

DrowsGuard: A Hardware-Aware Edge AI and Hybrid Cloud Ecosystem for Real-Time Driver Drowsiness Detection

Sion Sebastian Valentin
University of Craiova
Faculty of Automation,
Computers and Electronics
Craiova, Romania
sion.sebastian03@gmail.com

Paul Ștefan Popescu
University of Craiova
Faculty of Automation,
Computers and Electronics
Craiova, Romania
stefan.popescu@edu.ucv.ro

Marian Cristian Mihăescu
University of Craiova
Faculty of Automation,
Computers and Electronics
Craiova, Romania
cristian.mihaescu@edu.ucv.ro

ABSTRACT

This paper addresses road safety by introducing DrowsGuard, a software ecosystem that detects driver fatigue using commercial smartphones. It employs a distributed client-cloud-client architecture: a native Android application (Kotlin, Jetpack Compose) and a desktop administrative dashboard (Electron, React). To ensure robust real-time evaluation under device constraints, the analytical core implements a user-selectable dual-model architecture. The primary engine runs at the Edge using CameraX and Google ML Kit, assessing fatigue through the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR), with adaptive calibration for glasses wearers and a dynamic MAR baseline. A custom Convolutional Neural Network (CNN), optimized via Transfer Learning, is offered as an alternative engine. Although the CNN reached high validation accuracy, in-cabin tests revealed severe domain shift from lighting and vibration, making the geometric model substantially superior in practice (94.5% accuracy, 12 ms per frame). Data management uses an offline-first strategy (Room) synchronized with a NoSQL database (Firestore). The system reaches 97.0% sensitivity (Recall) at 94.5% overall accuracy on the real-world test subset, and an expert heuristic inspection indicates good coverage of the usability dimensions that matter in an in-vehicle context. The evaluation is a single-vehicle feasibility study rather than a fleet trial, but it shows that a smartphone-only pipeline can meet the latency and reliability requirements of in-cabin fatigue monitoring without prohibitive hardware costs.

AUTHOR KEYWORDS

Drowsiness Detection; Edge AI; EAR; MAR; Transfer Learning; Domain Shift; Usability Heuristics; Firestore; Offline-First; Jetpack Compose.

DOI: 10.37789/icusi.2026.10

INTRODUCTION

Driver fatigue is a major risk factor on public roads, contributing significantly to severe traffic accidents that cause extensive material damage and loss of human life. Decreased reaction capacity, reduced vigilance, and episodes of somnolence — commonly known as micro-sleeps — are especially dangerous at highway speeds, where a single second of inattention translates into tens of meters travelled without control. Surveys of the field repeatedly identify fatigue among the leading human-factor causes of road crashes [17, 28], which has made proactive monitoring of the driver's state a priority within the automotive industry.

However, advanced fatigue-detection solutions are often integrated exclusively into premium, next-generation vehicles. These factory-installed systems operate as opaque “black-box” mechanisms, leveraging proprietary hardware and closed-source algorithms. This commercial paradigm leaves a massive segment of the existing global vehicle fleet without access to active safety systems. The primary motivation of this research is the democratization of road safety through an accessible prevention system, DrowsGuard, that capitalizes on ubiquitous hardware: the smartphones the vast majority of drivers already own and mount on their dashboards for navigation.

The engineering challenge is to embed complex Computer Vision algorithms into a resource-constrained mobile environment while guaranteeing strict real-time processing and respecting user privacy. To meet these demands, DrowsGuard adopts a distributed client-cloud-client architecture. The Android application acts as an autonomous Edge AI system, processing visual data locally to avoid latency and privacy breaches, while a desktop monitoring component for dispatchers and fleet managers is mediated by a serverless Firebase infrastructure for real-time telemetry.

The contributions of this work are fourfold:

- a hardware-aware, user-selectable dual-model engine that combines a robust geometric EAR/MAR analyzer with an optional CNN, adapting to device and lighting conditions;
- adaptive biometric calibration — a glasses-aware EAR threshold, a per-session MAR baseline, and temporal (sustained-closure) validation — that markedly reduces false alarms;
- an offline-first, privacy-by-design data layer with sub-second NoSQL synchronization that keeps critical alerts reliable under poor connectivity;
- a complete client-cloud-client ecosystem, including an administrative dashboard, evaluated both on detection accuracy and on usability through Nielsen's heuristics.

The remainder of the paper reviews the state of the art, details the system architecture and the analytical methodology, and reports the experimental and usability evaluation before concluding with future directions.

STATE OF THE ART

The concern for road safety and the reduction of human-factor accidents has driven the development of several generations of Advanced Driver Assistance Systems (ADAS). The evolution of these technologies can be categorized into three major stages according to the nature of the collected data, the level of intrusiveness, and the

computational paradigm employed. Comprehensive surveys confirm this taxonomy and highlight the gradual shift from vehicle-centric sensing toward non-intrusive, vision-based monitoring [17, 23, 28].

