using Multi-Cue Facial Landmark Fusion and Graded Alerting

Lalitha P.C.[1], Rajendhran A.[2], Deepa T.[3] and Akila K.[*4]

Department of Computer Science and Engineering, SRM Institute of Science and Technology, Vadapalani Campus, Chennai[1-4]

Abstract: A drowsy driver reacts slowly, judges poorly, and often gives no outward sign of it. Most systems built to catch this rely on infrared cameras, wearable sensors, or heavy deep-learning hardware, so they rarely reach ordinary drivers. This paper describes a drowsiness detection and alert system that runs completely on an Android smartphone and needs nothing but its front camera. Every sampled video frame goes through MediaPipe face detection and landmark extraction. From the landmarks we measure four cues: eye closure and blinking (Eye Aspect Ratio, EAR), yawning (Mouth Aspect Ratio, MAR), and head-pose deviation. The cues are collected over a sliding window and merged into one weighted fatigue score between 0 and 100%. A threshold-based decision engine turns that score into one of four states: Normal, Warning, Drowsy, or Severe Drowsiness. Each state has its own response, which grows from an on-screen warning to audio and vibration alarms and, in severe cases, an optional emergency notification. Sessions, events and alerts are saved in a local SQLite database that feeds an offline dashboard, with optional cloud sync. The app is built with Flutter, OpenCV, MediaPipe and TensorFlow Lite. We explain the architecture, the scoring and decision rules, and an evaluation protocol based on accuracy, precision, recall, response time and false-alert rate, and we report eight functional test cases. The results indicate that a cheap, private, graded monitoring pipeline is practical on ordinary phones.

Keywords: driver drowsiness detection, eye aspect ratio, mouth aspect ratio, facial landmarks, MediaPipe, TensorFlow Lite, road safety, mobile application

INTRODUCTION

Road crashes kill about 1.19 million people a year, according to the World Health Organization [1]. Drowsiness is one of the most dangerous contributors because it wears down reaction time, judgement and awareness, and the driver usually gets little warning that it is happening. Long-haul and night-shift drivers face the greatest exposure, yet a commuter on a dull stretch of highway can also slip into a microsleep. These are involuntary lapses of a few seconds, which is long enough at highway speed to drift out of a lane or miss a hazard completely [2,3]. Alcohol can be tested at the roadside; drowsiness cannot, and it is probably under-recorded in crash investigations.

Regulators have started to act. Regulation (EU) 2019/2144 obliges new vehicles to include a drowsiness and attention warning function [4], and driver monitoring is increasingly expected in cars with partial automation [2]. But factory-fitted systems only protect new, usually more expensive vehicles. Older cars, taxis and commercial fleets are left out.
Methods for automatic drowsiness detection are usually grouped by what they observe. Indirect methods are vehicle-based and infer fatigue from steering or lane position. Direct methods watch the driver. Hybrid methods use both [5]. Vehicle signals arrive late and are muddied by road conditions and driving style. Physiological signals such as EEG, ECG or photoplethysmography (PPG) can be accurate, but the driver has to wear a sensor [6]. Cameras avoid that burden, and in-cabin infrared (IR) cameras hold up well across lighting conditions [2], although they need dedicated hardware.

A phone camera is already in the driver's pocket, and simple landmark-based measures are light enough to compute on the phone itself. The Eye Aspect Ratio reduces six landmarks around an eye to one number whose sustained drop signals closure [7]. PERCLOS summarises how much of a time window the eyes spend closed [8]. MediaPipe and its Face Mesh model supply dense facial landmarks at interactive speeds on mobile hardware [9,10]. What we found missing in the literature is a complete app that works offline, combines several of these cues, escalates in steps instead of sounding one fixed alarm, and keeps a local record for later review.

