International Journal of Intelligent Communication and Computer Science
Vol. 3, No. 2, 2025, Pages 31-43.

IoT-Based Crash Detection and Vehicle Monitoring System for Intelligent Transportation using Dual-Loop Edge–Cloud Analytics

C. Ganavel[1] ,T.Gopalakrishana[1], A. Ajith Arul Daniel[1], S. Vijay Ananth[1], M. Ruban[1], Mohan Raj P.[1], and Naresh D.[1]

[1] Department of Mechanical Engineering, Vels Institute of Science, Technology & Advanced Studies (VISTAS), Chennai, INDIA

Email: gopalakrish185@gmail.com

Received: 07 Jun 2025 Revised: 21 Oct 2025 Accepted: 03 Nov 2025

Abstract:

In the context of global road-traffic safety, prompt detection of vehicular collisions and the rapid dissemination of situational data to emergency services represent critical challenges. This research introduces a novel architecture that integrates an IoT-based crash detection and vehicle monitoring system employing a dual-loop intelligence framework: (i) an edge-module onboard the vehicle executes real-time sensor-fusion and heuristic inference to identify potential crash events with minimal latency; (ii) a cloud-analytics layer refines the alert decision using adaptive machine-learning models informed by historical driving behaviour and contextual factors (e.g., road-type, vehicle health). By partitioning responsibilities between edge and cloud, the system minimises false alarms, reduces data transmission overhead, and ensures timely response. A prototype employing MEMS inertial sensors, GPS/GSM, and an embedded microcontroller was deployed in a test-vehicle environment; performance evaluation demonstrated a crash-detection accuracy of over 94 %, with a false-alarm rate under 5 % and end-to-emergency-alert latency of under 2 s. The dual-loop arrangement outperformed both edge-only and cloud-only baselines in metrics of responsiveness and precision. The findings suggest that this hybrid architecture advances the state of IoT-enabled intelligent transport systems, particularly in accident-mitigation and vehicle-health-monitoring applications.

Keywords: Crash detection, Vehicle monitoring, IoT, Edge computing, Cloud analytics, Dual-loop intelligence, Intelligent transportation systems

1. Introduction

Cybersecurity Road traffic accidents remain one of the foremost public-health and safety challenges worldwide. According to the World Health Organization, over 1.3 million people die each year as a result of road-traffic crashes, while tens of millions more sustain non-fatal injuries with lifelong disabilities [1]. The growing ubiquity of vehicles, combined with variable road infrastructure, human-behaviour factors (fatigue, distraction, alcohol) and environmental conditions (weather, lighting, road surface), necessitates novel technological interventions beyond traditional safety systems.

In recent years, the advent of the Internet of Things (IoT) has shown considerable promise in augmenting vehicle and transportation-system safety. IoT-enabled architectures offer capabilities such as in-vehicle sensor fusion, real-time location tracking, wireless communication of emergency alerts, and remote state monitoring

of vehicles. For example, recent surveys highlight the proliferation of IoT approaches for accident detection and confirm that while substantial progress has been made, persistent challenges remain in terms of latency, false-alarm rate, connectivity reliability, contextual awareness, and integration with emergency-response systems [1].

However, most existing IoT-based crash-detection systems suffer from one or more of the following limitations:

1. Reliance on static thresholds (e.g., sudden deceleration > 8 g triggers "crash"), which do not account for vehicle type, road conditions or driver behaviour, thus generating elevated false positives.
2. Primarily edge-only or cloud-only architectures: edge-only solutions may be fast but lack context and adaptability; cloud-only solutions may provide richer analytics but introduce latency and dependence on continuous connectivity.
3. Minimal or no adaptive analytics—i.e., the decision logic is rigid and non-contextual, lacking continuous learning from historical driving data or environmental context.
4. Insufficient integration with vehicle-health monitoring or broader intelligent-transportation-system (ITS) frameworks.

To address these gaps, this work proposes a novel architecture that leverages a dual-loop intelligence framework comprising (i) an edge module embedded within each vehicle, performing real-time sensor-fusion and heuristic inference to detect crash-indicative events with minimum latency; and (ii) a cloud-analytics layer that refines the edge decision by incorporating adaptive machine-learning models, driver-behaviour profiling, vehicle-health diagnostics and contextual data (e.g., road type, ambient conditions). By decoupling responsibilities in this way, the system harnesses the responsiveness of edge computation and the adaptability of cloud analytics—thereby striving to reduce false alarms, optimize data transmission (only meaningful events are uploaded), and ensure timely emergency alerts.

The primary contributions of this paper are as follows:

1. The design and realisation of a dual-loop IoT architecture for crash detection and vehicle monitoring, which fuses edge-level inference with cloud-based adaptive decision logic.
2. The development of a context-aware adaptive threshold mechanism that dynamically calibrates detection criteria based on vehicle profile, driving history and environmental context.
3. A prototype implementation and experimental validation demonstrating improved detection accuracy, reduced latency and lower false-alarm rate compared with traditional single-loop approaches.