Mechanical and Vehicle-Dynamics Approaches

Early approaches did not monitor the driver directly; instead, they analyzed vehicle dynamics on the road. These early systems relied on sensors placed on the steering column to measure steering-wheel angle and rotation speed, or pressure on the acceleration pedal, on the premise that somnolence alters the probability of smooth vehicle control and leads to erratic micro-corrections. Subsequently, lane-departure warning systems were introduced, evaluating the standard deviation of the vehicle's lane position. The main disadvantage of these mechanical methods was decision latency: the system alerted the driver only after the driving behaviour had already severely degraded, which in many critical scenarios proved too late to avert a collision [22, 28].

Intrusive Physiological Monitoring

To obtain an earlier and more precise evaluation of wakefulness, researchers pivoted toward monitoring human vital functions. Laboratory systems measured brain activity through Electroencephalograms (EEG), heart-rate variability (HRV) through Electrocardiograms (ECG), and muscular or ocular activity (EMG, EOG) through electrodes attached directly to the driver's body. While offering high medical precision and the ability to detect the onset of sleep stages before physical manifestations occur, the direct physical contact of the sensors causes significant discomfort. This intrinsic intrusiveness renders such methods impractical for daily commercial use, relegating them mostly to clinical studies and simulated driving environments [22, 23].

Vision-Based Geometric Methods

The third and current stage is marked by systems based on artificial vision. This non-intrusive approach analyzes the driver's facial features and behaviour: blink frequency, degree of eye closure, yawning patterns, and head inclination. A seminal contribution in this category is the Eye Aspect Ratio (EAR) proposed by Soukupová and Čech, which derives a single scalar from the geometry of facial landmarks to quantify eye openness in real time [26]. The analogous Mouth Aspect Ratio (MAR) extends the same principle to detect yawning. Such geometric descriptors are computationally lightweight and well suited to embedded execution, but classical pipelines built on Haar cascades remain highly susceptible to failure under dynamic ambient lighting and varying camera angles [17, 21].

Deep Learning, TinyML and Edge AI

The evolution of Deep Learning made it possible to transition complex Convolutional Neural Networks (CNNs) directly onto mobile processors. Lightweight architectures such as MobileNetV2 [24], combined with Transfer Learning and optimizers like Adam [16], allow accurate facial-state classification within a constrained compute budget. By exporting models to optimized formats such as TensorFlow Lite, and by adopting emerging TinyML techniques [10], these algorithms can run at the Edge with very low inference latency and reduced memory consumption, without relying on external server processing. This shift also ensures compliance with modern data-privacy standards, since sensitive biometric video feeds never leave the local device.

Recent deep-learning drowsiness detectors report strong accuracy on curated datasets [21], yet their robustness in real driving conditions remains an open issue. IoT-oriented surveys further note the difficulty of fusing on-device inference with reliable cloud telemetry under intermittent connectivity [25].

Positioning of DrowsGuard

Most existing smartphone or IoT systems commit to a single detection paradigm and seldom address the full software lifecycle — calibration, offline persistence, synchronization, and fleet-level administration. DrowsGuard differs in three respects. First, it offers a user-selectable dual-model engine that pairs a robust geometric EAR/MAR analyzer with an optional CNN, letting the driver adapt to hardware and lighting conditions. Second, it embeds an offline-first persistence layer [2] with asynchronous NoSQL synchronization [14] so that critical alerts survive poor connectivity. Third, it couples the mobile client with a complete administrative dashboard, extending a detection prototype into an end-to-end client-cloud-client ecosystem.

SYSTEM ARCHITECTURE

To ensure robust, scalable operation capable of processing real-time data without compromising the user experience, the DrowsGuard ecosystem was designed using a micro-layered architecture. This approach cleanly separates the heavy local processing logic (Edge AI) from persistent cloud storage and remote administrative management. Figure 1 shows the three interconnected tiers and the data flow between them.

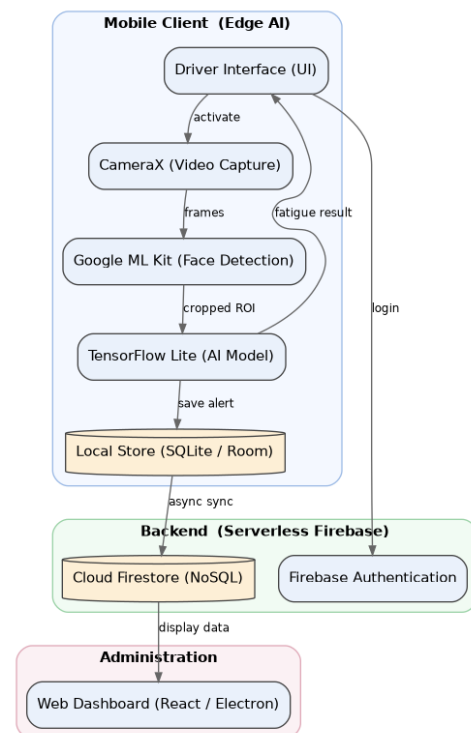


Figure 1. Layered architecture of the DrowsGuard ecosystem: an on-device Edge AI tier, a serverless Firebase backend, and a web administration tier.

