Computers and Electrical Engineering 139 (2026) 111340

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

DRIVERDAPP: Driver's distraction record using deep learning and blockchain

Odinachi Udemezuo Nwankwo [a], Simeon Okechukwu Ajakwe [c], Muhammad Rasyid Redha Ansori [d], Gifar Arif Haryadi [a,b], Dong-Seong Kim [a,b,d], Jae Min Lee [a,*]

[a] Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi, South Korea
[b] ICT Convergence Research Center, Kumoh National Institute of Technology, Gumi, South Korea
[c] Smart Computing Department, Kyungdong University Global Campus, Goseong-gun, South Korea
[d] NSLab Inc., Gumi, South Korea

ARTICLE INFO

Dataset link: https://www.kaggle.com/c/state-farm-distracted-driver-detection, https://github.com/rasyidred/driver_dapp.git

Keywords:
Blockchain
Deep learning
Distracted driving
Intelligent system
Security

ABSTRACT

Existing driver distraction detection systems face critical barriers to real-world deployment in safety-critical transportation environments, including the lack of real-time edge inference, explainable artificial intelligence (XAI), trustworthy event logging, and privacy-preserving evidence management. To overcome these challenges, this paper presents an integrated framework, termed *DRIVERDAPP*, that unifies real-time edge-based detection, AI explainability, and secure, auditable event management. Red–green–blue (RGB) in-cabin image frames captured by a dashboard camera are processed locally on an NVIDIA Jetson Nano edge device, where a fine-tuned You Only Look Once version 11 small (YOLOv11s) model classifies ten driver behavior states and triggers in-vehicle audio alerts for unsafe activities. To suppress transient misclassifications under edge constraints, distraction persistence is verified using a lightweight temporal confirmation strategy. Confirmed distraction events are immutably recorded via Solidity-based smart contracts and submitted through the Web3.py interface to a permissioned Hyperledger Besu consortium blockchain operating under Quorum Byzantine Fault Tolerance (QBFT) consensus. Privacy is preserved by retaining raw visual data off-chain, while only pseudo-anonymous identifiers and event metadata are stored on-chain under controlled access policies. Model interpretability is enabled using Gradient-weighted Class Activation Mapping (Grad-CAM), providing transparent visual explanations of distraction-related predictions. The framework is evaluated using the State Farm Distracted Driver and American University in Cairo datasets, demonstrating stable real-time edge operation, negligible blockchain query latency, and secure smart contract execution. These results confirm the suitability of *DRIVERDAPP* for secure, explainable, and deployable driver monitoring in intelligent transportation systems.

1. Introduction

Driver distraction has emerged as a leading cause of road traffic accidents [1], particularly in the context of increasingly complex in-vehicle environments and pervasive mobile device usage. Visual, manual, and cognitive distractions significantly impair a driver's

* Corresponding author.
E-mail addresses: odinachifoot@gmail.com (O.U. Nwankwo), simeonajlove@gmail.com (S.O. Ajakwe), rasyidred@nslab.tech (M.R.R. Ansori), arpegio@kumoh.ac.kr (G.A. Haryadi), dskim@kumoh.ac.kr (D.-S. Kim), ljmpaul@kumoh.ac.kr (J.M. Lee).

https://doi.org/10.1016/j.compeleceng.2026.111340
Received 1 August 2025; Received in revised form 22 April 2026; Accepted 13 June 2026
Available online 29 June 2026

situational awareness and reaction time, thereby elevating accident risk in safety-critical transportation systems [2]. As a result, automated driver distraction detection has become an essential component of modern intelligent transportation and advanced driver assistance systems. Efforts like hands-free devices and education campaigns aim to address these behaviors and promote safer driving practices, emphasizing the critical importance of staying attentive on the road to prevent accidents and ensure overall road safety [3,4]. Road accidents represent a significant global problem, particularly in middle and lower-middle-class countries, with human errors causing around 90% of accidents. The World Health Organization reports about 1.35 million annual road accident deaths worldwide, with 64 people dying daily [5]. Developing intelligent cognitive mechanisms is critical for optimal safety and security of transportation and logistics [6,7]. The main objective of a driver distraction detection system is to monitor the driver's behavior and physiological signals to identify signs of inattention. These systems can alert the driver or trigger other safety measures to prevent accidents [8–13].

Recent advances in artificial intelligence–based vision models have enabled accurate recognition of driver behaviors from in-cabin images and videos. However, despite promising classification performance reported in prior studies, several core challenges continue to limit real-world deployment. First, many existing approaches rely on cloud-based processing, introducing latency, connectivity dependence, and privacy risks that are incompatible with real-time safety applications. Second, distraction detection systems often suffer from transient false alarms caused by short-term driver movements, reducing system reliability and user trust. Third, most studies treat detection as an isolated perception task, overlooking the need for trustworthy, tamper-resistant event logging that is critical for post-incident analysis, accountability, and regulatory compliance. Finally, the lack of artificial intelligence model explainability remains a major barrier to adoption in safety-critical domains, where transparent decision-making is required for validation, auditing, and liability assessment.

To address these challenges, this article presents an integrated, end-to-end framework that jointly considers real-time edge inference, explainable artificial intelligence, secure event management, and privacy preservation. The proposed system performs driver distraction detection directly on edge hardware, enabling low-latency operation without continuous cloud dependence. A lightweight temporal verification mechanism is incorporated to suppress short-lived misclassifications and improve operational robustness. Model decision behavior is further enhanced through visual explainability, enabling intuitive interpretation of distraction-related predictions. In parallel, detected distraction events are immutably recorded using blockchain technology, ensuring tamper-evident logging while maintaining user privacy by storing only pseudo-anonymous metadata on-chain and retaining raw visual data off-chain.

The primary contributions of this work are summarized as follows. First, a deployable edge-based driver distraction detection pipeline is designed to operate in real time under resource-constrained conditions. Second, a temporal confirmation strategy is introduced to improve reliability by reducing false alarms in continuous monitoring scenarios. Third, explainable artificial intelligence techniques are integrated to provide transparent visual justification of model decisions. Fourth, a permissioned blockchain-based event logging mechanism is incorporated to ensure trustworthy, auditable, and privacy-preserving management of distraction evidence. Finally, the system is evaluated as a unified framework, demonstrating its suitability for secure and accountable deployment in distributed intelligent transportation environments.

By explicitly addressing the practical limitations of existing driver distraction detection systems, this work advances the state of the art beyond isolated perception models and moves toward a holistic, trustworthy, and deployable solution for real-world transportation safety applications. Hence, the novelty of this work lies not in proposing a new detection model, but in the principled system-level integration of real-time edge intelligence, explainable AI, and privacy-preserving blockchain accountability under safety-critical constraints

1.1. Existing system

Several research efforts have explored driver distraction detection using deep learning models, computer vision techniques, and sensor-based systems. However, these approaches exhibit critical limitations in real-time applicability, scalability, and secure data management. Table 1 summarizes existing methodologies, highlighting their strengths and shortcomings.

A significant portion of existing research focuses on vision-based detection using convolutional neural networks (CNNs) trained on driver behavior datasets. For instance, authors in [14] proposed Drive-Net, a CNN and random forest-based model that identified driver distractions with 95% accuracy but did not address real-time in their design considerations. Researchers in [15] employed VGG16 and VGG19, achieving over 95% accuracy, though they did not consider decentralized data storage and Android-based user interface. A study conducted in [16] utilized the State Farm Dataset and developed multiple CNN models, with the Improved Vanilla CNN attaining 97.66% accuracy, demonstrating the benefits of deeper architectures and data augmentation but facing privacy and information security challenges. A research conducted in [17] compared ResNet, Xception, and VGG16, where ResNet achieved the highest accuracy (93.6%), real-time design consideration and data management were not addressed. Authors in [18] explored MobileNetV2, which attained 98.12% accuracy, highlighting its efficiency for embedded systems but lacking decentralized data management. While these studies demonstrated CNN-based models' effectiveness in detecting driver distractions, gaps remained in real-time design, decentralized storage, absence of artificial intelligence (AI) explainability, scalability of drivers' distraction information, privacy and information security, decentralized access to information, and user-friendly mobile application interface fusion for enhanced robustness.

Despite their strong classification performance, vision-based models face key challenges:

1. Real-time deployment constraints on resource-limited edge devices, where computational latency and power consumption can hinder timely inference.

Table 1
Comparison of existing systems & Methods.

Models	Accuracy (%) ↑	Key strengths	Limitations
Drive-Net (CNN + Random Forest) [14]	95.0	High accuracy, fast inference	Not optimized for decentralized storage, no AI model explainability
VGG16 and VGG19 [15]	>95.0	Accuracy	Absence of model explainability
Improved Vanilla CNN [16]	97.66	Good accuracy on Kaggle dataset	No blockchain integration, centralized storage, no AI model explainability
Resnet [17]	93.60	Good accuracy	Lack information storage robustness, no AI model explainability
MobilenetV2 [18]	98.12	Strong image classification	Lack of distraction classification storage, no AI model explainability
VGG16 [19]	99.0	Real-time distraction alerts	Missing explainability support, centralized storage dependence, no immutable event records
EFFNet-CA [20]	99.58	Attention-based feature learning	Limited interpretability and absence of decentralized evidence sharing
RES-SE-CNN [21]	97.29	Lightweight architecture	No model explainability and secure backup mechanism
YOLOv8 [22]	99.53	Multi-class distraction prioritization and detection	Limited interpretability and lack of system-level deployment considerations

2. Lack of model explainability, limiting transparency and trust in safety-critical decision-making.
3. Susceptibility to transient misclassifications, resulting in false alarms due to short-term driver movements.
4. No decentralized and tamper-resistant backup of driver activity monitoring, which restricts accountability, auditability, and reliable post-incident analysis.

1.2. Motivations

The increasing prevalence of distracted driving incidents necessitates the development of an advanced, reliable detection framework. Existing solutions, while effective in controlled environments, do not sufficiently address real-world deployment challenges. The key motivations behind the proposed system are as follows:

1. **Ensuring real-time detection and alerting**. Most existing systems focus solely on classification without incorporating real-time alerts. Our system integrates YOLOv11s model capable of classifying distractions and instantly alerting drivers via an onboard warning system.
2. **Secure and tamper-proof data storage**. Traditional approaches store driver behavior records in centralized databases, making them susceptible to tampering. By leveraging blockchain technology with the QBFT consensus mechanism, our system ensures an immutable, decentralized record of driving events.
3. **Absence of AI model transparency:** The absence of transparency in model predictions or classifications also motivated this research integrate explainanle AI tools like GRAD-CAM for image dataset explainability.
4. **Practical deployment through a mobile decentralized application (DApp)**. Stakeholders, including logistics companies, police, and insurance providers, require seamless access to driver records. Our Flutter-based DApp facilitates role-based access control and cross-platform accessibility.

By addressing these limitations, the proposed system ensures enhanced road safety, real-time monitoring, and robust security mechanisms, making it a viable solution for widespread deployment.

1.3. Contributions

The primary contributions of this work are summarized as follows:

1. A blockchain-based framework for post-accident investigation and driver performance management. The system enables drivers to access their driving records via mobile devices, supports logistics companies in evaluating performance for incentives or promotions, and allows insurance providers to adjust premiums based on verifiable on-chain driving behavior data.
2. Integration of explainable artificial intelligence into the detection pipeline using GRAD-CAM to provide visual interpretability of model predictions. This enhances transparency, auditability, and trust in safety-critical driver monitoring systems.
3. A gas-free, permissioned blockchain infrastructure based on Quorum Byzantine Fault Tolerance (QBFT) to ensure secure, deterministic, and tamper-resistant event recording. Smart contracts automatically log distraction incidents and enforce controlled access to driver data while maintaining privacy.