 This work is licensed under a Creative Commons Attribution 4.0 International License

This paper describes such a system. Its main contributions are:

- A detection pipeline that runs fully on the device and needs only the front camera of a standard Android phone, with no IR camera or wearable.
- A fatigue score that combines eye-closure and blink statistics (from EAR), yawning (from MAR) and head-pose deviation over a sliding window.
- A four-level decision engine with matching audio, vibration, visual and emergency alerts.
- An offline-first storage and dashboard layer with optional cloud synchronisation, plus an evaluation protocol and functional test results.

RELATED WORK

Driver-state monitoring has been studied for over twenty years, and several surveys sort the field by sensing modality [3,5]. Here we look at the nine studies that shaped our design most (Table 1).

Infrared landmark-based monitoring. Kim et al. [2] built a monitoring system around an in-house IR camera fixed behind the steering column. After the face is found, its landmarks go to two modules. One estimates head pose to flag inattention. The other applies an eye-closure filter to a thresholded eye crop to flag drowsiness. On their self-recorded IR dataset the authors report detection above 99%, running at 20–25 frames per second on a desktop-class computer and about 10 frames per second on an embedded board. The approach is fast and accurate, but its lighting robustness comes from the IR sensor. Using it with an ordinary visible-light camera would need a different way of judging eye state.

Fusion with physiological signals. Yu et al. [6] locate 68 facial landmarks to get 2-D facial and 3-D head-pose parameters, add five heart-rate-variability features computed from PPG, and feed the ten combined features to an LSTM network. They report 97.36% accuracy. The extra physiological channel helps, but it needs a PPG sensor on the driver, the very dependency we want to avoid.

MediaPipe and EAR/MAR pipelines. Several studies compute EAR, MAR and sometimes head pose from MediaPipe or Dlib landmarks. Albadawi et al. [11] combine EAR, MAR and head pose with conventional machine-learning classifiers on the NTHU driver drowsiness dataset. Safarov et al. [12] pair MediaPipe landmarks with a deep-learning classifier to find blinks and yawns in a custom dataset. A MediaPipe Face Mesh system [13] relies on EAR and MAR, and a Dlib-based one [14] on blink analysis. Taken together, these works show that landmark ratios are cheap and informative. They also admit trouble with lighting, occlusion and differences between faces [11–13], and blink-based rules can raise false alarms when a normal blink is read as closure [14].

Deep and embedded approaches. Deep networks learn richer features but cost more to run. The IoT-oriented system in [15] tests an LSTM and several convolutional backbones (VGG16, InceptionV3, DenseNet) and sends alerts over an IoT link, yet it demands substantial computing power. The embedded system in [16] runs a CNN with Dlib-based eye and mouth detection on low-cost hardware and is held back by that hardware's speed. The CNN–LSTM model in [17] combines PERCLOS, mouth closure and head pose over continuous video, again at a higher computational price.

Table 1. Summary of the reviewed studies.

Ref.	Methods	Dataset	Reported limitation
[2]	Facial landmarks, eye-closure filter, head pose	IR driver video (self-recorded)	Requires IR camera
[6]	68 landmarks, head pose, PPG-derived HRV features, LSTM	30 drivers	Requires additional sensor
[11]	MediaPipe Face Mesh, EAR, MAR, head pose, ML classifiers	NTHU-DDD [18]	Lighting, occlusion, facial variation
[12]	Deep learning, MediaPipe landmarks, EAR, blink and yawn detection	Custom dataset	Lighting, facial variation, limited diversity
[13]	MediaPipe, OpenCV, EAR, MAR	Video data	Lighting and occlusion
[14]	Dlib, OpenCV, eye-blink analysis	Video data	Blinking can cause false alerts

 This work is licensed under a Creative Commons Attribution 4.0 International License

Ref.	Methods	Dataset	Reported limitation
[15]	LSTM, VGG16, InceptionV3, DenseNet, IoT alerting	Custom driver dataset	High computational resources
[16]	CNN, Dlib, eye and mouth detection	YawDD [19]	Limited processing speed on embedded hardware
[17]	CNN, PERCLOS, mouth closure, head pose, LSTM	Driver fatigue video dataset	Continuous video and higher computation

What the survey shows. Three things stand out. EAR and MAR are the most common and cheapest indicators, used by several independent groups [6,11–13]. MediaPipe Face Mesh is a favoured landmark front end because it balances accuracy against cost on mobile hardware [11–13]. And accuracy usually costs something: extra hardware such as IR cameras [2] or wearables [6], or extra computation such as deep recurrent models [15,17]. The recurring failure modes are strong or poor lighting, sunglasses and masks, and false alarms from blinking or talking [11–14].