System Modelling and Use Cases

The functional model of DrowsGuard is organized around three actors. The Driver operates the mobile client — starting and stopping biometric scanning, viewing the current vigilance state, managing the account, and configuring alerts

(a use case extended with the choice of stimulus and with sensitivity adjustments such as Glasses Mode). The on-device AI System is modelled as an autonomous actor whose “Detect Drowsiness” use case includes eye-movement (EAR) and facial-expression (MAR) analysis and, critically, a “Process Data Locally” case that enforces Privacy by Design. The Administrator interacts only through the web dashboard — managing users (CRM), handling support tickets, and analyzing anonymized, aggregated metadata to refine the detection algorithm. This separation keeps the safety-critical loop entirely on the device while exposing only administration and analytics to operators.

Mobile Application (Client Tier)

The DrowsGuard mobile application was developed natively for Android, targeting devices running API level 24 and above. The implementation uses the Kotlin programming language and the Jetpack Compose framework [5] to build a declarative, reactive user interface. The overall structure follows the Model-View-ViewModel (MVVM) design pattern, strictly decoupling the UI lifecycle from the underlying business logic and data-persistence layers [4].

The critical component of the client tier is the video operational flow. Frames captured by the device's optical sensor through the CameraX API [3] are routed asynchronously to the inference engines. The logic enforces a strict Privacy-by-Design principle: video frames are processed exclusively in the volatile memory of the terminal and are immediately discarded after the biometric metrics are extracted. To minimize the driver's cognitive load, the interface employs a dark theme, high-contrast typography, and large interactive elements suitable for an in-vehicle environment (Figure 2).

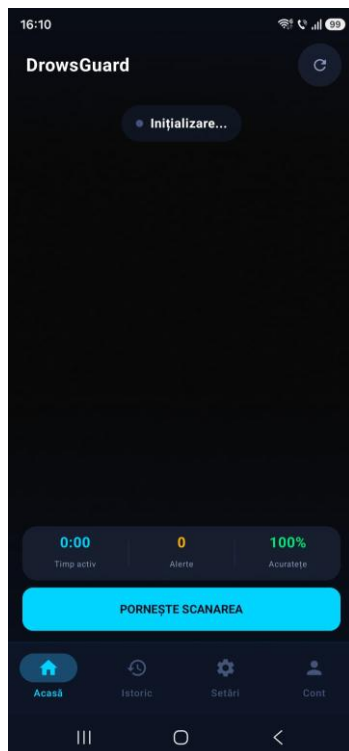


Figure 2. Main interface of the DrowsGuard mobile application running directly on the driver's terminal.

NoSQL Database and Offline-First Strategy

Traditional Relational Database Management Systems (RDBMS) impose rigid, highly normalized structures that often become performance bottlenecks when rapid, asynchronous writing of thousands of telemetry events is required in mobile environments. The DrowsGuard ecosystem adopts a document-oriented NoSQL model through the Firebase Firestore platform [14]. Data is stored hierarchically in JSON-like collections and documents, allowing massive horizontal scaling and dynamic schema evolution.

To ensure uninterrupted operation in areas with poor cellular coverage (e.g., mountainous regions, tunnels), the system employs a robust offline-first paradigm based on the native Android Room library [2, 6]. Biometric events and alerts are first written to a local SQLite database. A background synchronization worker continuously monitors network availability; once the connection is stable, the local records are asynchronously pushed to Cloud Firestore. A reactive Publish-Subscribe mechanism (onSnapshot listeners) instantly notifies subscribed interfaces, guaranteeing sub-second propagation of critical alerts without polling the server.

Administrative Panel (Desktop Tier)

The administrative component, designed for fleet management and technical support, was developed as a hybrid desktop application. The technology stack uses the React library [20] to create a modular, component-based user interface, and the Electron platform [9] for native desktop encapsulation across operating systems.

The panel acts strictly as a data consumer and does not communicate directly with the mobile clients over local networks. It displays real-time dashboards, aggregated statistical analysis through Chart.js [8] (including heatmaps and hourly alert distributions), and a CRM technical-support module based on Kanban ticketing [1]. Administrative actions, such as suspending a user account or resolving a bug report, interact natively with the Firebase Authentication API and Firestore [12], propagating state changes back to the mobile clients instantly.

Operational Analytics

Beyond live monitoring, the dashboard includes a business-intelligence module that turns anonymized, aggregated events into actionable insight. It reports the distribution of alert types and severities, a response-time histogram (the interval between an alarm firing and the driver dismissing it), an hour-by-weekday heatmap that exposes the most vulnerable driving windows, and an at-risk ranking of the drivers with the highest incidence of critical events. The Kanban support module ingests in-app problem reports and automatically attaches device metadata (model, OS and application version) to each ticket, accelerating diagnosis. Together, these views give fleet operators both a real-time and a longitudinal picture of driver vigilance, and supply the development team with evidence for tuning detection thresholds.