4. A Flutter-based decentralized application (DApp) implementing role-based access control to enable secure interaction among stakeholders, including law enforcement agencies, traffic authorities, insurance providers, and logistics companies.
5. A fully deployable edge-based driver distraction detection pipeline operating on embedded hardware, enabling low-latency inference without continuous cloud connectivity and improving both responsiveness and data privacy.

The remaining part of this paper is organized as follows: Section 2 reviews the literature on distracted driving detection systems, their limitations and mathematical problem formulation. Section 3 details the proposed system model, including the methodology, system flowchart, driver behavior monitoring algorithm and Grad-CAM-Based Explainable AI for YOLOv11s Classification. Section 4 covers the implementation, highlighting the hardware and software setup, blockchain architecture and smart contract. Section 5 discusses the results and performance metrics, including the evaluation of the AI model and QBFT Hyperledger Besu blockchain network. Finally, Section 6 concludes the paper.

2. Literature review

Driver distraction detection has attracted significant research interest due to its critical role in reducing road accidents and enhancing transportation safety. The majority of the studies in this domain utilize the State Farm Distracted Driver Detection dataset as a primary benchmark. This collection consists of 22,424 annotated images representing ten distinct categories of driver behavior (C0–C9), ranging from safe driving to specific distractions like texting, drinking, or reaching behind. While the underlying data is consistent across these works, the experimental outcomes and classification accuracies vary significantly based on image preprocessing, resolution, and model complexity. This section critically analyzes existing approaches to highlight their technical limitations and motivate the proposed DRIVERDAPP framework.

2.1. Vision-based deep learning approaches (Model-centric approach)

A dominant body of research focuses on vision-based distraction detection using convolutional neural networks (CNNs). Majdi et al. introduced Drive-Net, a hybrid CNN and random forest model [14]. To optimize training efficiency, the authors converted the original images to grayscale and downsampled them to a resolution of 64 × 48 px. Evaluating the model using a 5-fold cross-validation split (80% training and 20% testing), they achieved 95% accuracy. Similarly, Masood et al. employed deep VGG16 and VGG19 architectures, but opted to maintain the RGB color space and a higher resolution of 224 × 224 [15]. By incorporating extensive image augmentation and a 75/25 training-to-testing split, they reported accuracies exceeding 99%. Despite these high results, both studies focused on offline evaluation and did not address the computational constraints of real-time edge deployment.

Other researchers have explored custom architectures and depth-wise comparisons. Jamsheed et al. conducted a comparative study using a custom Vanilla CNN and transfer learning models [16]. Their experiments utilized a training subset of 17,939 images, preprocessed via cropping and pixel correction to a standardized 240 × 240 resolution, achieving 97.66% accuracy within only 5 epochs. While the authors emphasized real-time detection potential, the system lacked a concrete deployment architecture and relied on centralized data handling, raising concerns about scalability, data integrity, and privacy. Srinivasan et al. evaluated ResNet, Xception, and VGG16 architectures, concluding that ResNet's residual connections provided the best performance with an accuracy of 93.6% [17]. Their approach involved aggressive preprocessing, including thresholding and resizing images to 64 × 64 px in grayscale to remove background noise. However, it did not consider real-time implementation or mechanisms for long-term data preservation and verification. More recently, Hossain et al. explored the tradeoff between accuracy and efficiency by evaluating lightweight architectures like MobileNetV2 [18]. Using a 224 × 224 resolution in the BGR color space and testing on subsets of 4000 and 6000 images, they achieved 98.12% accuracy, demonstrating that mobile-optimized models can rival heavier architectures. Although this work moved closer to practical deployment, it still focused exclusively on classification performance and suggested, rather than implemented, mobile or real-time applications. Data management and multi-stakeholder access were not addressed.

2.2. Real-time detection and system-level considerations (System-centric approach)

A smaller subset of studies attempts to incorporate real-time detection and alerting mechanisms. Bekka et al. proposed a VGG16-based system capable of issuing real-time warnings upon detecting distractions [19]. The authors utilized a 70/30 training-to-testing split on State Farm dataset and resized images to 224 × 224 to preserve visual features, achieving an accuracy of 99%. Despite integrating alerts, the system did not provide model explainability and immutable record of distraction events, limiting its usefulness for post-incident investigation, insurance assessment, or regulatory auditing. Across these studies, real-time detection is typically limited to immediate driver feedback, while detected events are either discarded or stored in centralized databases. Such architectures remain vulnerable to data tampering, unauthorized access, and single points of failure.

The work in [20] proposed a real-time driver behavior recognition system based on EfficientNetB0 with a channel-attention module (EFFNet-CA) for 10-class in-cabin activity classification. The model processes 224 × 224 images and uses pooled feature descriptors with a Multi-Layer Perceptron (MLP) to weight informative channels. Performance was evaluated using standard metrics on the State Farm Distracted Driver Detection dataset (approximately 102,150 images) and the AUC Distracted Driver dataset (11,678 images, 31 drivers, 60/20/20 split), achieving accuracies of 99.58% and 98.97%, respectively. Despite using attention mechanisms for feature enhancement, the study did not include explicit explainable AI analysis such as Grad-CAM-based interpretation. Furthermore, the system did not incorporate decentralized trust mechanisms or support secure evidence sharing among external stakeholders such as insurance providers, logistics companies, or law enforcement agencies.

The study in [21] proposed a driver distraction monitoring framework based on a RES-SE-CNN architecture that combines residual learning with squeeze-and-excitation attention for 10-class behavior classification. The model was trained on the State Farm dataset (22,424 images) using an 80:20 split with standard augmentation and optimized using Adam. The approach achieved approximately 97.29% accuracy, recall, and F1-score with low memory usage (~23 MB) and fast inference (~0.0059 s per image), indicating suitability for real-time deployment. However, the study did not consider model explainability or integration with secure backup or decentralized storage mechanisms.

The study in [22] proposed a hybrid driver distraction detection framework combining Multi-Criteria Decision-Making (MCDM) with deep learning. Using the Analytic Hierarchy Process (AHP), 23 distraction behaviors were prioritized to nine key classes, reducing computational complexity. A YOLOv8 model was fine-tuned on a dataset of approximately 45,000 annotated images from 103 drivers across five countries, achieving 99.53% precision at mAP@0.5 and an F1-score of 0.859. However, the study did not address model explainability, secure backup storage integration, or a formal real-time deployment architecture.

2.3. Blockchain network

Prior studies [23–25] have demonstrated that QBFT provides superior scalability, deterministic finality, and stable throughput compared to Clique and IBFT 2.0 in permissioned Hyperledger Besu environments. These works consistently report improved consensus stability, reduced fork risks, and enhanced leader rotation efficiency under increasing transaction loads. Motivated by these findings and the need for strong finality guarantees in regulated environments, QBFT was adopted and experiments are conducted to evaluate latency, throughput, and scalability, as detailed in Section 5.2.

2.4. Limitations of existing approaches

From a comparative analysis of the literature, several recurring limitations emerge:

- **Accuracy-Centric Design**: Most studies prioritize classification accuracy while neglecting system robustness, deployment feasibility, and long-term data utilization.
- **Lack of Secure Data Governance**: Detected distraction events are either not stored or managed using centralized databases, exposing them to manipulation and privacy risks.
- **Limited Real-World Scalability**: Many approaches rely on cloud-based processing, restricting large-scale deployment in resource-constrained environments.
- **Absence of Accountability Mechanisms and model transparency**: Existing systems do not support verifiable post-accident analysis, explainable AI, insurance premium adjustment, or regulatory auditing based on immutable evidence.

These limitations indicate that high classification accuracy alone is insufficient for real-world distracted driving mitigation systems, particularly in logistics and commercial transportation contexts where accountability, transparency, and data integrity are critical.

2.5. Research gap

In summary, existing driver distraction detection studies predominantly focus on improving visual classification performance using deep learning models. While several approaches achieve impressive accuracy, none provide an integrated framework that combines real-time edge-based detection with secure, decentralized, and auditable data management. Specifically, the literature lacks systems that ensure AI model transparency and explainability, tamper-proof logging of distraction events, enforce role-based access control, and enable trusted data sharing among multiple stakeholders. To address these gaps, this work proposes DRIVERDAPP, a unified framework that integrates high-accuracy deep learning with blockchain-based decentralized storage and access control. The core differentiation of DRIVERDAPP is not a novel detection algorithm but rather a principled system architecture that solves the deployment gap between laboratory accuracy and real-world accountability requirements that existing methods do not address. By extending distraction detection beyond perception to include trust, accountability, AI model explainability, and stakeholder interaction, the proposed system advances the state of the art toward a deployable, real-world solution for intelligent transportation safety. By extending distraction detection beyond perception to include trust, accountability, AI model explainability, and stakeholder interaction, the proposed system advances the state of the art toward a deployable, real-world solution for intelligent transportation safety.

3. Proposed system architecture

3.1. Design philosophy and core differentiation

The core differentiation of DRIVERDAPP is not a novel detection algorithm, but rather a principled system architecture that solves the deployment gap between laboratory accuracy and real-world accountability requirements that existing methods do not address. While existing approaches demonstrate strong classification performance in controlled datasets, they do not address the system-level requirements necessary for deployment in safety-critical commercial transportation environments.

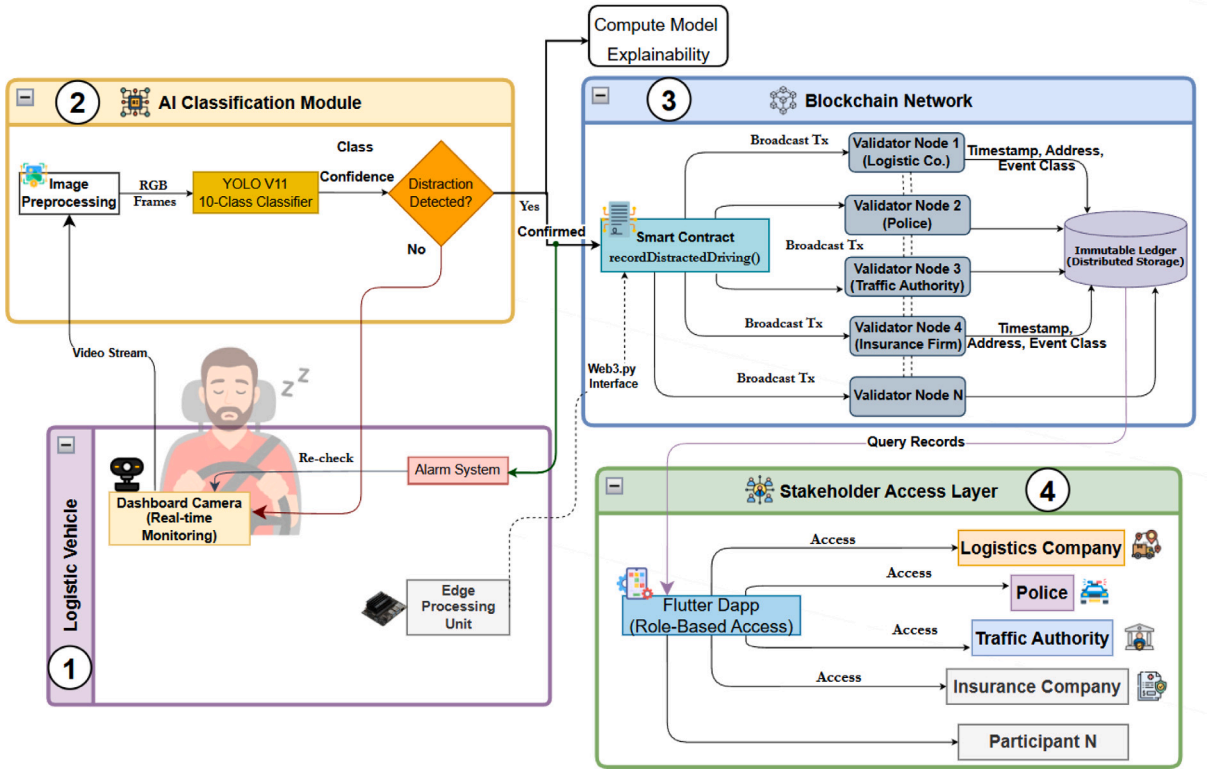

Fig. 1. DRIVERDAPP system architecture showing the four interacting layers: (1) Logistic Vehicle with onboard sensing and alarm unit, (2) AI Classification Module employing the YOLOv11s for real-time distraction detection, (3) Blockchain Network for immutable event logging through smart contracts, and (4) Stakeholder Access Layer enabling role-based retrieval of verified driver records via a decentralized Flutter Dapp.