Research gap. As far as we can tell, few of the reviewed works put landmark ratios, head pose and temporal accumulation into one light application that runs wholly on a standard phone, avoids IR hardware and wearables, answers in graded multimodal steps, and keeps a local history. Their output is typically a yes/no decision or a single alarm. This work aims at that gap.

METHODS

System overview

The system turns a live video stream into alerts and stored history in six modules that run in sequence for every sampled frame: (1) camera input and preprocessing, (2) facial feature extraction, (3) drowsiness analysis and fatigue scoring, (4) a decision engine, (5) an alert manager, and (6) data storage, dashboard and reports (Figure 1). Modules 1 to 5 form the real-time loop and run entirely on the phone. Module 6 saves results in the background so that storage never slows detection. An optional FastAPI backend with MongoDB or Firebase can sync derived statistics between devices and deliver emergency notifications, but the core loop does not depend on a network.

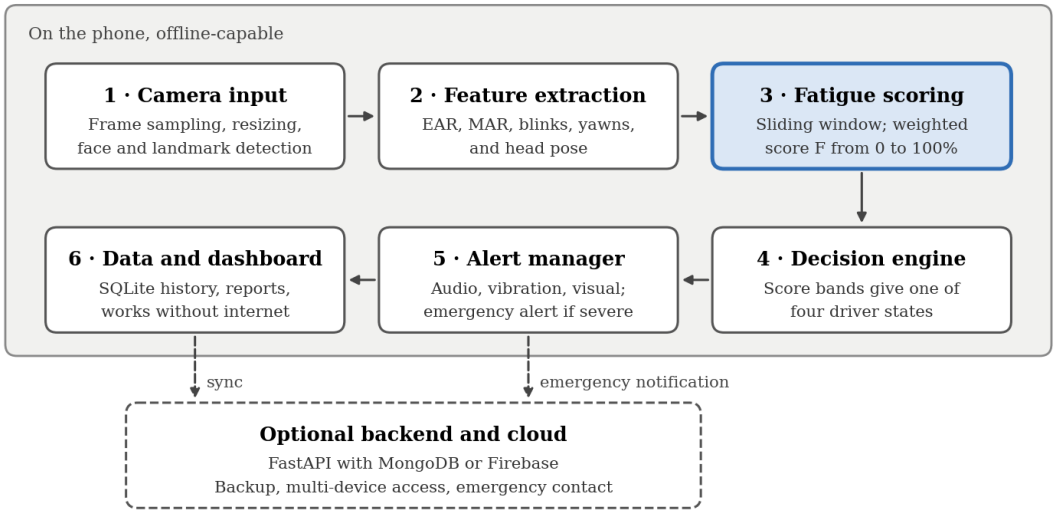


Figure 1. System architecture. Front-camera frames pass through preprocessing, landmark extraction, feature computation, fatigue scoring and classification to the alert manager and the local database.

Video acquisition and preprocessing

The phone sits on the dashboard or windscreen with the front camera facing the driver. Instead of analysing every frame, the system takes frames at regular intervals, a compromise between quick response and processor and battery load. Each sampled frame is resized and normalised to soften the effect of ambient light before analysis.

Face detection and landmark extraction use MediaPipe Face Mesh. It pairs a lightweight single-shot face detector [20] with a network that regresses 468 three-dimensional points on the face surface [9,10]. The points around the eyes, mouth and face contour go to Module 2 as coordinates. OpenCV handles supporting image operations such as resizing, colour conversion and normalisation [21].