Privacy and Data Protection

Because the system processes biometric video, privacy was a primary design constraint rather than an afterthought. DrowsGuard follows a Privacy-by-Design model [7]: camera

frames and facial landmarks are processed exclusively in the device's volatile memory and discarded immediately after the EAR and MAR indices are computed. Only lightweight alert metadata — a timestamp and an event type — is transmitted to the cloud, so no biometric imagery ever leaves the terminal, a guarantee surfaced to the user through an explicit “100% local biometric data” banner and aligned with the data-minimization principle of the GDPR [11]. On the administration side, the web panel exposes no public sign-up; administrator accounts are provisioned manually and access is gated by privileged authentication tokens, isolating user metadata and management tools from unauthorized access.

ANALYTICAL METHODOLOGY

To maximize accuracy, prevent resource exhaustion, and minimize false-positive rates, the analytical core does not rely on a single, rigid algorithm. Instead, it implements a dynamic, user-selectable dual-model architecture. Through the mobile application's settings, the user can toggle between two distinct inference engines depending on the environmental conditions and the device's hardware capabilities.

Geometric Model (Landmark Detection)

The primary inference engine uses facial-landmark regression to map the user's physiognomy. Through the Google ML Kit framework [13], the system identifies a dense set of spatial coordinates on the face, focusing exclusively on the ocular and oral regions. From these coordinates, the analytical class computes two standardized biometric indices in real time: the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR) [26].

For the EAR index, the system triggers a micro-sleep pre-alert when the value drops below a critical threshold of 0.08 and stays there for a sustained period (> 2.5 seconds). A dedicated “Glasses Mode” raises that threshold to 0.12 for drivers wearing prescription lenses or sunglasses. The origin of these values is detailed in the next subsection.

For the MAR index, which indicates yawning and generalized fatigue, a static threshold is inefficient due to large physiognomic variation among users. Consequently, the model employs an adaptive calibration mechanism: during the first 30 frames of a driving session, the system computes a personalized resting baseline. A yawn alert is then triggered only when the current MAR value exceeds this baseline by a multiplication factor of 1.55.

Threshold Selection and Glasses Mode

The decision parameters — the EAR threshold of 0.08, the sustained-closure window of 2.5 s, the Glasses Mode threshold of 0.12 and the MAR multiplication factor of 1.55 — were fixed by manual tuning on the development recordings rather than by an exhaustive search, and each encodes an explicit trade-off between sensitivity and false-alarm tolerance. The EAR threshold is deliberately conservative: it lies well below the aperture observed during ordinary blinks, so that only a near-complete lid closure can arm the alarm, and it is never used alone, since a micro-sleep is defined by how long the eyelids remain closed rather than by an instantaneous value [21, 28]. The MAR factor is applied to a per-session baseline computed over the first 30 frames instead of to an absolute value, because mouth geometry varies far more across individuals than eye geometry; a relative criterion therefore transfers between

drivers without per-user configuration. Because these values were tuned rather than searched, their sensitivity is not quantified here: a systematic sweep of the operating point, reporting the false-alarm and miss rates obtained at each threshold, is part of the validation study outlined in the limitations.

Glasses Mode is a manual toggle exposed in the settings screen (Figure 3) rather than an automatic detector. Frames, tinted lenses and specular reflections reduce the contrast between the iris and the eyelid, so the landmark regressor reports a smaller vertical eye opening than the anatomical one; with the default threshold this systematic bias produces spurious micro-sleep pre-alerts for drivers who wear glasses. Raising the threshold to 0.12 compensates for the underestimation, while the sustained-closure requirement keeps ordinary blinks below the alarm condition. The mode alters only the EAR threshold: the MAR baseline, the temporal validation and the alerting logic are unchanged. The present evaluation aggregates sessions recorded with and without eyewear and therefore does not report separate accuracy figures for the two configurations; quantifying this ablation on a dedicated eyewear subset is left to future work.

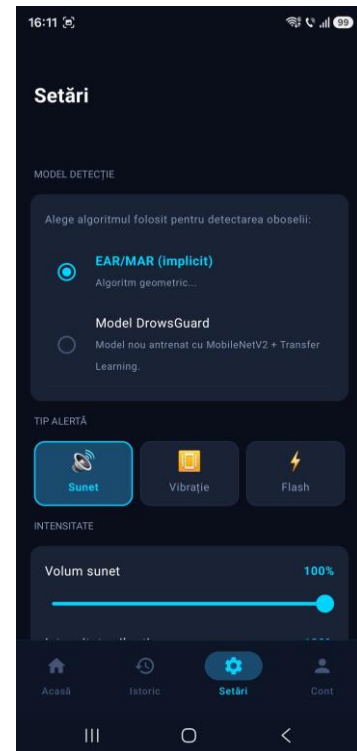


Figure 3. DrowsGuard settings menu: the user can toggle between the geometric EAR/MAR algorithm and the CNN-based “DrowsGuard Model”, alongside customizable alert-notification settings.