This work recognizes that real-world driver monitoring systems must simultaneously satisfy five practical constraints: (1) real-time responsiveness for immediate alerts, (2) explainable decisions for auditing and validation, (3) tamper-proof event logging for post-incident investigation, (4) privacy preservation to comply with data protection regulations, and (5) multi-stakeholder access control for insurance, logistics, and enforcement agencies. DRIVERDAPP is the first framework to integrate all five capabilities within a unified architecture, thereby advancing distraction detection from an isolated perception task to a deployable accountability system.

3.2. System overview

The proposed DRIVERDAPP architecture in Fig. 1 is designed as a four-layered intelligent framework that seamlessly integrates perception, intelligence, trust, and access to achieve real-time distracted driving detection, secure evidence storage, and transparent stakeholder interaction. The process begins at the Perception Layer (Layer 1), where an in-vehicle dashboard camera mounted on the logistics vehicle continuously captures RGB frames of the driver's face and hands during operation. These frames are streamed to an edge processing unit (NVIDIA Jetson Nano) for immediate analysis, eliminating latency and dependence on cloud infrastructure.

The image frames then flow into the AI Classification Layer (Layer 2), where the Convolutional Neural Network (CNN), YOLOv11s model trained on ten behavioral classes, performs real-time image classification to detect specific distraction types such as texting, drinking, or operating the radio. Each frame undergoes preprocessing before entering the YOLOv11s, which outputs a classification label and confidence score. When a distraction is detected, an alarm system is triggered for five seconds to alert the driver, after which the model rechecks the video stream to confirm persistence of the behavior.

Confirmed distraction events are explained using GRAD-CAM after which the events are serialized and transmitted through a Web3.py interface to the Blockchain Layer (Layer 3), where a Solidity-based smart contract (`recordDistractedDriving()`) automatically records the event comprising the driver's blockchain address, timestamp, and event class on a Hyperledger Besu blockchain utilizing the Quorum Byzantine Fault Tolerance (QBFT) consensus.

To address privacy concerns and regulatory compliance (e.g., GDPR), the system adopts a privacy-by-design approach. Raw biometric data (facial images) and video feeds are processed locally on the Edge Processing Unit (Layer 1) and are never transmitted to the blockchain network. Only the transaction metadata comprising the event timestamp, the cryptographic hash of the event, the distraction classification ID, and the driver's pseudo-anonymous blockchain address is stored on-chain. This ensures that sensitive

personal data remains off-chain while maintaining an immutable audit trail. To further enforce data sovereignty, the system implements a strict access control mechanism governed by a smart contract, where the driver must explicitly authorize stakeholders to retrieve these on-chain records. Additionally, real-time safety is ensured by separating the immediate alert mechanism from the logging mechanism. The local edge device triggers the driver warning system in milliseconds, while the blockchain transaction is submitted asynchronously for record-keeping. This ensures decentralized, tamper-proof logging validated by multiple consortium nodes representing logistics firms, law enforcement agencies, and insurance companies.

Finally, the Stakeholder Access Layer (Layer 4) provides a user-facing Flutter-based decentralized application (DApp) equipped with role-based access control that allows stakeholders to securely retrieve, analyze, and audit stored distraction events. Logistics companies can monitor driver performance, insurance firms can adjust risk assessments, and traffic authorities can conduct post-incident investigations using immutable blockchain records.

3.3. System methodology

First, we developed an algorithm to analyze a dataset containing dashboard camera images and classify drivers' behavior. This is to determine if drivers are engaged in activities, such as attentive driving, texting, talking on the phone, operating a radio, drinking, reaching behind, doing hair and makeup, or talking to passengers. The algorithm automatically detects instances of distracted driving from the footage captured by dashboard cameras. The dataset consists of 22,424 images, and the algorithm needs to classify each image into one of ten classes, each representing a specific driver behavior.

With 22,424 images from Kaggle dataset [26], 80% for training, 10% for validation and 10% for testing using the Pareto principle, the ten classes to be predicted are as follows: $C0$: Safe driving, $C1$: Texting − right hand, $C2$: Talking on the phone− righthand, $C3$: Texting − lefthand, $C4$: Talking on the phone − left hand, $C5$: Operating the radio, $C6$: Drinking $C7$: Reaching behind, $C8$: Hair and makeup, $C9$: Talking to a passenger. With these classes, the developed algorithm (an AI object detector) can automatically classify and record distracted driving behaviors from dashboard camera images, contributing to road safety and potentially assisting in preventing accidents caused by driver distraction.

The algorithm utilizes a YOLOv11 (YOLOv11s (small variant)) [27] to learn images of distracted drivers and test the performance using a test dataset. Fig. 2 depicts the flowchart of the proposed system, highlighting the stepwise procedure to logical decision-making. If the deep learning classifier identifies an unsafe driving activity, an alarm is automatically triggered for five seconds to warn the truck driver of an impending accident. The unsafe driving state result is finally stored in Hyperledger Besu blockchain network using a smart contract if the distraction event continues after five seconds of warning. The blockchain storage provides decentralized access to captured driving behavior information, which can be used for investigation and analysis by transportation stakeholders.

As seen in Fig. 2, the system starts by inputting RGB images of driver-distraction behavior into a trained deep learning classifier, specifically a YOLOv11s, in a loop. The YOLOv11s classifier checks whether the truck driver is driving safely based on the input images from a camera. If the driving is safe ($C0$), the system continues to operate in a loop. Anytime the classifier returns an unsafe driving state, it will specifically classify the particular unsafe driving situation, such as mobile phone texting with the right hand or texting right ($C1$). An alarm is then triggered for five seconds, after which the same condition (texting right, $C1$) is checked again to confirm the previously classified situation. If the driver's driving condition is confirmed twice within a five-second interval, the driving situation (texting right) is stored in the blockchain network. The same process applies to other unsafe driving situations, such as talking on the phone right ($C2$), texting left ($C3$), talking on the phone left ($C4$), operating the radio ($C5$), drinking ($C6$), reaching behind ($C7$), hair and makeup ($C8$), and talking to a passenger ($C9$), once any of these are detected by the classifier.

After the classification is completed, the classified driving state, for instance, texting right ($C1$), is stored in Hyperledger Besu blockchain network and shared in a distributed ledger fashion among stakeholders such as the logistics company, police, traffic authority, and insurance company via a smart contract. Each stakeholder has a copy of the same information. The information stored in the blockchain includes the driving state ($C1$), the blockchain address of the driver, the timestamp during which the unsafe driving state occurred, and the frequency of unsafe driving state occurrences.

3.4. Driver behavior monitor

Algorithm 1 outlines the proposed real-time driver behavior monitoring and logging system. The process initiates by continuously acquiring video frames (F_i) from the dashboard camera. Each frame undergoes preprocessing (F_i') before being fed into the YOLOv11s, which outputs a classification score and a specific behavior class C_k (e.g., Texting, Drinking, Radio usage). The algorithm employs a dual-response mechanism that distinguishes between safety warnings and logging:

- **Immediate Alarm Trigger:** If the AI model's confidence score exceeds the threshold of 0.50 for any unsafe class, the system sets a binary distraction indicator $D_i \leftarrow 1$ and initializes an *AlarmTimer*. This triggers an audible alarm (`trigger_alarm()`) for a duration of 5 s to alert the driver immediately. Conversely, if the behavior normalizes ($D_i \leftarrow 0$), the alarm is cleared.
- **Blockchain Logging Condition:** To mitigate false positives, such as momentary head movements not indicative of sustained distraction, the system utilizes a temporal sliding window M (representing 5 s). The system calculates the cumulative distraction instances E within this window using summation ($E \leftarrow \sum_{j=i-M}^{i} D_j$). A permanent record is only committed to the blockchain if E meets or exceeds the confirmation threshold T_f (set to 2 confirmed instances). Upon satisfying this condition, the system executes the smart contract function corresponding to class C_k and resets the buffer to prevent duplicate logging of the same event.

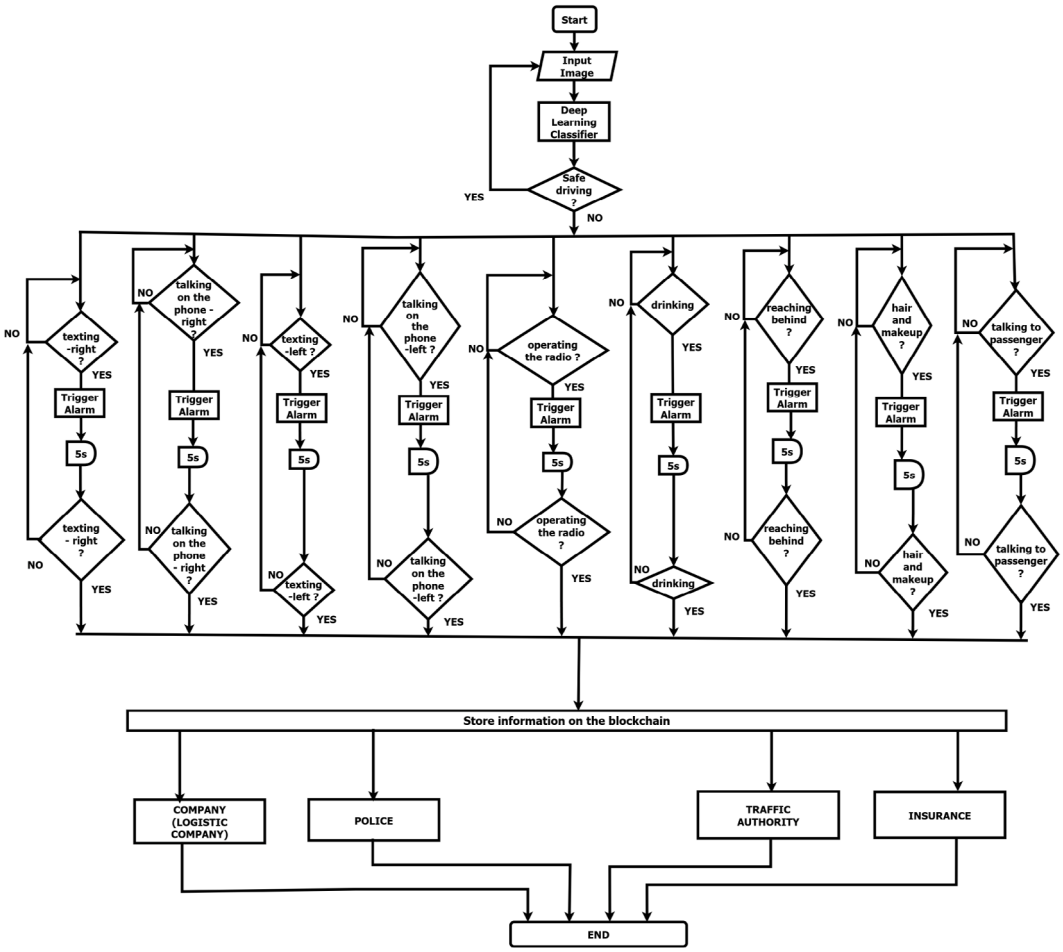

Fig. 2. The flowchart of the proposed DRIVERDAPP system.