Facial feature extraction

Eye aspect ratio and blinks. For each eye we pick six landmarks p_1 to p_6. Points p_1 and p_4 sit at the horizontal corners, p_2 and p_3 on the upper lid, and p_6 and p_5 on the lower lid. The EAR [7] is

$$EAR = (\|p_2 - p_6\| + \|p_3 - p_5\|) / (2\|p_1 - p_4\|) \quad (1)$$

and the values of both eyes are averaged. EAR stays roughly steady while the eye is open and falls toward zero as it closes. A closure is registered when EAR stays below a threshold τ_e for at least n_{min} consecutive frames. A closure shorter than T_{blink} counts as a blink, and a longer one as prolonged closure. From the stream of closures the module derives blink frequency and mean blink duration. The share of frames in a window of N frames with EAR below τ_e approximates PERCLOS [8]:

$$PERCLOS = (1/N) \Sigma_i I(EAR_i < \tau_e) \quad (2)$$

Mouth aspect ratio and yawning. Eight landmarks q_1 to q_8 around the mouth give

$$MAR = (\|q_2 - q_8\| + \|q_3 - q_7\| + \|q_4 - q_6\|) / (2\|q_1 - q_5\|) \quad (3)$$

where q_1 and q_5 are the mouth corners. A yawn is recorded when MAR stays above τ_m for at least T_{yawn}. Requiring a minimum duration keeps speech from counting, since the mouth only opens briefly while talking. The number and mean length of yawns in the window go forward to the scoring stage.

Head pose. The head's orientation relative to the camera is expressed as pitch, yaw and roll angles computed from landmark geometry. Perspective-n-Point methods [22] are a common route to such angles, as in [2]. Nodding (pitch), drooping or tilting (roll) and turning away from the road (yaw) beyond a limit Δ from the forward-facing reference are flagged as head-pose deviation, which may point to low alertness or a microsleep.

Fatigue score

The score is recalculated periodically over a sliding window of W seconds rather than on every frame, which smooths out momentary landmark noise. From the buffered values we compute four raw statistics: an eye-closure burden x_e (PERCLOS and the length of the longest closure), a blink statistic x_b (blink frequency and mean duration), a yawning burden x_y (count and duration of yawns), and a head-pose burden x_h (the fraction of the window with head-pose deviation). Each statistic is scaled to a sub-score between 0 and 1,

$$S_k = \min(1, x_k / x_k^{max}), k \in \{ e, b, y, h \} \quad (4)$$

and the sub-scores are combined as a weighted sum,

$$F = 100 \times (w_e S_e + w_b S_b + w_y S_y + w_h S_h), \text{ with } w_e + w_b + w_y + w_h = 1 \quad (5)$$

so that F runs from 0 to 100%. The weights are set empirically (Table 2). Because several independent cues feed the score, losing one of them, for example when a mask hides the mouth and disables MAR, does not stop detection altogether. The windowed approach follows the idea behind PERCLOS-based and recurrent temporal models [8,17] but is far lighter than a recurrent network.

Table 2. Parameters of the feature-extraction and scoring stages. Values are to be completed from the final configuration. EAR thresholds in the literature usually fall between about 0.2 and 0.3 and need tuning to the landmark model and camera.

Parameter	Symbol	Role	Value used
EAR threshold	τ_e	Below this value the eye counts as closed	[insert]
Minimum closure length	n_{min}	Consecutive frames needed to register a closure	[insert]
Blink/closure boundary	T_{blink}	Longest closure still counted as a blink	[insert]
MAR threshold	τ_m	Above this value the mouth counts as open for a yawn	[insert]

 This work is licensed under a Creative Commons Attribution 4.0 International License

Parameter	Symbol	Role	Value used
Minimum yawn duration	T_{yawn}	Continuous opening needed to register a yawn	[insert]
Head-pose limit	Δ	Angular deviation from the forward reference	[insert]
Window length	W	Sliding window for score computation	[insert]
Saturation values	x_k^{max}	Raw value at which a sub-score reaches 1	[insert]
Cue weights	w_e, w_b, w_y, w_h	Contribution of each cue to F	[insert]

Decision engine and alert manager

The decision engine compares F with fixed thresholds and assigns one of four states (Table 3). A move to a more severe state is accepted only if F stays inside the new band for a minimum time, so a long blink or a laugh does not set off an alarm. The alert manager then scales its response to the state. Nothing is sent in the Normal state, which avoids pointless interruptions during ordinary driving.