Convolutional Neural Network (CNN)

The second, alternative engine is an end-to-end Convolutional Neural Network based on the MobileNetV2 architecture [24]. This model is trained to extract complex features directly from the pixel matrix of isolated Regions of Interest (ROIs), specifically the cropped ocular zones.

A crucial architectural optimization was the integration of data normalization directly into the network graph. By appending a native Rescaling($1./127.5$, $\text{offset}=-1$) layer as the input node, the raw pixel distribution is scaled to the $[-1, 1]$ interval during inference, eliminating redundant, CPU-

intensive bitmap preprocessing in the Android environment. The coefficients are not arbitrary: MobileNetV2 was pre-trained on inputs normalized to the $[-1, 1]$ interval, and the preprocessing function shipped with the architecture applies exactly this affine map [15, 24]. Embedding it in the graph guarantees that the transformation applied on the device is identical to the one used during training — a frequent and silent source of accuracy loss when preprocessing is reimplemented in the mobile client — in addition to removing per-frame bitmap arithmetic from the CPU.

A major technical challenge arose during model serialization to the TensorFlow Lite (.tflite) format [27]. During aggressive 8-bit (INT8) quantization — aimed at reducing the model footprint — a severe degradation of network weights was observed: the quantized model collapsed, constantly returning a saturated probability value (e.g., 0.984375) regardless of the input frame. This deficiency was remedied by adjusting the export parameters and keeping computation at floating-point (Float32) precision, which preserved the dynamic range of the probability scores at the cost of a slightly larger file size.

Alerting Subsystem

Detection is only useful if it reliably reaches the driver, so DrowsGuard provides a configurable multimodal alerting subsystem. When a sustained event crosses the EAR or MAR thresholds, a dedicated alarm manager can fire any combination of an audible alarm (via MediaPlayer), a haptic signal (via VibrationEffect), and a camera-flash strobe, while sliders control the alarm volume and vibration amplitude. A “Detection Level” setting (Low / Medium / High) adjusts the number of consecutive frames required to raise an alert, letting the driver trade sensitivity against false-alarm tolerance. All preferences persist locally and apply immediately to the active session, ensuring that the warning channel matches both the driver's preferences and the ambient noise and lighting of the cabin.

EXPERIMENTS AND EVALUATION

Validating the efficiency of an Edge AI system requires rigorous quantitative and qualitative assessment, spanning from initial data curation to hardware profiling and usability inspection in real-world scenarios.

Experimental Setup

The CNN was trained in Keras on a GPU-accelerated environment (Google Colab) using an 80/20 train/validation split. Evaluation combined two complementary protocols: an offline validation pass on the held-out test subset, and in-cabin field testing under realistic driving conditions — daylight glare, tunnels, and night driving, with and without prescription glasses or sunglasses. Deployment was profiled on a high-end handset (Samsung Galaxy S24 Ultra) and on entry-level Android devices (API level 24+), where adaptive frame skipping preserved real-time throughput.

Exploratory Data Analysis (EDA)

During the exploratory analysis phase, the use of public datasets from the Kaggle platform (e.g., the Drowsiness Dataset by D. Perumandla and the Rakibul Drowsiness Prediction Dataset) was evaluated. Although these sets were quantitatively balanced across classes, the EDA revealed critical limitations: a severe lack of lighting diversity, purely frontal perspectives unrepresentative of bottom-up dashboard

camera angles, and a purely static nature of the samples. The public sets also remained below 3,000 images, which is insufficient to train or fine-tune a deep network for the demographic and accessory variation found in real cabins. Importantly, fatigue is a temporal phenomenon: a micro-sleep is defined by how long the eyelids stay closed (typically > 2.5 s), so single still images cannot capture it, motivating a continuous, sequence-based pipeline.

Consequently, a proprietary, robust dataset was compiled and augmented. The custom dataset incorporated critical variables such as camera rotations (-15° to $+15^\circ$), dynamic low-light conditions, and the presence of visual accessories (sunglasses, prescription glasses), maintaining a perfect balance between the vigilance (Active) and fatigue (Fatigue) classes to prevent optimization bias.

Training and Transfer Learning

Weight optimization for the CNN was performed with the Binary Cross-Entropy loss function and the Adam optimizer [16]. To prevent overfitting, an Early Stopping mechanism monitoring the validation loss (val_loss) was applied, together with a ModelCheckpoint callback. A two-stage Transfer Learning strategy was used. In the initial phase (epochs 0–8), the lower feature-extraction layers were frozen and only the top classification layers were trained, reaching an accuracy plateau of approximately 85%. Subsequently, in the fine-tuning phase (epochs 8–18), deeper layers were unfrozen and the learning rate was substantially reduced. This allowed the model to learn subtle biometric patterns, pushing validation accuracy beyond 90% with stable convergence, as shown in Figure 4.