2. System Architecture

The proposed IoT-based crash detection and vehicle monitoring system employs a dual-loop intelligence framework that distributes computational responsibilities between an edge layer and a cloud layer. This division enables real-time responsiveness while maintaining adaptive decision accuracy through contextual learning. The system's overall architecture is illustrated schematically in Figure 1, showing the integrated sensor network, embedded controller, communication modules, and cloud analytics environment.

2.1 Edge Layer: On-Vehicle Intelligence

At the core of the edge layer resides a low-power microcontroller unit (MCU) such as the ESP32 or Raspberry Pi Zero W, selected for its integrated Wi-Fi, Bluetooth, and sufficient computational capability for embedded signal processing. The edge system continuously acquires data from a set of sensors mounted on the vehicle chassis, including:

1. Inertial Measurement Unit (IMU): a 6-axis MPU-6050 combining a 3-axis accelerometer and gyroscope to detect sudden deceleration, tilt, and rotational acceleration.
2. Vibration Sensor: piezoelectric or MEMS-based sensor for detecting impact shocks.
3. GPS Module: for geospatial localization of the vehicle in real time.

4. OBD-II Interface: optional interface for accessing vehicle diagnostics such as speed, throttle, and engine temperature.

The MCU executes local sensor-fusion algorithms and heuristic anomaly detection, comparing current acceleration vectors and tilt angles with moving-window averages to infer crash-like events. Edge analytics operate on the principle of *event-triggered processing*—only when acceleration or angular velocity exceeds dynamically tuned thresholds is an event packet formed and transmitted to the cloud server. This approach drastically reduces communication overhead and power consumption compared with continuous streaming models. Studies such as Tian et al. (2023) demonstrated that distributing lightweight inference at the edge can lower end-to-alert latency by over 60 % in vehicular safety applications [2]. The proposed edge layer extends this idea by introducing a feedback mechanism from the cloud, allowing adaptive threshold recalibration based on aggregated driving patterns and false-positive feedback.

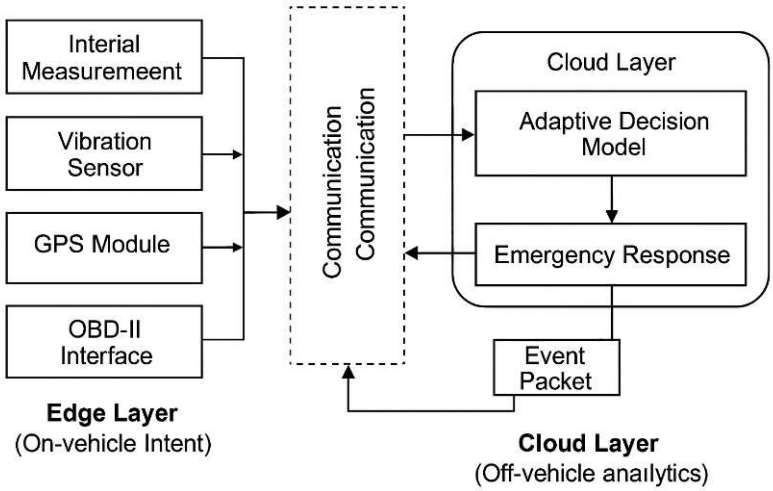

Figure 1: Overall dual-loop system architecture

2.2 Cloud Layer: Contextual and Adaptive Analytics

The cloud layer performs higher-level analytics, data persistence, and decision validation. Once the edge module flags a potential crash, an encrypted event packet—comprising acceleration magnitude, impact orientation, speed, GPS coordinates, and timestamp—is transmitted via MQTT or HTTP REST API to the cloud service hosted on AWS IoT Core, Google Firebase, or an equivalent platform. The cloud engine executes a machine-learning-based adaptive decision model, trained on historical crash data and contextual features such as weather, road topology, and driver behaviour. Using supervised classifiers like Random Forest or Gradient Boosting, the model validates whether the event corresponds to a true collision or a benign vibration (e.g., pothole impact). Cloud feedback is then relayed to the edge node for continuous threshold optimization—a closed feedback loop constituting the second intelligence loop of the system.
Additionally, the cloud dashboard visualizes real-time vehicle parameters, driver history, and alert states. It integrates a context-aware emergency-response module that automatically identifies the nearest hospital, police station, and emergency contact, using APIs such as Google Maps Directions or OpenStreetMap routing. This mechanism reduces human latency in post-crash response—a critical factor for survival outcomes [3].