Table 3. Fatigue-score bands, driver states and system responses.

Fatigue score	State	Interpretation	System response
0–30%	Normal	Driver is alert	No alert
31–60%	Warning	Early signs of fatigue	Mild visual warning showing the fatigue level
61–80%	Drowsy	Clear drowsiness	Audio alarm, vibration and visual warning
81–100%	Severe drowsiness	High-risk condition	Strongest audio, vibration and visual alerts; optional emergency notification to a pre-set contact through the backend

Data management, dashboard and synchronisation

All persistent data live in a local SQLite database built around four entities (Table 4). Events and alerts are written in the background, so database delays never hold up detection. A login and registration flow stores the driver profile and basic vehicle details. The dashboard shows live status, the current fatigue score and start/stop controls, while history and report screens list past sessions and events. If the optional backend is switched on, chosen records are synchronised to MongoDB or Firebase for backup and use on several devices.

Table 4. Entities of the local database.

Entity	Key attributes
Driver Profile	driver_id, name, vehicle_details, created_at
Driving Session	session_id, driver_id (FK), start_time, end_time, duration
Drowsiness Event	event_id, session_id (FK), fatigue_score, event_type, timestamp
Alert Log	alert_id, event_id (FK), alert_type, severity, delivered_at

Implementation

The app is written in Flutter and Dart, which give one code base for several platforms and an asynchronous model that suits running the camera, inference and database access at the same time. OpenCV and MediaPipe handle landmark detection, TensorFlow Lite runs any learned or optimised inference parts on the device, and SQLite stores the data. The code is organised in feature folders that mirror the six modules (camera, vision, analysis, decision, alerts, data and UI), so each module can be built and tested separately. Table 5 lists the hardware and software requirements.

Table 5. Hardware and software requirements.

Component	Specification
Mobile device	Android smartphone with front camera
Processor and memory	Quad-core processor or better; 4 GB RAM or more
Storage	32 GB or more

 This work is licensed under a Creative Commons Attribution 4.0 International License

Component	Specification
Output devices	Speaker and vibration motor
Development PC	Intel Core i5 or AMD Ryzen 5; 8 GB RAM or more
Operating system	Android 8.0 or later; Windows 10/11 for development
Framework and language	Flutter and Dart
IDE	Android Studio or Visual Studio Code
Vision and ML	OpenCV, MediaPipe, TensorFlow Lite
Local database	SQLite
Optional services	FastAPI backend; MongoDB or Firebase

Several design choices answer the failure modes seen in the literature. Frame sampling limits battery and heat. Normalisation in preprocessing lessens sensitivity to lighting. Smoothing over time and minimum-duration rules on blinks and yawns avoid alerts from ordinary blinking or conversation. Detection, scoring and alerting all work offline, and cloud sync is an optional background task. To limit distraction, the in-session screen shows only a small status indicator, and detailed dashboards sit in a separate flow meant for when the vehicle is stationary.

Camera frames are processed on the phone, and only derived statistics such as scores and events are stored locally or optionally synchronised. No video has to leave the device for the system to work, which keeps the design privacy-conscious.

RESULTS

Evaluation protocol

Performance is judged on labelled recordings in which each analysis window is marked as alert or drowsy. For scoring, the Drowsy and Severe Drowsiness states count as positive predictions and Normal and Warning as negative, since Warning is only advisory. This choice should be stated next to any reported figures. With TP, TN, FP and FN for true positives, true negatives, false positives and false negatives, the standard measures are

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (6)$$

$$\text{Precision} = TP / (TP + FP) \quad (7)$$

$$\text{Recall} = TP / (TP + FN) \quad (8)$$

$$F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) \quad (9)$$

Three more measures suit a warning system in particular. Response time is the delay between the start of a labelled drowsy episode and the moment the alert is issued. The false-alert rate is the number of alerts that match no labelled drowsy episode, per hour of monitoring. Alert effectiveness is the share of alerts after which the fatigue score drops below the Warning band within a set interval, which makes the question of whether the cues win back the driver's attention measurable.