The training configuration follows standard transfer-learning practice for MobileNetV2 rather than the outcome of a tuned search: the Adam step size was left at its default value during the frozen phase and reduced substantially for fine-tuning, so that the pre-trained convolutional filters are not destroyed by large gradient updates; the boundary between the two phases was placed at the epoch where validation accuracy plateaued (Figure 4), and Early Stopping on val_loss together with the ModelCheckpoint callback selected the final weights. No grid or random search over learning rate, batch size or unfreezing depth was performed, so the sensitivity of the reported figures to these choices is not measured. The comparison in Table 1 should therefore be read as a comparison between the geometric pipeline and a reasonably trained CNN baseline, not as a claim about the best achievable CNN.

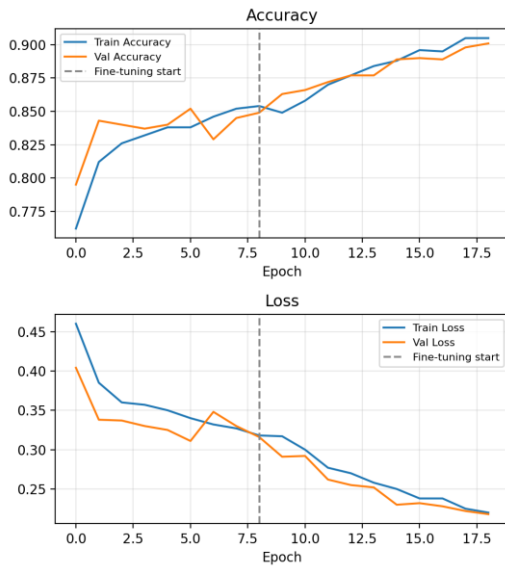


Figure 4. Training and validation accuracy and loss across epochs; the dashed line marks the start of fine-tuning.

Comparative Evaluation and Domain Shift

Although counter-intuitive relative to the metrics obtained in the isolated training environment, empirical in-cabin tests demonstrated a clear superiority of the geometric ML Kit model over the custom CNN. The CNN suffered heavily from domain shift: sudden lighting changes (e.g., passing under streetlights or through tunnels), sun glare, and high-frequency vibrations degraded its real-world classification performance to roughly 82%. Conversely, the ML Kit geometric model exhibited exceptional robustness, maintaining a real-world accuracy of 94.5% on the test subset of Figure 5. Latency analysis confirmed the value of the user-selectable architecture: the geometric model executes full inference in just 12 ms per frame, whereas the CNN requires 45 ms per frame. Table 1 summarizes the comparison.

Criterion	Geometric (ML Kit)	CNN (MobileNetV2)
In-cabin accuracy	$\approx 94.5\%$	$\approx 82\%$
Latency / frame	12 ms	45 ms
Domain-shift robustness	High	Low
On-device resource cost	Low	Moderate-high

Table 1. Geometric (ML Kit) vs. custom CNN under real driving conditions.

System-Level Hardware Profiling

Evaluating the solution's sustainability involved continuous profiling of the mobile device's hardware resources during a simulated long-duration driving session. The results, which demonstrate the viability of Edge AI execution, are detailed in Table 2.

Parameter	Average	Impact & Justification
Frame Rate (FPS)	24 – 30	Optimal fluidity, via frame skipping on entry-level devices.
RAM Utilization	180 – 220 MB	Efficient footprint; prevents OS process termination.
CPU Utilization	15% – 25%	Moderate; async computation does not block the UI thread.

Battery Drain	12% – 15% /h	Sustainable for long routes; offset by car chargers.
Thermal Regime	35 – 38 °C	High stability; eliminates thermal-throttling risks.

Table 2. System-level hardware profiling results.

Confusion Matrix and Global Metrics

Final system validation on the real-world test subset — 1,000 labelled instances collected during the in-cabin sessions — produced the distribution shown in Figure 5: 485 True Positives (correctly detected fatigue), 460 True Negatives (correctly identified vigilance), 40 False Positives (false alarms), and 15 False Negatives (missed fatigue episodes).

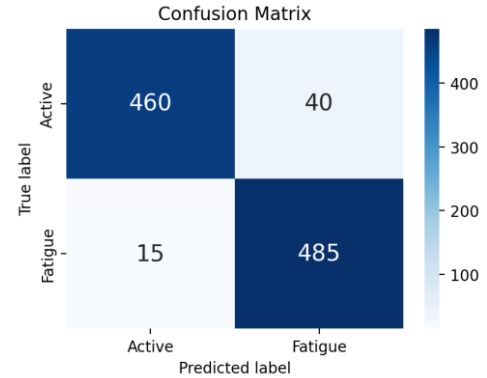


Figure 5. Confusion matrix on the real-world test subset (positive class = Fatigue).

From these absolute values, the following metrics were derived:

- Precision: $485 / (485 + 40) \approx 92.3\%$
- Recall (Sensitivity): $485 / (485 + 15) \approx 97.0\%$
- F1-Score: $\approx 94.6\%$
- Accuracy: $(485 + 460) / 1000 = 94.5\%$

From an engineering perspective, the architecture was intentionally calibrated to be marginally more sensitive. The critical Recall of 97.0% shows that the system inherently prefers an occasional false alarm over ignoring a potentially fatal micro-sleep episode, strictly prioritizing driver safety.