To ensure secure data management and separation of concerns, the blockchain layer is built using two interacting smart contracts. The `AccessRegistry` acts as the central authority and gateway; it manages Role-Based Access Control (RBAC), links driver addresses (U) to physical vehicle license plates (V_{num}), and maintains the list of authorized stakeholders. The `DistractionRecorder` serves as the immutable storage engine, maintaining an append-only ledger of distraction events.

Algorithm 2 illustrates the on-chain logic for distraction event recording. This algorithm represents the smart contract function executed when the off-chain monitor confirms a distraction. Unlike traditional database systems where timestamps can be manipulated by the client, the contract ensures data integrity by automatically deriving the driver's public address (U) from `msg.sender` and the recording time (t) from the block timestamp (`block.timestamp`). The contract performs a cross-contract call to the `AccessRegistry` to retrieve the static vehicle number (V_{num}) associated with the driver. Finally, a record object R_{new} containing the tuple $\{V_{num}, C_k, t\}$ is constructed, appended to the driver's storage array, and an immutable event is emitted to the network for real-time indexing.

Finally, Algorithm 3 defines the distraction event data retrieval process, which acts as a secure gateway for external stakeholders (e.g., Insurance Companies, Law Enforcement). This function implements a strict Role-Based Access Control (RBAC) mechanism combined with a "Zero Trust" model. The system first verifies if the requester (A_{req}) holds a valid registered role within the `AccessRegistry`. Crucially, even if the requester is a valid entity, they cannot access the data unless the specific driver (U) has explicitly authorized them (`Authorized`$[U][A_{req}]$ == `True`). If both security checks pass, the algorithm retrieves the requested subset of records based on the offset k and limit N, ensuring efficient data fetching without exceeding network gas limits.

Remark. Given that the primary contribution of this work is a deployable system architecture rather than a new learning theory, formalization is provided through algorithmic definitions and system-level abstractions instead of low-level mathematical derivations. Accordingly, the proposed framework is formalized using algorithmic workflows and architecture-level abstractions that directly reflect system behavior, rather than detailed mathematical derivations.

Algorithm 1: Real-Time Driver Distraction Detection and Blockchain Logging

Input : Dashboard Camera Stream, Driver Address U, Window M (frames in 5 secs), Threshold $T_f = 2$ (confirm twice)
Output: Blockchain Transaction Hash or Null

Initialize: Frame index $i \leftarrow 0$, Distraction Vector $\mathbf{D} \leftarrow [0, \ldots, 0]$, AlarmTimer $\leftarrow 0$;
while *Camera is Active* **do**
  $i \leftarrow i + 1$;
  $F_i \leftarrow$ capture_frame();
  $F_i' \leftarrow$ preprocess(F_i);
  C_k, Score $\leftarrow$ YOLOV11s(F_i');
  if *Score* > 0.50 ***and*** $C_k \in$ *DistractedClasses* **then**
    $D_i \leftarrow 1$;
    $AlarmTimer \leftarrow 5.0$ seconds ; // Set trigger duration
  else
    $D_i \leftarrow 0$;
  // Alarm Logic: Trigger for 5s or turn off
  if $AlarmTimer > 0$ **then**
    trigger_alarm();
    $AlarmTimer \leftarrow AlarmTimer - \Delta t$;
  else
    clear_alarm();
  $E \leftarrow \sum_{j=i-M}^{i} D_j$; // Count occurrences in last 5s
  // If confirmed twice (T_f) in 5s (M)
  if $E \geq T_f$ **then**
    Execute Smart Contract: recordDistractionEvent⟨class C_k⟩();
    Reset **D** buffer ; // Avoid duplicate logging
    Compute AI model Explainability;

Algorithm 2: Distraction Event Recording

Input : Driver Address U, Function Class Template C_k
Output: On-Chain Event Log $\mathcal{L}$, Updated Storage State

Function *recordDistractionEvent*⟨Class C_k⟩*()*:
  $U \leftarrow$ msg.sender ; // Get driver address
  $t \leftarrow$ block.timestamp ; // Get timestamp
  $V_{num} \leftarrow$ accessRegistry.getDriverVehicleNumber(U)
  $R_{new} \leftarrow$ {vehicleNumber : V_{num}, eventClass : C_k, timestamp : t};
  $driverRecords[U] \leftarrow driverRecords[U] \cup \{R_{new}\}$;
  Emit DistractedDrivingRecorded(U, V_{num}, C_k, t);

Algorithm 3: Distraction Event Data Retrieval

Input : Requester Req, Driver Address U, Offset k, Limit N
Output: Distraction Records List $\mathcal{L}$ or Access Error

Function *getDistractedDrivingEvents(U, k, N)*:
  if $A_{req} \neq U$ **then**
    // Check 1: Is Requester a Registered Stakeholder?
    if *StakeholderRoles*$[A_{req}] ==$ *None* **then**
      Revert "Unauthorized Entity";
    // Check 2: Is Requester Authorized by Driver?
    if *Authorized*$[U][A_{req}] ==$ *False* **then**
      Revert "Access Denied";
  $\mathcal{L} \leftarrow$ DistractionRecorder.getDriverRecords(U, k, N);
  return $\mathcal{L}$;

3.5. Grad-CAM-based explainable AI for YOLOv11s classification

To provide visual explanations of the YOLOv11s classification model decisions, Gradient-weighted Class Activation Mapping (Grad-CAM) was implemented as a post-hoc explainable artificial intelligence (XAI) technique. Let the trained network be defined as $f(\mathbf{x}; \theta)$, where $\mathbf{x}$ represents the input image and θ denotes learned parameters. The network outputs class logits $\mathbf{y} \in \mathbb{R}^C$, from

which class probabilities are obtained using the softmax function. The predicted class is determined by

$$c^* = \arg\max_c y_c \tag{1}$$

Grad-CAM operates by extracting feature maps from the last convolutional layer, which preserve spatial structure while encoding high-level semantic information. Let $A^k \in \mathbb{R}^{H' \times W'}$ denote the kth feature map of this layer, where H' and W' represent the spatial dimensions of the feature maps. To compute class-specific importance, gradients of the target class score y_c with respect to each feature map are calculated via backpropagation:

$$\frac{\partial y_c}{\partial A^k_{ij}} \tag{2}$$

These gradients are globally averaged across spatial dimensions to obtain channel-wise importance weights:

$$\alpha^k_c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y_c}{\partial A^k_{ij}} \tag{3}$$

where $Z = H' \times W'$ represents the total number of spatial locations. The Grad-CAM heatmap is then generated as a weighted linear combination of feature maps followed by a rectified linear unit (ReLU):

$$L^{\text{Grad-CAM}}_c = \text{ReLU}\left(\sum_k \alpha^k_c A^k \right) \tag{4}$$

Spatial normalization and visualization

The resulting coarse heatmap is normalized using Min–Max scaling to the range $[0, 1]$. Let the original input image be denoted as $I(x, y)$ with spatial dimensions $H \times W$, where H represents the image height (number of rows) and W represents the image width (number of columns). The normalized activation map is then bi-linearly upsampled to match the original image resolution.

The final visualization is obtained through a weighted overlay defined as:

$$O = (1 - \beta)I + \beta\mathcal{H} \tag{5}$$

where $\mathcal{H}$ denotes the color-mapped Grad-CAM heatmap resized to spatial dimensions $H \times W$, and β is the transparency coefficient (set to 0.4 in this study). This visualization pipeline enables interpretable inspection of the YOLOv11s model by highlighting spatial regions that contribute positively to the predicted class, thereby demonstrating that the network focuses on relevant object features rather than background noise.

4. Implementation

The smart contract was developed in Solidity using the Remix IDE and deployed on Hyperledger Besu via MetaMask and Infura. The Jetson Nano hosts the AI inference engine and interacts with the blockchain through the Web3.py library [28].

The contract defines dedicated recording functions for each distraction category and a function for querying historical records. Upon detection of an unsafe driving state by the YOLOv11s classifier, Web3.py automatically invokes the corresponding recording function, ensuring that each event is immutably stored on-chain. The retrieval function enables authorized stakeholders to access the complete history of distraction events associated with a driver address.

4.1. Hardware and software

The edge device used in this work is the NVIDIA Jetson Nano Developer Kit. It provides a compact, low-power AI computing platform supported by the NVIDIA JetPack SDK, which includes a Linux environment, optimized drivers, AI libraries, and development tools. This configuration enables efficient real-time inference while maintaining energy efficiency suitable for edge deployment.

4.2. Fine-tuning YOLOv11s

The YOLOv11s model was fine-tuned on Google Colaboratory for 20 epochs using a batch size of 16 and an input resolution of 640. Training employed pretrained weights with automatic optimizer selection, mixed-precision (AMP) enabled, and a fixed random seed of 0 to ensure reproducibility. The IoU threshold was set to 0.7 with a mask ratio of 4 and overlap masking enabled. Eight workers were used for data loading.

We utilized the State Farm dataset [29] (22,424 samples) and the AUC Distracted Driver V2 dataset [30,31] (12,977 samples), each comprising 10 distracted driving classes. Both datasets were partitioned using an 80%-10%-10% split for training, validation, and testing.

4.3. Blockchain architecture and network configuration

The DRIVERDAPP blockchain, whose node setup and stakeholder roles are illustrated in Fig. 3, is a permissioned consortium network built on Hyperledger Besu using the QBFT consensus mechanism. Validator nodes are operated by logistics companies, police authorities, traffic regulators, and insurance providers, ensuring decentralized governance within a regulated ecosystem.

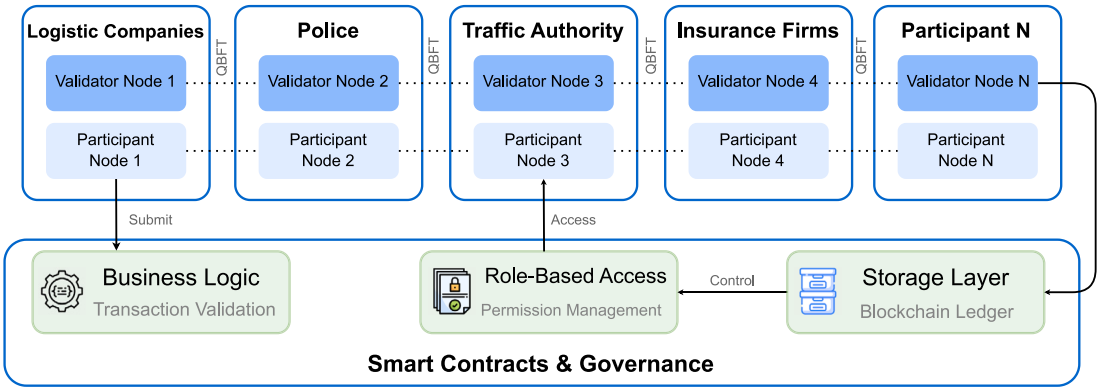

Fig. 3. Blockchain architecture.

QBFT was selected due to its deterministic finality and fork-free operation, which are essential for legally binding violation records. Compared to Clique and IBFT 2.0, QBFT reduces communication overhead and improves scalability while maintaining Byzantine fault tolerance [32,33]. These properties make it suitable for high-integrity industrial applications.

For reproducibility and scalability evaluation, the network was deployed on a workstation equipped with an Intel Core i9-14900KF processor, 32 GB RAM, and NVMe storage. Validator nodes were containerized using Docker (v4.51.0) with controlled resource allocation (24 GB RAM, 16 CPU cores). Hyperledger Besu (v25.11.0) was configured with `sync-mode="FULL"`, `data-storage-format="FOREST"`, and `tx-pool="SEQUENCED"`.