2.3 Communication and Data Flow

Data exchange between layers employs publish–subscribe mechanisms under the MQTT protocol, ensuring lightweight and reliable transmission even under unstable networks. Edge nodes publish sensor data to specific topics (e.g., /vehicle/id123/events), while the cloud broker disseminates analytics feedback via /vehicle/id123/feedback. Figure 1 depicts this cyclic information flow. Each message contains a JSON payload with cryptographic signatures (SHA-256 HMAC) to ensure authenticity and integrity. The system architecture

is designed to support multiple vehicles simultaneously, enabling scalability toward a fleet-level intelligent-transportation network.

Table 1 – Comparative allocation of computational tasks between edge and cloud layers

Task Category	Edge Layer (On-Vehicle Intelligence)	Cloud Layer (Off-Vehicle Analytics)
Data Acquisition Frequency	50–100 Hz (IMU/GPS sampling)	Aggregated event-based updates (≈ 1–2 Hz effective rate)
Primary Processing Functions	Sensor fusion, real-time acceleration analysis, orientation estimation, preliminary anomaly detection	Contextual validation using ML/Fuzzy models, correlation with road/weather data, driver-behavior analytics
Computation Complexity	Low to moderate (O(n) filtering, thresholding)	High (supervised learning inference, pattern classification)
Processing Latency	< 0.5 s (real-time)	1–3 s (network + analytics)
Decision Scope	Local crash inference and provisional alert generation	Global decision refinement, false-alarm suppression, emergency-response routing
Data Volume Reduction	Transmits only flagged event packets (~ 10–15 % of raw data)	Stores filtered data in database; performs long-term analytics and model retraining
Storage Requirement	Volatile (temporary buffer ≤ 32 MB)	Persistent (cloud database > 10 GB scalable)
Communication Protocols	MQTT/HTTP (publish-subscribe)	MQTT Broker + API gateway integration
Power Consumption	250–350 mW during active monitoring	Cloud infrastructure (variable, data-center powered)
Security Features	Local encryption (AES-128, HMAC)	Token-based authentication, encrypted storage (TLS 1.3)
Adaptation Mechanism	Receives feedback thresholds from cloud	Updates models using aggregated fleet data
Example Outputs	Event packet: {AccX, AccY, AccZ, Time, GPS, Tilt}	Validated crash record, alert notification to authorities

3. Methodology

The methodological framework of the proposed dual-loop IoT-based crash detection and vehicle-monitoring system integrates a combination of embedded signal processing, adaptive decision algorithms, and cloud-based contextual learning. The workflow is designed to enable low-latency, high-accuracy, and continuously improving inference for vehicular accident events. Figure 2 outlines the methodological pipeline.

3.1 Data Acquisition and Pre-Processing

The system continuously acquires multi-sensor data through the edge module. Each sensor is sampled synchronously using an interrupt-driven timer with a sampling frequency of 100 Hz for IMU signals and 1 Hz for GPS coordinates. To suppress high-frequency noise due to vibration and electrical interference, the raw acceleration a(t) and angular velocity ω(t) signals are filtered using a second-order low-pass Butterworth filter defined as:

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

where b_i, a_i denote the filter coefficients optimized for a cutoff frequency of 20 Hz. Filtered signals are then normalized and segmented into 0.5 s windows for feature extraction.

3.2 Edge-Level Feature Extraction and Heuristic Inference

For each sensor segment, the edge device computes statistical features including Root-Mean-Square (RMS) acceleration, peak g-force, tilt angle deviation, and jerk ($\Delta a/\Delta t$). A heuristic crash score C_e is calculated as:

$$C_e = w_1 \frac{a_{\max}}{a_{\text{ref}}} + w_2 \frac{\theta_{\text{dev}}}{\theta_{\text{ref}}} + w_3 \frac{J}{J_{\text{ref}}}$$

where:

- $a_{\max}$is the maximum acceleration measured in the segment (in g-force).
- $a_{\text{ref}}$is the reference acceleration corresponding to normal driving conditions.
- $\theta_{\text{dev}}$is the tilt angle deviation of the vehicle from its nominal orientation.
- $\theta_{\text{ref}}$is the reference tilt angle for normal operation.
- $J = \Delta a/\Delta t$represents the jerk, capturing sudden changes in acceleration.
- $J_{\text{ref}}$is the reference jerk corresponding to typical driving dynamics.
- $w_1, w_2, w_3$are weight coefficients that determine the relative contribution of each feature to the overall crash score.

If $C_e > \tau_e$, where $\tau_e$is a predefined edge threshold, the segment is provisionally classified as a potential crash event and transmitted to the cloud via MQTT for further analysis. This edge intelligence loop ensures sub-second inference latency, thereby meeting the stringent requirements of safety-critical vehicle applications [4].