[Describe the evaluation data here: number of participants, sessions and total duration; recording conditions (indoor daylight, artificial light, simulated in-vehicle lighting); camera position; and how drowsy and alert segments were labelled. Public benchmarks such as NTHU-DDD [18] and YawDD [19] can be used to check the EAR and MAR logic externally.]

Functional testing

Testing ran at unit, integration, functional, performance and usability levels. Unit tests exercised each module on its own with known inputs. Integration tests confirmed that landmarks, features, scores and states pass correctly from stage to stage. Functional tests ran the whole pipeline end to end, performance tests looked at frame-processing delay, and a usability review concentrated on keeping driver distraction low. Tests were run on representative Android phones that met the minimum specification in Table 5, in indoor daylight, indoor artificial light and simulated in-vehicle lighting, with participants seated at typical driver-to-dashboard distances. All eight functional test cases in Table 6 passed.

 This work is licensed under a Creative Commons Attribution 4.0 International License

Table 6. Functional test cases and outcomes.

Test ID	Test description	Expected result	Status
TC-01	Face detection under normal lighting	Face and landmarks correctly detected	Pass
TC-02	EAR calculation on open and closed eyes	EAR value reflects eye state	Pass
TC-03	Blink detection versus prolonged closure	Brief blinks not classified as drowsiness	Pass
TC-04	MAR calculation during yawning	Yawn detected and logged	Pass
TC-05	Fatigue-score threshold classification	Correct state assigned (Normal, Warning, Drowsy, Severe)	Pass
TC-06	Alert escalation across states	Alert intensity matches classified severity	Pass
TC-07	Local data persistence across app restart	Session and event history retained	Pass
TC-08	Operation without internet connectivity	Core detection and alerts work offline	Pass

Detection performance

Table 7 gives the design target for each measure and leaves space for the observed values.

Table 7. Performance targets and observed values.

Metric	Target	Observed
Detection accuracy	> 90%	[insert]
Precision	> 85%	[insert]
Recall	> 85%	[insert]
Average response time	< 1 s	[insert]
False-alert rate	< 10%	[insert]
Alert effectiveness	[define target]	[insert]

[Insert the measured values, a confusion matrix, and charts of the response-time distribution and the number of false alerts per hour.]

These measures only make sense together. Precision and recall show how well real drowsiness is told apart from normal driving: high precision spares the driver needless alarms, and high recall means real episodes are not missed. Response time limits the value of a correct detection, because an accurate alert that comes late leaves no time to react. The false-alert rate decides whether drivers keep trusting the system, since one that sounds off for ordinary blinks or conversation may be muted or uninstalled. Alert effectiveness shows the result that finally matters, whether the driver becomes alert again.

Demonstration

The whole pipeline, from camera input and landmark detection through EAR/MAR feature extraction, fatigue scoring, classification and graded alerting to local storage and dashboard reports, was demonstrated end to end on an Android smartphone (Figure 2).

Figure 2. Output of each module on the smartphone: (a) live camera feed with landmark overlay; (b) EAR, MAR and head-pose readings; (c) fatigue-score and state indicator; (d) alert screens for the Warning, Drowsy and Severe states; (e) dashboard and history screens. [Insert screenshots.]

DISCUSSION

Compared with existing approaches. Table 8 contrasts the main sensing strategies. Next to IR-camera systems [2] and wearable systems [6], our approach gives up some low-light robustness in return for much lower cost and easy deployment: nothing has to be bought, installed or worn beyond the phone the driver already owns. Next to deep recurrent models [15,17], it replaces learned temporal modelling with an explicit windowed score. That needs no large labelled training set and is light enough for a phone CPU, though it may be less accurate in the lab and cannot adjust to an individual driver on its own. Recurrent networks such as LSTM [23] remain a sensible direction for later versions.