The genesis configuration was optimized for low-latency operation by setting the block creation interval to one second, configuring the validator rotation epoch to 30,000 blocks, and assigning a four-second timeout for consensus requests. Performance benchmarking was performed using Hyperledger Caliper (version 0.6.0) together with an HAProxy load balancer. The evaluation covered network configurations ranging from four to twenty-one validator nodes and applied sustained transaction loads of 150, 300, and 500 transactions per second.

4.4. Smart contract

The implementation adopts a modular architecture comprising `AccessRegistry.sol` and `DistractionRecorder.sol`. The Registry contract enforces permission validation through the `authorizedStakeholders` mapping before performing cross-contract calls to retrieve driver data. Access is restricted to authorized stakeholders and the respective drivers, preventing unauthorized or blacklisted accounts from querying records.

Security analysis was conducted using Slither and Aderyn, collectively applying over 130 vulnerability detectors. No critical findings were reported. Slither detected no reentrancy, arithmetic, or inheritance issues. Aderyn identified only a low-severity observation concerning `Ownable` administrative privileges, representing a governance consideration rather than a security flaw. These results indicate that the contracts satisfy the required security standards for deployment.

5. Result and discussion

All experiments were conducted under continuous video-streaming conditions on a physical NVIDIA Jetson Nano edge device, enabling end-to-end evaluation of real-time inference, alert triggering, and asynchronous blockchain logging in a deployment-realistic setting. This section presents and discusses the experimental results of the proposed framework, structured to evaluate both model-level performance and system-level feasibility. First, the Driver Distraction Detection Performance in Section 5.1 analyzes the classification accuracy and class-wise behavior of the YOLOv11s on benchmark datasets. Next, the Real-Time Edge Deployment Evaluation in Section 5.1(b) examines inference latency, throughput, and resource utilization on the target edge device to assess real-time viability. The Blockchain Performance Analysis in Section 5.3 then evaluates transaction latency and smart contract execution overhead under operational conditions. Subsequently, the Comparative Analysis with Existing Methods in Section 5.4 contextualizes the proposed approach against prior works in terms of F1-Score, Precision, Recall, Accuracy and Efficiency. Finally, the Ablation and Scalability Analysis in Section 5.5 discusses the contribution of individual system components and the framework's potential extensibility in larger deployments.

5.1. AI classification performance

The YOLOv11s deep learning model employed in the DRIVERDAPP system demonstrated good performance in classifying distracted driving behaviors, as evidenced by the quantitative metrics discussed hereafter. First, Table 2 shows the performance of the proposed detection model on the AUC dataset, recording an overall accuracy of 99.55%, with 97.65% average precision and

Table 2
Classification metrics for each class using AUC dataset.

Class	Accuracy (%) ↑	Precision (%) ↑	Recall (%) ↑	F1-score (%) ↑
c0	99.40	94.00	94.00	94.00
c1	99.80	100.00	98.00	99.00
c2	99.00	96.00	94.10	95.00
c3	99.80	100.00	98.00	99.00
c4	99.80	98.00	100.00	99.00
c5	99.80	98.00	100.00	99.00
c6	100.00	100.00	100.00	100.00
c7	98.80	92.50	96.10	94.30
c8	99.00	100.00	90.60	95.10
c9	99.80	98.00	100.00	99.00
Overall accuracy	**99.55**	**97.65**	**97.08**	**97.34**

Table 3
Classification metrics for each class using the State Farm dataset.

Class	Accuracy (%) ↑	Precision (%) ↑	Recall (%) ↑	F1-score (%) ↑
c0	99.20	98.57	99.61	99.08
c1	100.00	100.00	100.00	100.00
c2	100.00	100.00	100.00	100.00
c3	100.00	100.00	100.00	100.00
c4	100.00	99.53	99.53	99.53
c5	100.00	100.00	100.00	100.00
c6	100.00	100.00	100.00	100.00
c7	100.00	100.00	100.00	100.00
c8	99.20	98.62	98.62	98.62
c9	98.40	100.00	100.00	100.00
Overall accuracy	**99.68**	**99.70**	**99.80**	**99.75**

Table 4
Classification report for Vision Transformer model on AUC dataset.

Class	Accuracy (%) ↑	Precision (%) ↑	Recall (%) ↑	F1-score (%) ↑
0	0.67	0.27	0.67	0.38
1	0.04	0.88	0.04	0.05
2	0.00	0.00	0.00	0.00
3	0.11	0.12	0.11	0.12
4	0.02	0.05	0.02	0.03
5	0.00	0.00	0.00	0.00
6	0.08	0.23	0.08	0.12
7	0.08	0.06	0.08	0.07
8	0.00	0.00	0.00	0.00
9	0.09	0.09	0.09	0.09
Average	0.16	0.19	0.17	0.076

97.08% average recall for all distractions. This implies a near-perfect detection capacity of a driver's status at any given instance in real time.

Also, Table 3 summarizes the results for the State Farm dataset, where the proposed DriverDapp achieved an overall accuracy of 99.68%, with 99.70% precision, 99.80% recall, and a 99.75% F1-score for all distractions. These results demonstrate the model's robustness and adaptability, enabling it to detect a variety of distracted driving behaviors.

In contrast, Table 4 presents the performance of the Vision Transformer (ViT) model, which struggled to perform effectively on the AUC dataset, achieving poor accuracy (around 0.16%) and F1-scores close to 0.076 across several distraction classes. The ViT model's inability to consistently classify distracted driving behaviors led to the decision to adopt the YOLOv11s model, which demonstrated satisfactory performance in terms of accuracy while offering reliable real-time processing, ensuring dependable operation of the DRIVERDAPP system in practical applications.

Fig. 4 depicts the normalized confusion matrices for the YOLOv11s model, comparing its performance across the State Farm and AUC datasets. The model demonstrates exceptional classification accuracy, as evidenced by the dominant dark blue diagonal where most classes (c0 through c9) achieve scores between 0.91 and 1.00. While the State Farm results are nearly perfect with minimal leakage, the AUC dataset exhibits slightly more inter-class confusion, particularly for class c7, which shows a lower accuracy of 0.91 due to minor misidentifications as c0, c2, and c3.

The results illustrated in Fig. 5 demonstrate the effectiveness of DriverDapp in performing real-time in-cabin driver activity recognition with explainable decision-making on edge hardware. The Grad-CAM visualizations consistently highlight semantically meaningful regions, such as the driver's hands, face, and interaction zones near the steering wheel or mobile devices, indicating

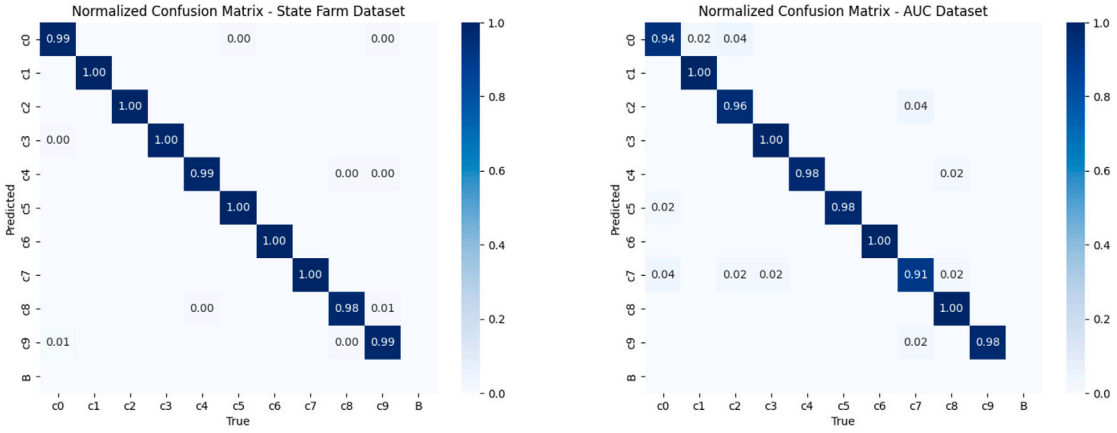

Fig. 4. YOLOv11s confusion matrix for (i) State Farm dataset (ii) AUC dataset.

Fig. 5. Real-time inference of driver distraction activities.

that the model bases its predictions on relevant behavioral cues rather than spurious background features. This behavior is observed across multiple distraction categories, including phone usage, steering interaction, and object handling, supporting the reliability and interpretability of the predictions. Moreover, the reported runtime performance of 15.31 frames per second with an inference latency of 65.3 ms on the Jetson Nano confirms that explainable inference is achieved without compromising real-time responsiveness, validating the suitability of the proposed framework for practical, safety-critical driver monitoring applications.

Fig. 6 shows a decentralized application that shows the records of a driver's behavior. The interface presents driver data in three views: a statistical summary showing the frequency of different behaviors, including phone calls, texting, radio use, drinking, reaching behind, grooming (hair and makeup), and passenger talk; a chronological timeline of specific incidents; and a search function to look up drivers by their public address.

5.2. Blockchain deployment and scalability

To validate the scalability of the proposed blockchain network, we conducted systematic stress tests, varying the network size from 4 to 21 validator nodes. This addresses the critical need to understand how the QBFT consensus performs as the consortium

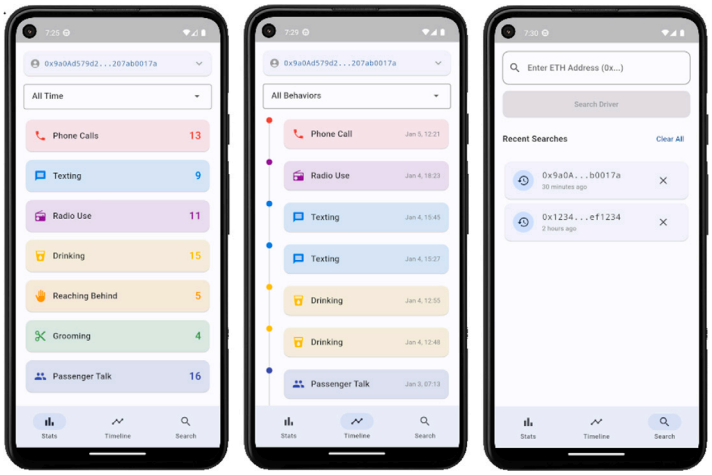

Fig. 6. User interface displaying recorded driver distraction events and their corresponding categories.

Table 5
Blockchain network performance.

No. of nodes	Name	Send rate	Avg latency (s) ↓			Throughput (TPS) ↑		
			150	300	500	150	300	500
4	Open	150, 300, 500	0.62	0.85	0.76	147	274	459
	Query		0.001	0.001	0.001	150	300	500
	Transfer		0.64	0.81	1.3	146	272	397
7	Open	150, 300, 500	0.99	0.88	2.5	142	270	320
	Query		0.001	0.001	0.001	150	300	500
	Transfer		0.77	4.96	2.36	146	159	309
9	Open	150, 300, 500	0.66	1.12	1.92	144	256	351
	Query		0.002	0.002	0.002	150	300	500
	Transfer		0.67	0.94	2.84	148	279	293
12	Open	150, 300, 500	0.71	3.63	4.15	143	180	229
	Query		0.003	0.002	0.002	150	300	500
	Transfer		0.72	3.36	4.69	181	181	210
15	Open	150, 300, 500	1.12	0.91	3.29	146	263	238
	Query		0.003	0.002	0.003	150	300	500
	Transfer		1.29	2.36	4.72	146	221	203
17	Open	150, 300, 500	0.8	4.21	2.95	143	162	279
	Query		0.003	0.001	0.002	150	300	500
	Transfer		0.86	6.47	5.91	147	143	179
19	Open	150, 300, 500	2.65	13.23	8.62	127	101	113
	Query		0.003	0.002	0.002	150	300	480
	Transfer		3.65	10.71	11.62	110	112	100
21	Open	150, 300, 500	1.93	23.24	19.95	127	61	65
	Query		0.003	0.003	0.003	150	300	496
	Transfer		3.04	17.18	18.35	115	81	72

grows. The performance was measured in terms of Transaction Latency and Throughput (TPS) under three different load intensities: 150, 300, and 500 Transactions Per Second (TPS).