Usability Evaluation (Nielsen's Heuristics)

Because DrowsGuard operates in a safety-critical, attention-limited context, the mobile and desktop interfaces were inspected against Nielsen's ten usability heuristics [18]. Table 3 maps each principle to concrete design decisions in the system. The inspection indicates strong coverage of the dimensions most relevant to in-vehicle use — visibility of system status, error prevention, and minimalist design — while help and documentation emerges as the main direction for future refinement, for instance through a short guided onboarding flow.

The inspection was carried out by the authors, who walked through the mobile and desktop interfaces against the ten principles and then consolidated their observations into Table 3. This is a limitation of the study rather than a full heuristic evaluation: the protocol assumes several evaluators who are independent of the design team and who rate each violation on a severity scale [19], and neither condition is satisfied here. Table 3 should therefore be read as a structured design rationale that documents how each principle was addressed and exposes the most visible gap — help and documentation

— while a properly staffed heuristic evaluation and a task-based user study remain necessary to quantify usability.

Heuristic	Implementation in DrowsGuard
Visibility of system status	Live preview with overlaid landmarks and a dynamic “OK”/alert indicator; dashboard shows uptime and a LIVE event feed.
Match with the real world	Plain driving terminology and intuitive red/orange/green severity coding for alerts.
User control and freedom	Driver can start/stop scanning, switch detection model, and tune alert type, volume and sensitivity.
Consistency and standards	Shared dark theme and cyan accents across mobile and desktop; standard Material components.
Error prevention	Sustained closure (≥ 2.5 s) and an adaptive MAR baseline suppress false alarms; Glasses Mode prevents misdetection.
Recognition over recall	Large labelled controls grouped into clear zones; colour-coded history avoids memorisation.
Flexibility and efficiency	Selectable models for different hardware; quick filters and search speed up frequent tasks.
Aesthetic, minimalist design	Minimal, high-contrast dark interface designed to reduce driver cognitive load.
Recover from errors	“Report a problem” attaches device metadata to a support ticket for fast diagnosis.
Help and documentation	In-context labels explain each setting; a privacy banner clarifies local data handling.

Table 3. Heuristic evaluation of the DrowsGuard interfaces against Nielsen's ten usability heuristics.

LIMITATIONS AND THREATS TO VALIDITY

Several limitations should be acknowledged. The geometric pipeline depends on the quality and placement of the front camera; extreme glare, very low light, or severe occlusion can still reduce landmark accuracy, and the CNN alternative remains vulnerable to domain shift. The reported accuracy figures were obtained on a proprietary dataset and a single-vehicle in-cabin protocol, so broader on-road validation across drivers, vehicles, and cameras is required before strong generalization claims can be made. Finally, the usability assessment is an expert heuristic inspection rather than a controlled user study: it surfaces design strengths and the documentation gap but does not yet quantify task performance or subjective satisfaction. Four further threats should be made explicit. First, the custom dataset is described only qualitatively — the augmentation variables and the class balance are reported, but the number of subjects, sessions and recording devices is not — so the absolute figures should be read as evidence of feasibility rather than as a benchmark result, and they are not directly comparable with results published on public corpora. Second, the results are aggregated over the in-cabin sessions and are not broken down by lighting condition, phone placement, camera quality or eyewear, so the robustness claim is not resolved per condition. Third, both the detection thresholds and the training hyperparameters were hand-tuned, and their influence on the reported metrics was not measured. Fourth, the heuristic inspection was performed by the authors rather

than by independent evaluators. These threats directly motivate the larger evaluation outlined below.

Reproducibility is constrained by the nature of the material: the in-cabin recordings contain identifiable facial imagery and cannot be redistributed without the explicit consent of the recorded subjects. To make the study replicable within that constraint, the trained TensorFlow Lite model, the training and evaluation notebooks, and the derived non-identifiable EAR/MAR traces will be deposited in a public repository together with a dataset card describing the acquisition conditions.

CONCLUSION

This paper presented the design and implementation of DrowsGuard, a comprehensive hardware-aware ecosystem dedicated to road safety. The architecture supports the hypothesis of shifting Artificial Intelligence computation from the cloud to the user's mobile terminal (Edge Computing). In doing so, the system achieves the reaction times required for real-time operation (12 ms per frame) while keeping biometric data on the device.

The comparative analysis highlighted the superiority of geometric landmark regression over a standard CNN in overcoming in-cabin domain-shift phenomena, while the heuristic inspection indicates that the interface is well adapted to the demands of in-vehicle interaction. Supported by offline-first local persistence and NoSQL synchronization, the hybrid client-cloud-client approach provides a complete and functional platform whose feasibility has been demonstrated in in-cabin testing; deployment at fleet scale remains to be validated in a longitudinal study with multiple drivers, vehicles and handsets.