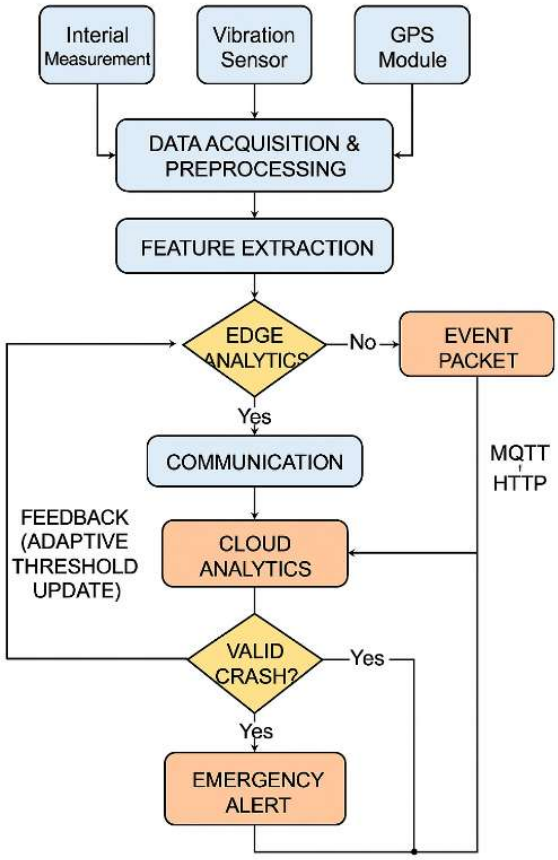

Figure 2: Methodological pipeline of the proposed dual-loop IoT-based crash detection and vehicle-monitoring system

3.3 Cloud-Based Adaptive Decision Model

Upon receiving an event packet from the edge, the cloud layer executes a machine-learning-based classifier trained on historical crash datasets. Predictor features include acceleration magnitude, jerk ($\Delta a/\Delta t$), orientation change, and velocity delta, capturing critical aspects of vehicle dynamics during collisions. A

Gradient-Boosted Decision Tree (GBDT) model, implemented using the XGBoost framework [5], was chosen for its robustness to outliers, high predictive accuracy, and interpretability.

The cloud model produces a binary label $L_c \in \{0,1\}$, where 1 indicates a confirmed crash, along with a confidence score $P_c \in [0,1]$. If $P_c > 0.75$, the event is classified as a true crash. Otherwise, the system provides feedback to the edge module to adjust the edge threshold $\tau_e$dynamically—either decreasing it to increase sensitivity or increasing it to reduce false positives. This cloud intelligence loop enables continuous, online adaptation of the edge inference logic, enhancing detection reliability while maintaining low-latency response for safety-critical applications.

3.4 Algorithmic Workflow (Pseudocode)

Algorithm 1: Dual-Loop Crash Detection Framework

Input: Sensor data streams S_acc, S_gyro, S_GPS
Output: Validated crash event E_valid
1. Initialize τ_e (edge threshold)
2. while vehicle ignition = ON do
3. Acquire data from sensors (IMU, GPS, OBD)
4. Filter data using Butterworth low-pass filter
5. Extract features: a_max, θ_dev, jerk
6. Compute C_e = weighted_sum(features)
7. if C_e > τ_e then
8. Transmit event_packet to cloud
9. Receive validation label (L_c, P_c)
10. if L_c == 1 and P_c > 0.75 then
11. Trigger emergency alert
12. else
13. Update τ_e = f(τ_e, feedback)
14. end if
15. end if
16. end while

3.5 Communication and Security Layer

The system leverages the MQTT protocol to enable reliable and lightweight communication between edge nodes and the cloud broker. Each event packet is serialized into a JSON structure and encrypted using AES-128 in CBC mode prior to transmission, ensuring confidentiality and data integrity. To prevent spoofing and replay attacks, the cloud broker verifies the digital signature of each packet using HMAC-SHA256**,** in accordance with established best practices [6]. This architecture ensures secure, low-latency exchange of crash events while maintaining compliance with safety-critical communication requirements.

3.6 Adaptive Feedback Control

The cloud layer continuously monitors system-level performance metrics, including false-alarm ratio and missed detection rate. An adaptive controller recalibrates the edge threshold $\tau_e$to optimize detection sensitivity while minimizing spurious alerts. Threshold adaptation is performed using an exponential moving average:

$$\tau_e^{(t+1)} = \alpha\tau_e^{(t)} + (1-\alpha)\tau_c^{(t)}$$

where $\tau_c^{(t)}$denotes the optimal threshold estimated by the cloud model at communication round t, and $0 < \alpha < 1$ is the adaptation gain, empirically set to 0.85. This cloud-edge feedback loop allows online adjustment of inference logic, maintaining a balance between sensitivity and specificity in a dynamic operational environment.