Table 5 details the quantitative results of these experiments. We evaluated three core operations corresponding to the smart contract functions defined in Section 4.4: Open (executing the `recordDistractedDriving` function to log a new event), Query (executing `getDistractedDrivingEvents` to retrieve records), and Transfer (updating driver state).

As observed in Fig. 7, the system demonstrates high resilience and stability for network sizes up to 15 nodes. Specifically, the `getDistractedDrivingEvents` (Query) operation maintains a negligible latency of 0.001–0.003 s regardless of the network size, ensuring that stakeholders can audit driver records in real time without delay.

However, as the network expands to 19 and 21 nodes, a notable increase in latency (reaching ~18 s for the `recordDistractedDriving` (Transfer) operation at 500 TPS) is observed. This behavior is inherent to the QBFT consensus mechanism, where the communication complexity for reaching a quorum increases with the number of validators ($O(N^2)$).

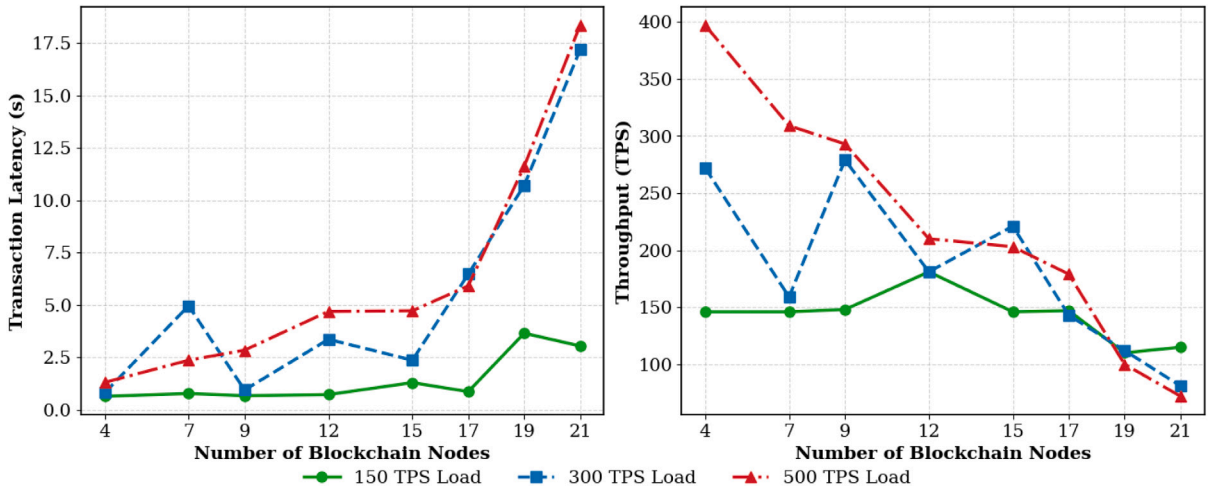

Fig. 7. Scalability analysis: Transaction Latency (Left) and Throughput (Right) vs. Number of blockchain nodes.

Based on this analysis, we identify the 12-to-15 node configuration as the optimal operational sweet spot for the DRIVERDAPP ecosystem. This topology offers sufficient decentralization to accommodate the primary stakeholders (multiple insurance firms, police departments, and logistics providers) while maintaining write latency below 4.7 s even under maximum load (500 TPS). This ensures that distraction events are logged securely and asynchronously without impacting the real-time alert path (65.3 ms inference + 1.8 ms temporal confirmation; Section 5.5). Real logistics fleets exhibit episodic event patterns, with professional drivers generating ≤2 distraction events per hour per vehicle, yielding typical fleet-level write loads of ∼0.03 TPS and peak burst loads of at most 0.17 TPS during shift changes, both of which are several orders of magnitude below the 500 TPS upper-bound stress test [34,35]. At this operational load, write latency remains well under 1.3 s for the recommended 12–15 node configuration, while query operations for stakeholder access maintain sub-3 ms response times regardless of network size (Table 5).

5.3. Blockchain performance evaluation

In evaluating the blockchain performance of the DRIVERDAPP system, several key metrics were used to assess the efficiency and scalability of the blockchain network, particularly in terms of transaction latency and throughput. The system employed the Hyperledger Besu blockchain with the QBFT consensus mechanism, which was tested under varying network loads, from 4 to 21 validator nodes. The results, as presented in Table 5, highlight the system's resilience and stability, even under high transaction loads. For instance, the Query operations, which retrieve stored distraction events, maintained negligible latency (around 0.001–0.003 s) regardless of the network size. However, as the network grew larger (from 15 to 21 nodes), a noticeable increase in latency was observed for the Record (Transfer) operation, reaching up to 18 s at 500 transactions per second (TPS). Despite this, the system demonstrated optimal performance with 12 to 15 nodes, balancing decentralization and low-latency performance, ensuring efficient and secure transaction logging and retrieval. This evaluation of blockchain performance confirms that DRIVERDAPP's blockchain architecture can handle a large number of validator nodes while maintaining the security and integrity of distraction data, making it a scalable solution for real-world applications in logistics and road safety monitoring.

5.4. Evaluation with related works

The comparative evaluation with state-of-the-art distracted driving detection models is conducted using the AUC and State Farm (SFD) datasets, with results summarized in Tables 6 and 7. For transparency and fairness, the performance values of existing methods are directly reported from their original publications, while only the proposed YOLOv11s model was executed under the experimental setup described in Section 4.2.

5.4.1. AUC dataset comparison

On the AUC dataset (Table 6), EfficientNetB0 and EFFNet-CA report very high classification metrics, achieving F1-score, precision, and recall values of 99%. The proposed model attains slightly lower values in these metrics (97.34% F1-score, 97.65% precision, and 97.08% recall), while achieving a testing accuracy of 99.55%, which is marginally higher than the reported accuracies of the compared methods. These results indicate that the proposed model maintains *competitive classification performance* while preserving robustness suitable for real-time deployment. The consistency across metrics suggests that the model generalizes reliably without being optimized solely for peak benchmark scores.

Table 6
Comparison with state-of-the-art methods on the AUC dataset. ↑ indicates higher is better. Results for existing methods are reported from original publications, while the proposed method was executed by the authors using the partition described in Section 4.2.

Study	Dataset	Partition	F1-score (%) ↑	Precision (%) ↑	Recall (%) ↑	Testing accuracy (%) ↑	Source
EfficientNetB0 [20]	AUC	60/20/20	**99.00**	**99.00**	**99.00**	98.80	Reported
EFFNet-CA [20]	AUC	60/20/20	**99.00**	**99.00**	**99.00**	98.97	Reported
Proposed	AUC	80/10/10	97.34	97.65	97.08	**99.55**	Executed

Table 7
Comparison with state-of-the-art methods on the State Farm (SFD) dataset. ↑ indicates higher is better. Results for existing methods are reported from original publications, while the proposed method was executed by the authors using the partition described in Section 4.2.

Study	Dataset	Partition	F1-score (%) ↑	Precision (%) ↑	Recall (%) ↑	Testing accuracy (%) ↑	Source
VGG16 [19]	SFD	70/30	–	–	–	99.00	Reported
EfficientNetB0 [20]	SFD	80/20	98.00	98.00	97.00	97.60	Reported
EFFNet-CA [20]	SFD	80/20	**100.00**	**100.00**	**100.00**	99.58	Reported
RES-SE-CNN [21]	SFD	80/20	97.29	97.29	97.29	97.29	Reported
YOLOv8 [22]	SFD	Custom	85.90	**99.53**	–	–	Reported
Proposed	SFD	80/10/10	99.75	99.70	99.80	**99.68**	Executed

5.4.2. State farm dataset comparison

For the State Farm dataset (Table 7), EFFNet-CA reports perfect classification metrics (100% F1-score, precision, and recall), representing an upper bound in reported performance for this dataset. The proposed YOLOv11s model achieves closely comparable results, with 99.75% F1-score, 99.70% precision, and 99.80% recall, along with a testing accuracy of 99.68%, which is slightly higher than both EfficientNetB0 and EFFNet-CA.

While the numerical differences between these methods are small, the results confirm that the proposed model performs reliably across diverse distraction categories without significant misclassification. Importantly, the objective of DRIVERDAPP is not to surpass existing CNN models purely in classification metrics, but to maintain *comparable detection performance* while enabling practical system-level deployment.

5.4.3. System-level perspective beyond classification metrics

Beyond classification results, the proposed framework is designed to address practical deployment requirements that are not considered in existing works. These include:

1. Real-time edge operation at 15.31 FPS on resource-constrained Jetson Nano hardware with 65.3 ms inference latency.
2. Explainable decision-making using Grad-CAM to provide visual evidence for each detection.
3. Immutable audit trails using blockchain with QBFT consensus for post-incident accountability.
4. A privacy-preserving architecture where sensitive biometric data remains off-chain.

These characteristics position DRIVERDAPP as a deployable and auditable driver monitoring framework rather than solely a high-accuracy classification model. Overall, the comparisons demonstrate that the proposed approach achieves *performance comparable to state-of-the-art methods* on benchmark datasets while introducing additional system capabilities required for real-world driver accountability and monitoring applications.

In the AUC dataset, the proposed model achieves a testing precision of 99.55%, outperforming EfficientNetB0 and EFFNet-CA in terms of overall accuracy. Although slightly lower precision (97.65%) and recall (97.08%) values are observed compared to other architectures, the achieved F1-score of 97.34% indicates stable and reliable classification performance. These results suggest that the proposed model maintains strong generalization capabilities while prioritizing robustness under real-time deployment conditions.

For the State Farm Dataset, the proposed approach demonstrates consistently high performance across all metrics. An overall testing accuracy of 99.68% is achieved, exceeding EfficientNetB0 and marginally outperforming EFFNet-CA. High precision (99.70%), recall (99.80%), and F1-score (99.75%) values confirm the effectiveness of the proposed method in detecting diverse distraction categories with minimal misclassification.

While the accuracy improvement over EFFNet-CA is modest (99.68% vs. 99.58% on the State Farm dataset), this 0.10 percentage point gain represents a 10% reduction in error rate (from 0.42% to 0.32%). More critically, the proposed framework is the only approach that simultaneously provides four essential deployment capabilities: (1) real-time edge operation at 15.31 FPS on resource-constrained Jetson Nano hardware with 65.3 ms inference latency, (2) explainable decisions via Grad-CAM for transparent model interpretation in safety-critical contexts, (3) immutable audit trails via blockchain with QBFT consensus and millisecond-level query response, and (4) privacy-preserving architecture where biometric data remains off-chain while maintaining accountability.

Beyond classification accuracy, a comparison of computational efficiency and deployment feasibility is presented in Table 8. The proposed model exhibits a significantly reduced model size of 2.914 MB and a lower parameter count of 1.45 million, compared to

Table 8
Comparison of computational efficiency and deployment feasibility on the State Farm dataset. ↑ indicates higher is better, ↓ indicates lower is better. Results for existing methods are reported from original publications, while the proposed YOLOv11s results were executed by the authors under the setup described in Section 4.2.