The multi-cue design aims to reduce the false-alarm weakness of single-cue rules [14]. A long blink lifts the eye-closure sub-score only briefly, whereas sustained closure together with yawning and head nodding pushes the score into the upper bands.

Table 8. Qualitative comparison of sensing strategies, drawn from the literature reviewed above.

Approach	Extra hardware needed	Burden on the driver	Principal drawback
Vehicle-based (steering, lane position)	Access to vehicle data	None	Indirect and late; confounded by road and driving style
Wearable physiological (EEG, ECG, PPG)	Worn sensor	High	Inconvenient to wear and maintain [6]
In-cabin IR camera	IR camera and control unit	None	Cost and hardware dependence [2]
Deep-learning vision (CNN, LSTM)	Often GPU or embedded accelerator	None	High computational demand [15,17]
Proposed smartphone system	None beyond the phone	None	Sensitive to extreme lighting and occlusion

Limitations. Accuracy may drop in very dim light or strong glare, because the system uses a visible-light camera and not an IR sensor. Sunglasses that hide the eyes weaken EAR, masks that cover the mouth weaken MAR, and the phone's position and angle affect landmark quality. The thresholds are fixed and not personalised, which can disadvantage drivers whose eye shape or blink pattern is unusual. The evaluation so far covers a limited number of participants and scenarios, and wider trials on real roads are still needed. The system is a proof of concept, not a certified automotive safety product.

Safety and human factors. The app is a supplementary aid and not a replacement for proper rest, and in-app messaging should say so to discourage over-reliance. The phone must be mounted where it does not block the driver's view. Alert tones need to be loud enough to work without startling the driver, and repeated alerts can be tuned out over time, which is one reason the alert intensity is graded. Emergency notifications to a contact should be opt-in and rest on the driver's consent.

CONCLUSION

We built a real-time drowsiness detection and alert system that needs only the front camera of an Android phone. Eye closure and blinks (EAR), yawning (MAR) and head-pose deviation are taken from MediaPipe facial landmarks, merged over a sliding window into one fatigue score, and sorted by a threshold-based engine into four states that trigger graded audio, vibration, visual and optional emergency alerts. Local storage with an offline dashboard keeps a history of sessions and events, and optional synchronisation allows backup and use on several devices. Functional tests showed that the modules behave as designed, including offline operation. Because it relies on hardware people already carry and keeps camera frames on the device, the approach offers a low-cost, privacy-conscious way to help reduce drowsiness-related crashes. [Add project-specific conclusions drawn from the measured results.]
Several directions follow from this work:

- Integration with vehicle telematics and in-car infotainment for automatic session start and stop.
- Automatic notification of predefined emergency contacts or fleet systems, with location, when severe drowsiness is detected.
- Better robustness to night driving, sunglasses, masks and head coverings, through extra sensing or improved on-device models.
- Personalised thresholds learned from each driver's baseline over several sessions.
- Growth of the optional cloud layer into a fleet-management analytics dashboard.
- Field trials with a larger and more varied set of drivers and vehicles.

REFERENCES

[1] Kaplan, S., Guvensan, M. A., Yavuz, A. G. & Karalurt, Y. Driver behavior analysis for safe driving: A survey. IEEE Trans. Intell. Transp. Syst. 16, 3017–3032 (2015).

[2] Dong, Y., Hu, Z., Uchimura, K. & Murayama, N. Driver inattention monitoring system for intelligent vehicles: A review. IEEE Trans. Intell. Transp. Syst. 12, 596–614 (2010).

[3] Mohanty, S., Hegde, S. V., Prasad, S., & Manikandan, J. (2019, November). Design of real-time drowsiness detection system using dlib. In 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE) (pp. 1-4). IEEE.