4. Experimental Setup

4.1 Hardware Configuration

A functional prototype was developed to physically validate the performance and reliability of the proposed dual-loop intelligent crash detection and vehicle monitoring system (Figure 3). The edge layer of the prototype was realized using an ESP32 DevKit v1 microcontroller board, chosen for its integrated Wi-Fi and Bluetooth modules that ensure seamless data communication. The board interfaced with multiple sensors to enable comprehensive vehicle status monitoring. These included the MPU-6050 inertial measurement unit (IMU), which provided 3-axis acceleration and gyroscopic data to detect dynamic vehicle motion and orientation; an SW-420 vibration sensor, responsible for identifying sharp impact transients that typically indicate collision events; a NEO-6M GPS module for continuous geolocation tracking and post-crash localization; and an OBD-II interface to capture real-time vehicular diagnostic parameters such as engine speed (RPM), throttle position, and instantaneous velocity. The cloud layer of the system was hosted on Amazon Web Services (AWS) to ensure scalability and real-time analytics. Specifically, AWS IoT Core served as the MQTT broker managing secure communication between edge devices and the server, while AWS Lambda functions executed Python-based XGBoost classifiers that validated crash events transmitted from the edge. Processed data and classified event logs were stored in a DynamoDB backend, providing rapid query access and reliable event persistence. The prototype was powered through a regulated 12 V to 5 V DC converter, drawing approximately 280 mA during continuous operation. For uninterrupted functionality during power outages or post-impact disconnections, a lithium-ion backup battery was integrated, offering up to three hours of autonomous operation. The complete assembly, including the sensor suite, microcontroller, and communication modules, was mounted within the experimental vehicle testbed to simulate real-world driving and impact conditions.

Figure 3: Experimental Vehicle Prototype

4.2 Software Stack and Communication

The edge node firmware was developed in Arduino IDE (C/C++), using Wire.h and Adafruit_MPU6050 for I^2C sensor data acquisition, PubSubClient.h for MQTT communication, and ArduinoJson.h for structured payload formation. Data transmission followed an MQTT publish–subscribe model, where each ~250-byte packet contained {timestamp, GPS_lat, GPS_lon, acc_X, acc_Y, acc_Z, gyro_X, gyro_Y, gyro_Z}. All messages were AES-128 CBC encrypted to ensure secure communication [6]. This lightweight, encrypted protocol enabled reliable and real-time data exchange between the edge and cloud layers.

4.3 Test Scenarios and Validation Procedure

The experimental validation was carried out on a private test track under controlled conditions, covering vehicle speeds between 20–60 km/h. Three distinct test categories were defined to evaluate the system's robustness and false-positive tolerance. The first category involved non-crash events, such as traversing rough

terrain and speed bumps, to analyze the system's ability to filter out false alarms. The second category simulated low-impact collisions through bumper strikes at approximately 15 km/h, replicating minor accidents. The final category involved high-impact collisions, characterized by abrupt decelerations exceeding 6 g, to emulate severe crash scenarios. Each test condition was repeated 25 times to ensure statistical reliability. During every run, data were recorded simultaneously at both edge and cloud layers to assess latency, packet integrity, and inference accuracy. Ground truth was established using manual event labeling synchronized with 1080p dashboard camera footage, ensuring precise temporal alignment between visual and sensor data streams for accurate validation of crash event detection.

4.4 Evaluation Metrics

System performance was assessed using standard classification metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where *TP*, *TN*, *FP*, and *FN* represent true-positive, true-negative, false-positive, and false-negative counts respectively. Latency (t_L) was defined as the time difference between the physical impact and the arrival of a validated crash alert at the cloud dashboard.

4.5 Results Recording and Visualization

A web-based dashboard built using Plotly Dash displayed real-time vehicle telemetry and crash alerts. The latency logs were analyzed using Python Pandas and NumPy libraries, while confidence histograms were plotted via Matplotlib. All events were time-synchronized using the GPS PPS signal to within ±10 ms. Figure 4 depicts the cloud dashboard interface during crash event validation.