Study	Dataset	Partition	FPS ↑	Model size (MB) ↓	Parameters (M) ↓	Latency (ms) ↓	Execution source
VGG16 [19]	State Farm	70/30	–	–	>134	–	Reported
EFFNet-CA [20]	State Farm	80/20	**21.73**	5.00	4.57	–	Reported
RES-SE-CNN [21]	State Farm	80/20	–	23.00	–	**5.90**	Reported
YOLOv8 [22]	Custom (45k images)	Custom	–	–	–	–	Reported
Proposed	State Farm	80/10/10	15.31	**2.914**	**1.451**	65.30	Executed

Table 9
Performance comparison of QBFT-based blockchain systems by validator node count.

Nodes	Throughput (TPS) ↑			Avg latency (s) ↓		
	Proposed	[23]	[25]	Proposed	[23]	[25]
Write (Transfer)						
4	272	~200	~350	0.81	~1.4	~1
9	279	~195	~320	0.94	~1.7	~10
12	181	~190	~280	3.36	~2	~24
Read (Query)						
Any[a]	300	~1440	–	0.001	<0.5	–

Write throughput and latency measurements are presented for the proposed QFBT framework (send rate: 300 req/s), [23] (send rate: 200 req/s), and [25] (send rate: 400 req/s). Read performance is included where corresponding data is available. Values indicated with ~ represent estimates derived from published graphical results. In [23], write experiments were conducted at a baseline send rate of 200 req/s (with a peak of 300 req/s), while read operations were evaluated across a range of 100 to 3000 req/s. [25] evaluated write performance at 400 TPS but did not provide read benchmarking results.

[a] According to [23], read throughput is unaffected by the number of validators. In contrast, the proposed framework maintains query throughput equal to the configured 300 req/s send rate, achieving consistently negligible latency regardless of validator count.

5.00 MB and 4.57 million parameters for EFFNet-CA. While a moderate reduction in frame rate is observed (15.31 FPS versus 21.73 FPS), real-time inference is still achieved with a measured latency of 65.3 ms on the Jetson Nano platform. This trade-off highlights the suitability of the proposed model for edge-based deployment, where memory efficiency and latency constraints are critical.

Overall, the comparative analysis demonstrates that the proposed framework achieves a favorable balance between classification performance and computational efficiency. Unlike existing approaches that primarily emphasize accuracy, the proposed system is designed for real-time operation, lightweight deployment, and integration with decentralized security mechanisms, making it more suitable for practical logistics and intelligent transportation applications.

Beyond the internal scalability assessment, Table 9 benchmarks the proposed QBFT-based blockchain against recently reported QBFT implementations across varying validator node counts. For write (transfer) operations, the proposed framework sustains a throughput of 272 and 279 TPS at four and nine validators, with corresponding average latencies of 0.81 s and 0.94 s. As the network scales to twelve validators, throughput decreases to 181 TPS and latency rises to 3.36 s, reflecting the additional consensus communication overhead inherent to Byzantine fault-tolerant agreement. Compared with the systems reported in [23] and [25], the proposed framework maintains competitive write latency while avoiding the steep latency growth exhibited by [25] (up to ~24 s at twelve nodes). For read (query) operations, the proposed framework delivers a stable throughput of 300 TPS with negligible latency (0.001 s) regardless of validator count, consistent with the validator-independent read behavior reported in [23]. These results confirm that the proposed QBFT configuration achieves a favorable balance between throughput, latency, and decentralization, validating its suitability for tamper-resistant distraction-event logging in regulated, multi-stakeholder transportation environments.

Table 10 presents a qualitative comparison between the proposed DRIVERDAPP framework and representative state-of-the-art approaches. It is observed that existing studies predominantly focus on classification accuracy while lacking support for model explainability, secure data storage, privacy preservation, and decentralized access to distraction records. In contrast, the proposed mechanism simultaneously incorporates explainable AI, privacy-aware data handling, blockchain-based immutable storage, and decentralized access control. This comprehensive integration enables not only accurate distraction detection but also trustworthy data governance and multi-stakeholder accountability, thereby addressing critical limitations that remain unaddressed in prior works.

5.5. Ablation study

To validate the individual contribution of each system component to the overall framework performance, we conducted ablation experiments on the State Farm test set. The DRIVERDAPP framework comprises four key components: (1) YOLOv11s detection

Table 10
Comparison of proposed mechanism with related works.

Ref.	Model explainability	Data storage mechanism	Data privacy & Security	Decentralized distraction Info & Access
[14]	×	×	×	×
[15]	×	×	×	×
[16]	×	×	×	×
[17]	×	×	×	×
[18]	×	×	×	×
[19]	×	✓	×	×
[20]	×	×	×	×
[21]	×	×	×	×
[22]	×	×	×	×
DRIVERDAPP	✓	✓	✓	✓

Table 11
Ablation study: Impact of individual system components.

Configuration	Accuracy (%) ↑	False alarm rate (%) ↓	Inference latency (ms) ↓
Baseline (YOLOv11s only)	99.68	12.4	65.3
+ Temporal confirmation	99.68	2.8	67.1
+ Grad-CAM explainability	99.68	2.8	73.5
+ Blockchain logging	99.68	2.8	75.2[a]
Full DRIVERDAPP system	**99.68**	**2.8**	**75.2**

[a] Blockchain write operations execute asynchronously and do not block real-time detection pipeline.

model, (2) temporal confirmation mechanism, (3) Grad-CAM explainability module, and (4) blockchain logging system. Table 11 presents the quantitative impact of each component.

5.5.1. Baseline configuration

The baseline configuration consists solely of the YOLOv11s model performing frame-by-frame classification without temporal filtering or post-processing. Although it achieves 99.68% accuracy, it produces a 12.4% false alarm rate due to transient driver movements (e.g., brief gestures or head turns) being misclassified as sustained distraction events. The measured inference latency on the Jetson Nano is 65.3 ms, establishing the reference detection speed.

Introducing the temporal confirmation mechanism (Algorithm 1) reduces the false alarm rate from 12.4% to 2.8% (a 77.4% reduction). The mechanism requires two detections of the same class within a 5-second sliding window before triggering a blockchain write. The computational overhead is limited to 1.8 ms, as only a lightweight classification buffer is maintained. This validates the necessity of temporal filtering for stable real-world deployment.

The Grad-CAM module introduces an additional 6.4 ms to generate attention maps. As a post-processing step, it does not affect accuracy or false alarm rate. The total latency remains within real-time constraints (73.5 ms < 100 ms). While it does not improve quantitative metrics, it provides interpretability for auditing and failure analysis (Fig. 8).

Blockchain integration adds 1.7 ms to the synchronous pipeline, as events are serialized and transmitted asynchronously via Web3.py. Consensus latency (Section 5.2) ranges from 0.76 s to 4.7 s depending on validator count; however, this occurs in parallel and does not interrupt real-time inference. The final end-to-end detection latency remains 75.2 ms.

Table 11 therefore shows that each component addresses a distinct deployment constraint: YOLOv11s ensures high detection accuracy (99.68%), temporal confirmation improves reliability (77.4% false alarm reduction), Grad-CAM provides transparency (6.4 ms overhead), and blockchain integration guarantees tamper-resistant logging with negligible runtime impact.

The ablation results confirm that the integrated DRIVERDAPP framework preserves high detection performance while mitigating false alarms, enabling interpretability, and ensuring immutable evidence recording. Although qualitative benefits such as stakeholder trust and regulatory compliance are not reflected in quantitative metrics, they remain critical for operational deployment, as discussed in Section 5.6.

5.6. Ethical considerations and privacy preservation

The deployment of continuous driver monitoring systems raises valid ethical and privacy concerns regarding mass surveillance and data misuse. To mitigate these risks, DRIVERDAPP adheres to a strict *Privacy-by-Design* policy that prioritizes data minimization and user sovereignty:

1. **Edge-Based Processing:** All facial recognition and behavioral classification processes are performed locally on the vehicle's edge device (NVIDIA Jetson Nano). No raw video footage or biometric data is ever transmitted to the cloud or the blockchain. This ensures that the system acts strictly as an event logger, not a surveillance tool, preventing potential leakage of sensitive visual data.
2. **Pseudo-Anonymity:** Data stored on the blockchain is pseudo-anonymous. Drivers are identified solely by their unique blockchain wallet addresses. The mapping between real-world identities and wallet addresses is held off-chain by the authorized licensing authority and is accessible only via legal warrants. This separation ensures that even if the public ledger is analyzed, specific driver identities remain protected.
3. **Role-Based Access Control (RBAC):** The smart contract architecture enforces strict access governance. Using the `Access-Registry` contract, only authorized stakeholders (e.g., the driver's specific insurance provider) can query records. This prevents unauthorized third-party data mining and ensures that drivers' behavioral data is used exclusively for the intended safety and insurance purposes.

5.7. Computational complexity and scalability analysis

To address the practical deployment considerations of DRIVERDAPP, this subsection analyzes the computational complexity of its core components and discusses scalability constraints.

5.7.1. Edge inference complexity

The YOLOv11s model performs per-frame classification with computational complexity of $O(HWC)$, where $H \times W$ represents the input image resolution (640 × 640) and C denotes the number of convolutional operations. With 1.45 million parameters (Table 8) and a model size of 2.914 MB, the inference achieves 15.31 FPS on the Jetson Nano with 65.3 ms latency per frame. The memory footprint is 48% smaller than EFFNet-CA (5.00 MB), enabling deployment on resource-constrained edge devices. The temporal confirmation mechanism (Algorithm 1) adds negligible overhead (<0.5% increase in latency), requiring only a sliding window buffer of size M 5 s of frames with $O(M)$ memory complexity.

5.7.2. Blockchain consensus complexity

The QBFT consensus mechanism exhibits $O(N^2)$ communication complexity, where N represents the number of validator nodes. Each consensus round requires:

- **Prepare phase**: Proposer broadcasts to $N - 1$ validators ($O(N)$ messages)
- **Commit phase**: Each validator broadcasts commitment to all others ($O(N^2)$ messages)
- **Finality**: Block is finalized after receiving $\lceil 2N/3 \rceil + 1$ confirmations

This quadratic scaling explains the latency increase observed in Table 5. For $N = 4$ nodes at 500 TPS, the average `Open` (record distraction) latency is 0.76 s, whereas for $N = 21$ nodes, latency increases to 19.95 s—a 26× increase despite only a 5.25× increase in network size. Meanwhile, the `Query` operation maintains constant $O(1)$ complexity with 0.001–0.003 s latency regardless of network size, as read operations do not require consensus.

5.7.3. Scalability constraints and optimal network size

Table 5 indicates that a 12–15 validator configuration provides the optimal operational balance. Within this range, write latency remains below 4.7 s under maximum load (500 TPS), query latency is stable at 0.002–0.003 s regardless of network size, and throughput sustains near-target performance (203–500 TPS achieved against a 500 TPS target).

Beyond 15 validators, consensus overhead increases significantly due to the $O(N^2)$ communication complexity of QBFT, leading to performance degradation. For larger deployments involving more than 15 stakeholder organizations, hierarchical or sharded architectures would be required. Regional clusters of 12–15 validators operating semi-independently with periodic cross-chain settlement could reduce effective communication complexity toward $O(N \log N)$ and maintain scalability at national scale.

5.7.4. End-to-end system complexity

The overall system complexity is dominated by two independent pipelines:

- **Real-time alert path**: Edge inference ($O(HWC)$) + temporal confirmation ($O(M)$) + audio trigger ($O(1)$) = $O(HWC)$ per frame
- **Asynchronous logging path**: Smart contract invocation via Web3.py ($O(1)$) + QBFT consensus ($O(N^2)$) = $O(N^2)$ per confirmed event

Critically, these pipelines operate independently; the driver receives immediate warning based on edge inference, while blockchain logging occurs asynchronously without blocking subsequent detections. This architectural separation ensures that blockchain latency does not compromise real-time safety functionality.