Future work aims to interconnect the Android application with the vehicle's CAN bus through OBD-II adapters, enabling the fusion of visual facial data with mechanical telemetry (steering-angle variance, acceleration micro-corrections) to build a multi-modal ADAS. Additionally, integrating wearable devices to monitor heart rate (BPM) and blood-oxygen saturation (SpO2) would provide physiological indicators deeply correlated with the onset of fatigue, while porting the vision pipeline to iOS (Core ML, Vision) would broaden accessibility. A larger user study with standardized usability metrics is also planned to complement the present heuristic inspection. That study is the immediate priority: a multi-driver, multi-vehicle protocol with a published dataset card, per-condition performance figures and independent usability evaluators would turn the present feasibility evidence into a generalizable result.

REFERENCES

1. Anderson, D.J. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, Sequim, WA, 2010.
2. Android Developers. Build an Offline-First App. 2026. <https://developer.android.com/topic/architecture/data-layer/offline-first>
3. Android Developers. CameraX Overview. 2026. <https://developer.android.com/training/camerax>
4. Android Developers. Guide to App Architecture. 2026. <https://developer.android.com/topic/architecture>
5. Android Developers. Jetpack Compose UI Toolkit. 2026. <https://developer.android.com/compose>

6. Android Developers. Save Data in a Local Database Using Room. 2026.
<https://developer.android.com/training/data-storage/room>
7. Cavoukian, A. Privacy by Design: The 7 Foundational Principles. Information and Privacy Commissioner of Ontario, Toronto, Canada, 2011.
8. Chart.js. Chart.js Documentation. 2026.
<https://www.chartjs.org/docs/latest/>
9. Electron. Electron Documentation: Build Cross-Platform Desktop Applications with JavaScript, HTML and CSS. 2026. <https://www.electronjs.org/docs/latest>
10. Essahraoui, S., Lamaakal, I., Maleh, Y., El Makkaoui, K., Filali Bouami, M., Ouahbi, I., Elmannai, H. and Abd El-Latif, A.A. FastKAN-DDD: A Novel Fast Kolmogorov-Arnold Network-Based Approach for Driver Drowsiness Detection Optimized for TinyML Deployment. PLOS ONE 20, 11 (2025), e0332577.
11. European Parliament and Council of the European Union. Regulation (EU) 2016/679 (General Data Protection Regulation). Official Journal of the European Union L119 (2016), 1–88.
12. Firebase. Cloud Firestore and Firebase Authentication Documentation. 2026. <https://firebase.google.com/docs>
13. Google. ML Kit Face Detection API. 2026.
<https://developers.google.com/ml-kit/vision/face-detection>
14. Kesavan, R., Gay, D., Thevessen, D., Shah, J. and Mohan, C. Firestore: The NoSQL Serverless Database for the Application Developer. In Proc. IEEE 39th International Conference on Data Engineering (ICDE) 2023, 3367–3379.
15. Keras. Keras Applications: MobileNet and MobileNetV2. 2026.
<https://keras.io/api/applications/mobilenet/>
16. Kingma, D.P. and Ba, J. Adam: A Method for Stochastic Optimization. In Proc. ICLR 2015.
17. Nasri, I., Karrouchi, M., Kassmi, K. and Messaoudi, A. A Review of Driver Drowsiness Detection Systems: Techniques, Advantages and Limitations. arXiv:2206.07489, 2022.
18. Nielsen, J. Enhancing the Explanatory Power of Usability Heuristics. In Proc. CHI 1994, 152–158.
19. Nielsen, J. Severity Ratings for Usability Problems. Nielsen Norman Group, 1994.
<https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/>
20. React. React: The Library for Web and Native User Interfaces. 2026. <https://react.dev/>
21. Safarov, F., Akhmedov, F., Abdusalomov, A.B., Nasimov, R. and Cho, Y.I. Real-Time Deep Learning-Based Drowsiness Detection: Leveraging Computer-Vision and Eye-Blink Analyses for Enhanced Road Safety. Sensors 23, 14 (2023), 6459.
22. Sahayadhas, A., Sundaraj, K. and Murugappan, M. Detecting Driver Drowsiness Based on Sensors: A Review. Sensors 12, 12 (2012), 16937–16953.
23. Saleem, A.A., Siddiqui, H.U.R., Raza, M.A., Rustam, F. et al. A Systematic Review of Physiological Signals Based Driver Drowsiness Detection Systems. Cognitive Neurodynamics 17, 5 (2023), 1229–1259.
24. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proc. CVPR 2018, 4510–4520.
25. Sharma, A., Verma, G., Mittal, M., Garg, M. and Rani, M. A Review of IoT-Based Driver Drowsiness Detection Systems: Current Approaches and Future Directions. International Journal of Scientific Development and Research 10, 3 (2025), 478–483.
26. Soukupová, T. and Čech, J. Real-Time Eye Blink Detection using Facial Landmarks. In Proc. 21st Computer Vision Winter Workshop (CVWW) 2016.
27. TensorFlow. TensorFlow Lite Guide – On-Device Machine Learning. 2026.
<https://www.tensorflow.org/lite/guide>
28. Xiao, Y. and Abas, A.B. A Review on Fatigue Driving Detection. ASP Transactions on Internet of Things 1, 3 (2021), 1–14.