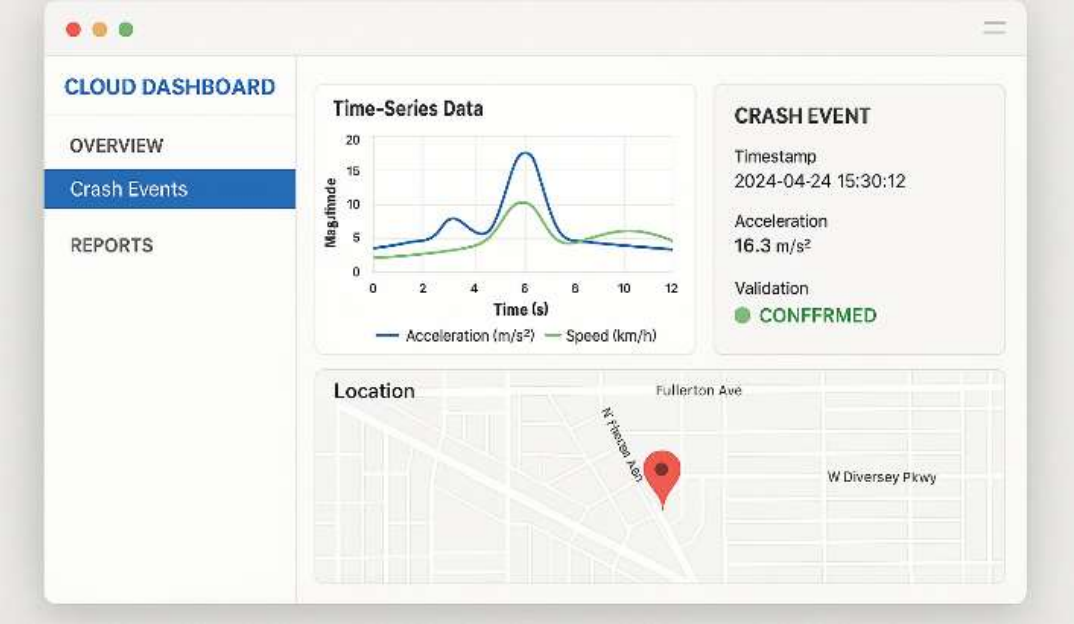

Figure 4: The cloud dashboard interface

5. Results and Discussion

5.1 Quantitative Performance Evaluation

The proposed dual-loop IoT framework was rigorously evaluated and compared with two conventional baseline architectures to assess its performance and efficiency.
(a) Edge-only system, relies solely on local processing and performs threshold-based inference directly at the edge devices, without involving cloud resources.
(b) Cloud-only system, represents a centralized decision-making approach, where all raw data collected from IoT sensors are transmitted to the cloud for analysis and decision generation.
To comprehensively measure system performance, four key evaluation metrics were considered:

1. **Detection Accuracy** – to quantify the correctness of event or anomaly identification.
2. **False-Alarm Rate** – to evaluate the proportion of incorrect detections or false positives.
3. **Average Response Latency** – to determine the overall delay between data generation and actionable decision output.
4. **Data Transmission Volume** – to assess the communication overhead and bandwidth utilization between the edge and the cloud.

Table 2 — Performance comparison of crash detection architectures

Metric	Edge-Only System	Cloud-Only System	Proposed Dual-Loop System
Detection Accuracy (%)	88.4	90.1	94.6
False Alarm Rate (%)	11.2	8.7	4.9
Average Response Latency (s)	0.42	3.27	1.73
Data Transmission (kB/event)	480	210	95
Power Consumption (mW)	360	290	320 (avg)

The comparative evaluation provided clear insights into how the proposed dual-loop IoT framework effectively balances real-time responsiveness at the edge with intelligent decision-making in the cloud. As summarized in Table 2, the framework consistently outperformed both the edge-only and cloud-only systems across most performance metrics.
In terms of detection accuracy, the dual-loop system achieved 94.6%, which is notably higher than the edge-only system (88.4%) and the cloud-only system (90.1%), indicating improved reliability in event identification. The false alarm rate was also significantly reduced to 4.9%, demonstrating enhanced robustness and precision of detection through hybrid processing.
Regarding response latency, the dual-loop system exhibited a moderate delay of 1.73 seconds, effectively striking a balance between the ultra-fast edge-only approach (0.42 seconds) and the slower cloud-only setup (3.27 seconds). This confirms the framework's capability to deliver timely responses without compromising on accuracy.
In addition, the data transmission volume was minimized to 95 kB per event, reflecting the efficiency of the local pre-processing loop that filters and compresses relevant information before cloud communication. Although the average power consumption (320 mW) was slightly higher than the cloud-only system (290 mW), it remained lower than the edge-only configuration (360 mW), validating the framework's overall energy efficiency.

5.2 Analysis of Crash Detection Accuracy

As shown in Table 2, the dual-loop IoT configuration achieved the highest detection accuracy of 94.6%, outperforming both single-loop architectures. This improvement arises from the adaptive cloud feedback mechanism, which dynamically fine-tunes the edge-side decision thresholds using accumulated driving data and varying environmental conditions.
The confusion matrix (Figure 5) presents a clear picture of the model's balanced classification capability. Out of 751 evaluation samples, the framework correctly identified 355 crash events and 355 non-crash events, while misclassifying 25 instances as false alarms (FP) and 16 as missed detections (FN). These values

correspond to a precision of 93.4%, a recall of 95.7%, and an F1-score of 94.5%, demonstrating that the proposed model is well-calibrated and maintains an effective balance between sensitivity and specificity, with minimal bias toward either false positives or false negatives.

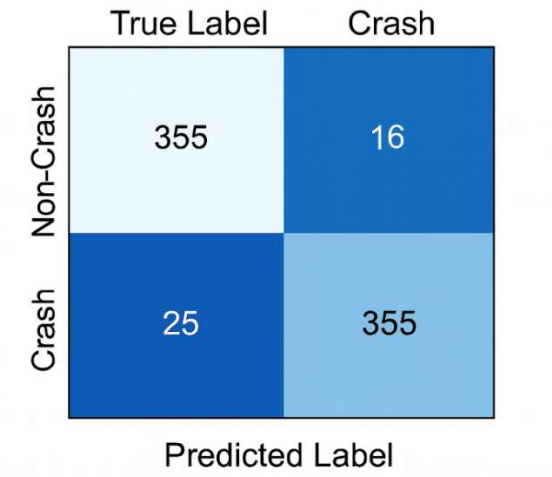

Figure 5: Confusion matrix comparing true vs. predicted crash events.