[4] Rana, M. R., & Baul, S. (2024). An Overview of Operating Systems Based on Microkernel Technology and their Essential Components. International Journal of Information Engineering and Electronic Business, 16(6), 18-28.

[5] Huang, H.; Zhou, Y.S.; Zhang, F.; Liu, F.C. An optimized eye locating and tracking system for driver fatigue monitoring. In Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China, 2–4 November 2007; Volume 3, pp. 1144–1149.

[6] Kavitha, K.R.; Lakshmi, S.V.; Reddy, P.B.K.; Reddy, N.S.; Chandrasekhar, P. Driver Drowsiness Detection Using Face Expression Recognition. Ann. Rom. Soc. Cell Biol. 2021, 25, 2785–2789.

[7] Chand, H.V.; Karthikeyan, J. Cnn based driver drowsiness detection system using emotion analysis. Intell. Autom. Soft Comput. 2022, 31, 717–728.

[8] Macrae, C. Learning from the failure of autonomous and intelligent systems: Accidents, safety, and sociotechnical sources of risk. Risk Anal. 42, 1999–2025 (2022).

[9] Shaily, S., Krishnan, S., Natarajan, S. & Periyasamy, S. Smart driver monitoring system. Multimed. Tools Appl. 80, 25633–25648 (2021).

[10] Kaur, R.; Singh, K. Drowsiness detection based on EEG signal analysis using EMD and trained neural network. Int. J. Sci. Res. 2013, 10, 157–161.

[11] Khushaba, R.N.; Kodagoda, S.; Lal, S.; Dissanayake, G. Driver drowsiness classification using fuzzy wavelet-packet-based feature-extraction algorithm. IEEE Trans. Biomed. Eng. 2010, 58, 121–131.

[12] McDonald, A.D.; Schwarz, C.; Lee, J.D.; Brown, T.L. Real-time detection of drowsiness related lane departures using steering wheel angle. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting; Sage Publications: Los Angeles, CA, USA, 2012; Volume 56, pp. 2201–2205.

[13] Triyanti, V.; Iridiastadi, H. Challenges in detecting drowsiness based on driver's behavior. IOP Conf. Ser. Mater. Sci. Eng. 2017, 277, 012042.

[14] Choi, Y., Han, S. I., Kong, S.-H. & Ko, H. Driver status monitoring systems for smart vehicles using physiological sensors: A safety enhancement system from automobile manufacturers. IEEE Signal Process. Mag. 33, 22–34 (2016).

[15] Shahverdy, M., Fathy, M., Berangi, R. & Sabokrou, M. Driver behavior detection and classification using deep convolutional neural networks. Expert Syst. Appl. 149, 113240 (2020).

[16] Marchisio, A. et al. Sevuc: A study on the security vulnerabilities of capsule networks against adversarial attacks. Microprocess. Microsyst. 96, 104738 (2023).

[17] Tayab Khan, M.; Anwar, H.; Ullah, F.; Ur Rehman, A.; Ullah, R.; Iqbal, A.; Lee, B.H.; Kwak, K.S. Smart real-time video surveillance platform for drowsiness detection based on eyelid closure. Wirel. Commun. Mob. Comput. 2019, 2019, 2036818.

[18] Sri Mounika, T.; Phanindra, P.; Sai Charan, N.; Kranthi Kumar Reddy, Y.; Govindu, S. Driver Drowsiness Detection Using Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and Driver Distraction Using Head Pose Estimation. In ICT Systems and Sustainability; Springer: Berlin/Heidelberg, Germany, 2022; pp. 619–627.

[19] Al Redhaei, A.; Albadawi, Y.; Mohamed, S.; Alnoman, A. Realtime Driver Drowsiness Detection Using Machine Learning. In Proceedings of the 2022 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 21–24 February 2022; pp. 1–6.

[20] Saradadevi, M.; Bajaj, P. Driver fatigue detection using mouth and yawning analysis. Int. J. Comput. Sci. Netw. Secur. 2008, 8, 183–188.