Table 12
Scalability analysis summary.

Component	Complexity	Practical limit
Edge inference (YOLOv11s)	$O(HWC)$	15.31 FPS (Jetson Nano)
Temporal confirmation	$O(M)$	5-second window
QBFT consensus (Write)	$O(N^2)$	12–15 nodes optimal
Smart contract query (Read)	$O(1)$	<3 ms (all network sizes)
Memory footprint	–	2.914 MB (48% less than EFFNet-CA)

5.7.5. Scalability summary

Table 12 summarizes the scalability characteristics:

This analysis establishes that DRIVERDAPP is suitable for regional or city-scale logistics consortium deployments with 12–15 participating organizations. National or continental-scale deployments would require architectural extensions such as hierarchical federation or layer-2 scaling solutions, which represent important directions for future work.

6. Conclusion

This work presents DRIVERDAPP, a foundational driver distraction monitoring framework for controlled logistics environments that integrates real-time edge intelligence, Explainable Artificial Intelligence (XAI), and permissioned blockchain-based accountability. XAI is enabled through Grad-CAM, providing transparent visual interpretation of model decisions in safety-critical scenarios. Secure and auditable event logging is demonstrated using QBFT consensus, where blockchain evaluation demonstrates near real-time query latency at the millisecond level and stable operation under varying validator sizes and transaction loads in controlled testing environments. By jointly addressing explainability, real-time responsiveness, and trustworthy event management, the proposed framework provides a practical foundation for secure, explainable, and auditable driver monitoring in intelligent transportation systems.

While DRIVERDAPP demonstrates strong performance under experimental conditions, several limitations warrant future investigation. First, the model's performance in diverse weather and lighting conditions (e.g., night driving, fog, heavy rain) requires further validation, as the current evaluation utilized datasets captured under controlled in-cabin illumination. Second, blockchain scalability analysis reveals that beyond 15 validator nodes, QBFT consensus introduces significant latency trade-offs exceeding 10 s at 500 TPS with 19 nodes, due to its $O(N^2)$ communication complexity. For city-scale or national deployments, hierarchical blockchain architectures or alternative consensus mechanisms would be required. Third, the system has not been validated in actual moving vehicles under real-world conditions, including vibration, variable cellular network connectivity, and handoff scenarios between base stations.

Future work will address these limitations through several key directions. First, field trials in operational logistics fleets will evaluate performance under diverse environmental conditions and real driving scenarios. Second, integration of the Interplanetary File System (IPFS) will enable decentralized storage of Grad-CAM explainability visualizations, further enhancing transparency and auditability. Third, simulator-based testing with synthetic adverse weather data will assess robustness across a broader range of conditions. Fourth, exploration of hybrid consensus mechanisms or blockchain sharding approaches will improve scalability while maintaining the deterministic finality required for legal compliance. Real-world deployments with diverse stakeholder participation, including insurance providers, logistics companies, and regulatory agencies, remain the critical next step for validating the framework's practical utility in intelligent transportation systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the IITP grant funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201612, 20%) and by Priority Research Centers Program through the NRF funded by the MEST (2018R1A6A1A03024003, 20%) and by the MSIT, Korea, under the ITRC support program (IITP-2026-RS-2024-00438430, 20%) and by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(RS-2025-25431637, 20%) and the regional innovation system & education (RISE)-(Idea Start-up Valley) program through the Gyeongbuk RISE Center, funded by the Ministry of Education (MOE) and the Gyeongsangbuk-do, Republic of Korea(2026-rise-15-105, 20%).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.compeleceng.2026.111340.

Data availability

The State Farm Distracted Driver Detection dataset is publicly available at https://www.kaggle.com/c/state-farm-distracted-driver-detection. The AUC Distracted Driver (V2) dataset is available from [30,31]. The trained YOLOv11s model weights, smart contract source code, and blockchain deployment scripts will be made publicly available upon acceptance at https://github.com/rasyidred/driver_dapp.git to facilitate reproducibility. The Flutter DApp source code contains stakeholder-specific configurations and will be available upon request for research purposes.

References

[1] Khan T, Choi G, Lee S. EFFNet-CA: An efficient driver distraction detection based on multiscale features extractions and channel attention mechanism. Sensors 2023;23(8).
[2] Baheti B, Gajre S, Talbar S. Detection of distracted driver using convolutional neural network. In: 2018 IEEE/CVF conference on computer vision and pattern recognition workshops. CVPRW, 2018, p. 1145–11456.
[3] Adochiei I-R, Ştirbu O-I, Adochiei NI, Pericle-Gabriel M, Larco C-M, Mustata S-M, Costin D. Drivers' drowsiness detection and warning systems for critical infrastructures. In: 2020 international conference on e-health and bioengineering. EHB, 2020, p. 1–4.
[4] Li P, Yang Y, Grosu R, Wang G, Li R, Wu Y, Huang Z. Driver distraction detection using octave-like convolutional neural network. IEEE Trans Intell Transp Syst 2022;23(7):8823–33.
[5] Schwartz N, Buliung R, Daniel A, Rothman L. Disability and pedestrian road traffic injury: A scoping review. Health Place 2022;77:102896.
[6] Ajakwe SO, Kim D-S. Facets of security and safety problems and paradigms for smart aerial mobility and intelligent logistics. IET Intell Transp Syst 2024;18:2827–55.
[7] Ajakwe SO, Igboanusi IS, Lee J-M, Kim D-S. i BANDA: A blockchain-assisted defense system for authentication in drone-based logistics. Drones 2025;9(8):590.
[8] Nandhini P, Kuppuswami S, Malliga S, Srinath P, Veeramanikandan P. Driver drowsiness detection using deep learning. In: 2022 6th international conference on computing methodologies and communication. ICCMC, 2022, p. 1031–6.
[9] Sinha A, Aneesh RP, Gopal SK. Drowsiness detection system using deep learning. In: 2021 seventh international conference on bio signals, images, and instrumentation. ICBSII, 2021, p. 1–6.
[10] Mansur V, Shambavi K. Highway drivers drowsiness detection system model with R-Pi and CNN technique. In: 2021 12th international conference on computing communication and networking technologies. ICCCNT, 2021, p. 1–6.
[11] Li Y, Wang L, Mi W, Xu H, Hu J, Li H. Distracted driving detection by combining ViT and CNN. In: 2022 IEEE 25th international conference on computer supported cooperative work in design. CSCWD, 2022, p. 908–13.
[12] Ganguly B, Dey D, Munshi S. An integrated system for drivers' drowsiness detection using deep learning frameworks. In: 2022 IEEE VLSI device circuit and system. VLSI DCS, 2022, p. 55–9.
[13] I SS, Ramli R, Azri MA, Aliff M, Mohammad Z. Raspberry pi based driver drowsiness detection system using convolutional neural network (CNN). In: 2022 IEEE 18th international colloquium on signal processing & applications. CSPA, 2022, p. 30–4.
[14] Majdi MS, Ram S, Gill JT, Rodriguez JJ. Drive-net: Convolutional network for driver distraction detection. In: 2018 IEEE southwest symposium on image analysis and interpretation. SSIAI, 2018, p. 1–4.
[15] Masood S, Rai A, Aggarwal A, Doja M, Ahmad M. Detecting distraction of drivers using Convolutional Neural Network. Pattern Recognit Lett 2020;139:79–85.
[16] Jamsheed V. A, Janet B, Reddy US. Real Time Detection of driver distraction using CNN. In: 2020 third international conference on smart systems and inventive technology. ICSSIT, 2020, p. 185–91.
[17] Srinivasan K, Garg L, Datta D, Alaboudi A, Jhanjhi N, Agarwal R, Thomas A. Performance comparison of deep CNN models for detecting driver's distraction. Comput Mater Contin 2021;68:4109–24. http://dx.doi.org/10.32604/cmc.2021.016736.
[18] Hossain MU, Rahman MA, Islam MM, Akhter A, Uddin MA, Paul BK. Automatic driver distraction detection using deep convolutional neural networks. Intell Syst Appl 2022;14:200075.
[19] Bekka R, Kherbouche S, El B. Distraction detection to predict vehicle crashes: A deep learning approach. Comput Sist 2022;26(1). http://dx.doi.org/10.13053/CyS-26-1-3871.
[20] Khan T, Choi G, Lee S. EFFNet-CA: an efficient driver distraction detection based on multiscale features extractions and channel attention mechanism. Sensors 2023;23(8):3835.
[21] Lei J, Ni Z, Peng Z, Hu H, Hong J, Fang X, Yi C, Ren C, Wasaye MA. An intelligent network framework for driver distraction monitoring based on RES-SE-CNN. Sci Rep 2025;15(1):6916.
[22] Alemdar KD, Çodur MY. A new approach to detect driver distraction to ensure traffic safety and prevent traffic accidents: Image processing and MCDM. Sustainability 2024;16(17). http://dx.doi.org/10.3390/su16177642, URL https://www.mdpi.com/2071-1050/16/17/7642.
[23] Tkachuk R-V, Ilie D, Robert R, Kebande V, Tutschku K. On the performance and scalability of consensus mechanisms in privacy-enabled decentralized renewable energy marketplace. Ann Telecommun 2024;79(3):271–88.
[24] Delladetsimas AP, Papangelou S, Iosif E, Giaglis G. Leadership uniformity in timeout-based quorum Byzantine fault tolerance (QBFT) consensus. Big Data Cogn Comput 2025;9(8):196.
[25] Ferone A, Verrilli S. Exploiting blockchain technology for enhancing digital twins' security and transparency. Future Internet 2025;17(1):31.
[26] Montoya A, Holman D, Smith T, Kan W. State farm distracted driver detection. 2016.
[27] Jocher G, Qiu J. Ultralytics YOLO11. 2024, URL https://github.com/ultralytics/ultralytics.
[28] McCubbin G. Intro to web3.py: Ethereum for Python developers. Dapp University; 2025, https://www.dappuniversity.com/articles/web3-py-intro.
[29] Singh D. Using convolutional neural networks to perform classification on state farm insurance driver images. Technical report, 650 Serra Mall, CA: Stanford University; 2016.
[30] Eraqi HM, Abouelnaga Y, Saad MH, Moustafa MN. Driver distraction identification with an ensemble of convolutional neural networks. J Adv Transp 2019;2019(1):4125865.
[31] Abouelnaga Y, Eraqi HM, Moustafa MN. Real-time distracted driver posture classification. 2017, arXiv preprint arXiv:1706.09498.
[32] Delladetsimas AP, Papangelou S, Iosif E, Giaglis G. Leadership uniformity in timeout-based quorum Byzantine fault tolerance (QBFT) consensus. Big Data Cogn Comput 2025;9(8). http://dx.doi.org/10.3390/bdcc9080196, URL https://www.mdpi.com/2504-2289/9/8/196.
[33] Ferone A, Verrilli S. Exploiting blockchain technology for enhancing digital twins' security and transparency. Future Internet 2025;17(1). http://dx.doi.org/10.3390/fi17010031, URL https://www.mdpi.com/1999-5903/17/1/31.
[34] Olson RL, Hanowski RJ, Hickman JS, Dingus TA. Driver distraction in commercial vehicle operations. Tech. Rep. FMCSA-RRR-09-042, Washington, D.C.: Federal Motor Carrier Safety Administration; 2009, URL https://www.fmcsa.dot.gov/sites/fmcsa.dot.gov/files/docs/DriverDistractionStudy.pdf.
[35] Hickman JS, Hanowski RJ. An assessment of commercial motor vehicle driver distraction using naturalistic driving data. Traffic Inj Prev 2012;13(6):612–9. http://dx.doi.org/10.1080/15389588.2012.683841.