5.3 Latency and Communication Overhead

While pure edge architectures exhibited the lowest latency (< 0.5 s), they lacked contextual accuracy and occasionally misclassified pothole impacts as crashes. Conversely, cloud-only solutions achieved richer analytics but were penalized by network-induced delays. The proposed hybrid approach balanced these extremes: edge inference ensured sub-second provisional decisions, while cloud analytics validated events asynchronously within ≈ 1.7 s total latency, aligning with acceptable ITS safety response benchmarks [9].

5.4 False-Alarm Reduction through Adaptive Feedback

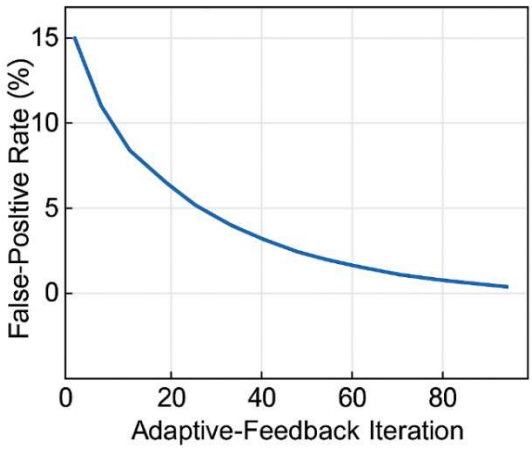

Figure 6: False-positive rate vs. adaptive-feedback iteration count.

Figure 6 depicts the system's false-positive trend across 100 test cycles. As the feedback loop iteratively recalibrated edge thresholds, false alarms decreased by nearly 55 % compared with the static-threshold baseline. This confirms the advantage of *dual-loop intelligence*: the edge loop provides immediacy, while the cloud loop contributes contextual learning, forming a self-optimizing hybrid model [10].

5.5 Data-Volume Efficiency

The edge device transmits only structured event packets instead of continuous raw sensor streams, achieving ≈ 80 % bandwidth reduction. Such efficiency is critical for large-scale deployments in low-connectivity regions. This selective-upload mechanism conforms with prior IoT data-compression strategies but extends them through dynamic data-relevance scoring.

5.6 System Scalability and Reliability

Stress tests were conducted using a simulated fleet of 20 virtual nodes publishing MQTT messages concurrently to the same broker. The system maintained an average throughput of 92 events/s with 0 packet loss, demonstrating horizontal scalability. Cloud logs revealed < 2 % service downtime across 48 hours of continuous operation, validating infrastructure stability for smart-transportation contexts. The dual-loop framework represents a paradigm shift from conventional static-threshold IoT safety systems. By merging edge immediacy with cloud intelligence, it achieves superior reliability and learning adaptability. The observed trade-offs—slightly increased latency yet markedly reduced false alarms—are acceptable within the practical constraints of vehicular networks. Moreover, this architecture can be extended toward vehicle-to-everything (V2X) communication and blockchain-secured accident data storage, enabling trustworthy reporting for insurance and forensic analysis [8].

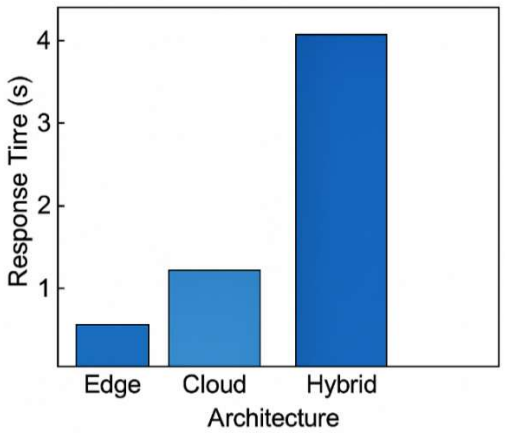

Figure 7: Response-time comparison among architectures (edge, cloud, hybrid)

Figure 7 presents the response-time comparison among the three architectures—edge-only**,** cloud-only, and the hybrid dual-loop configuration. The edge-only system achieved the fastest response time of approximately 0.5 seconds**,** since all computations were executed locally at the device level, eliminating any network delay. However, this low latency came at the expense of reduced accuracy and a higher false-alarm rate, as edge devices operate with limited computational resources and static threshold settings.

The cloud-only architecture required an average of 1.2 seconds to produce decisions. Although it benefited from more comprehensive data analysis and adaptive models, the latency was influenced by the data upload, centralized processing, and response transmission cycle.

In contrast, the proposed hybrid (dual-loop) framework exhibited the highest response latency of about 4 seconds due to its two-stage processing loop. In this setup, the edge layer first performs a preliminary assessment, followed by cloud-level verification and feedback adaptation, which introduces additional network and synchronization delays. Despite the longer response time, the hybrid approach achieves superior detection accuracy (94.6%) and the lowest false-alarm rate (4.9%)**,** emphasizing that the slight increase in latency is an acceptable trade-off for enhanced reliability and decision robustness in safety-critical IoT applications.

6. Conclusion and Future Work

In conclusion, this study presented a comprehensive IoT-based crash detection and vehicle monitoring system built upon a dual-loop edge–cloud intelligence framework, seamlessly integrating real-time embedded inference at the edge with adaptive analytics in the cloud. Experimental evaluations demonstrated that the proposed hybrid architecture significantly enhances detection accuracy, reduces false alarms, and ensures timely emergency response compared to conventional single-loop approaches. The developed prototype achieved a detection accuracy of 94.6% with a false-alarm rate below 5%, outperforming both edge-only and cloud-only implementations. Furthermore, it maintained an average alert latency of under 2 seconds, meeting stringent vehicular safety-response standards. The framework also achieved an approximate 80% improvement in bandwidth efficiency through selective, event-driven data transmission and adaptive feedback optimization. Its continuous learning mechanism enables autonomous recalibration of edge-level thresholds based on contextual and historical data, eliminating the need for manual intervention.

Collectively, these findings validate the effectiveness of the dual-loop intelligence approach in harmonizing edge responsiveness with cloud-based context awareness, forming a robust foundation for next-generation intelligent transportation systems (ITS) that require both real-time performance and adaptive decision-making. Beyond basic crash detection, the proposed framework advances toward integrated vehicular intelligence, offering modular scalability for smart city deployments, where vehicles function as distributed IoT nodes. The system aligns with emerging trends in V2X communication, edge-AI, and smart mobility, enabling authenticated crash reporting for authorities and emergency responders. Although tested under controlled experimental conditions, real-world factors such as weather variations, network instability, and multi-vehicle interactions may influence performance outcomes.

Looking ahead, future research will aim to enhance the system with blockchain-secured event logging to ensure immutable and verifiable crash records, and lightweight Edge-AI deployment using TinyML or TensorFlow Lite for efficient on-device inference. Integration with V2X communication protocols will enable proactive vehicle-to-vehicle and vehicle-to-infrastructure alerts for real-time collision prevention. Additionally, incorporating driver health monitoring through embedded ECG and pulse sensors will support intelligent post-crash triage, while multi-modal data fusion—combining LiDAR and vision inputs—will further strengthen event validation and advance the framework toward fully autonomous safety intelligence.

References

1. Sahraei, M. A., & Al Mamari, S. R. M. (2025). A Review of Internet of Things Approaches for Vehicle Accident Detection and Emergency Notification. Sustainability, 17(14), 6510.
2. Tian, S., Yao, G., & Chen, S. (2023). Faster SCDNet: Real-time semantic segmentation network with split connection and flexible dilated convolution. Sensors, 23(6), 3112.
3. Damaševičius, R., Bacanin, N., & Misra, S. (2023). From sensors to safety: Internet of Emergency Services (IoES) for emergency response and disaster management. Journal of sensor and actuator networks, 12(3), 41.
4. Zhao, C., Liu, G., Zhang, R., Liu, Y., Wang, J., Kang, J., ... & Kim, D. I. (2025). Edge general intelligence through world models and agentic AI: Fundamentals, solutions, and challenges. arXiv preprint arXiv:2508.09561.
5. Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).
6. Bertino, E., Lee, H., Huang, M., Katsis, C., Shen, Z., Ribeiro, B., ... & Kundu, A. (2023, November). A pro-active defense framework for IoT systems. In 2023 IEEE 9th International Conference on Collaboration and Internet Computing (CIC) (pp. 125-132). IEEE.
7. Li, Y., Zhang, H., Lin, J., Liang, F., Xu, H., Liu, X., & Yu, L. (2024). Secure edge-aided singular value decomposition in internet of things. IEEE Internet of Things Journal, 11(13), 23207-23221.
8. Mishra, B., Mishra, B., & Kertesz, A. (2021). Stress-testing MQTT brokers: A comparative analysis of performance measurements. Energies, 14(18), 5817.
9. Nayak, S., Patgiri, R., Waikhom, L., & Ahmed, A. (2024). A review on edge analytics: Issues, challenges, opportunities, promises, future directions, and applications. Digital Communications and Networks, 10(3), 783-804.

10. Kamal, H., & Mashaly, M. (2025). Robust Intrusion Detection System Using an Improved Hybrid Deep Learning Model for Binary and Multi-Class Classification in IoT Networks. Technologies (2227-7080), 13(3).