International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

# Intrusion Detection System in CAN-BUS Vehicle Networks Using Machine Learning

*Deekshith G. R.[1], S. R. Sheetal[2], Divya G. S.[3], Darshan B. M.[4], Aryan Anil[5], Balakrishna V[6]*
*[1,2,3,4,5,6]Department of Computer Science, AMC Engineering College, Bengaluru, India*
***Email:*** *deekshith.gr123@gmail.com[1], srsheeta@gmail.com[2], Divya.siddaraj@gmail.com[3], mahadeva3008@gmail.com[4], aryananil00007@gmail.com[5], balureddy2004@gmail.com[6]*

## Abstract
*As vehicles become more connected and software-rich, securing communications inside vehicles is increasingly important. The Controller Area Network (CAN-BUS) is simple and efficient, which is why it remains the dominant vehicle network. That simplicity comes at a cost: the protocol lacks built-in authentication or encryption, leaving it exposed to injection, flooding and impersonation attacks. Here we describe a practical, real-time Intrusion Detection System (IDS) for CAN-BUS that combines supervised machine learning with features useful for deployment. The core detector is a Random Forest trained on the OCS-Lab dataset; it distinguishes benign traffic from DoS, fuzzy and impersonation attacks and achieves a validation accuracy of 85.37%. Importantly, the IDS is more than a model: it includes a retraining dashboard, a timeline for investigation, live packet classification and automated email alerts so operators can react quickly and with context.*

***Keywords:*** *Automotive cybersecurity; CAN-BUS; Intrusion detection system; Machine learning; Real-time monitoring.*

## 1. Introduction
Modern vehicles host many electronic control units (ECUs) that coordinate critical functions such as engine control, brake, body electronics and driver assistance. CAN-BUS is the standard in-vehicle backbone because it provides deterministic arbitration and a degree of fault tolerance. That same simplicity, however, leaves out cryptographic protections. An attacker with local access — through OBD-II, a compromised telematics module, or an aftermarket device — can inject frames, flood the bus, or impersonate ECUs; any of these actions can degrade availability or, worse, safety.

- **Motivation:** Effective protections must run in real time on constrained hardware and give operators concise, actionable signals so they can triage incidents quickly.
- **Contributions:** This work presents (i) a modular ML-based IDS with operator-facing dashboards and automated alerts. (ii) an end-to-end capture-to-alert pipeline; (iii) an empirical evaluation using OCS-Lab data; and (iv) deployment guidance covering latency, observability and maintenance.

## 2. Background on CAN-BUS
### 2.1. Frame Structure and Arbitration
A standard CAN frame includes an identifier (11- or 29-bit), control fields (including DLC), up to eight data bytes, CRC, ACK and EOF. Arbitration uses wired-AND: nodes with lower identifier values win access to the bus. This scheme supports real-time priorities but can be abused — an attacker who repeatedly sends a high-priority identifier can effectively deny service.

### 2.2. Error Handling and Fault Confinement
CAN controllers track transmits and receives error counters; exceeding thresholds causes nodes to enter error-passive or bus-off modes and stop transmitting. Attackers can deliberately cause errors using malformed frames. Because low-level counters are not always exposed to applications, our IDS emphasizes timing and distributional statistics that a passive capture can reliably observe.

## 3. Related Work
The literature includes interval and frequency detectors [1], temporal deep models [2], and broader system-level analyses [4]. Recent surveys and

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

comparative analyses summarize AI- driven in-vehicle IDS approaches and provide a taxonomy of features and learning methods; this help place our work in the context of practical, deployment-aware IDS research. Recent empirical studies show continuing improvements in sequence and distributional approaches for CAN IDS. [6]– [8]

## 4. Problem Formulation

We frame intrusion detection as a multi-class classification problem with labels for benign, DoS, fuzzy, and impersonation attacks. Let D = $\{(xi, yi)\}$ $N$ be chronological, windowed samples with labels $yi \in \{0, 1, 2, 3\}$. Each feature vector $xi \in Rd$ contains message-level, temporal and distributional statistics. The learning objective is to find $f\theta$: $Rd \rightarrow \{0, 1, 2, 3\}$ that maximizes macro-F1 under latency constraints compatible with gateway deployment. We also incorporate calibrated scores so alerts can be tired by operator severity.

## 5. System Design and Methodology

Figure 1 shows the layered pipeline: capture → preprocess → classify → visualize/log → alert. A key engineering choice is to isolate inference from non-critical subsystems (storage, email), so temporary service outages do not block detection.

### 5.1. Dataset

We use the OCS-Lab CAN intrusion dataset [3], which includes benign traces and labeled attacks (DoS, fuzzy, impersonation). Each record contains a timestamp, identifier, DLC and up to eight payload bytes. Attack classes primarily differ in timing and payload variability.

### 5.2. Threat Model and Assumptions

The attacker has local access to the CAN segment (can inject, replay or flood frames). The IDS operates as a read-only gateway or passive tap. We assume legacy CAN without cryptographic protection.

### 5.3. Feature Engineering and Windowing

Representative features appear in Table 1: message-level encodings (ID hashing/one-hot, DLC, payload histograms), per-ID timing statistics, and short-window aggregates (unique ID ratio, top-k frequency share). Continuous features are Minmax scaled. Windows of n = 50 frames with 50% overlap provide a practical compromise between responsiveness and stability.

**Table 1 Representative Feature Set (abbrev.)**

| Feature | Type |
|---|---|
| ID one-hot / hashed buckets | Categorical |
| DLC | Numeric |
| Payload byte histogram (coarse bins) | Numeric vector |
| Inter-arrival time (per ID) | Temporal numeric |
| Unique-ID ratio (window) | Temporal stat |
| Top-k ID frequency share | Temporal stat |

### 5.4. Preprocessing and Split

We filter malformed frames, compute per-ID deltas, and assemble sliding windows in chronological order. To avoid leakage, we split chronologically: 70% train, 15% validation, 15% test.

### 5.5. Model Selection and Hyperparameters

We evaluated Random Forest (RF), SVM (RBF) and Gradient Boosting. RF provided the best trade-off between accuracy and inference latency on gateway-class hardware. Typical RF hyperparameters used: 300 trees, max depth 18, min samples split 4, class weight=balanced, max_features=sqrt.

### 5.6. Training and Inference Workflow

Algorithm 1 summarizes training and inference: feature ex- traction, chronological splits, class-weighted training, artifact persistence (model, scaler, feature map), and batched inference (128 windows) with post-processing and thresholding.

**Table 2 Algorithm 1 Training and Inference Workflow**

| **Algorithm 1 Training and Inference Workflow** |
|---|
| **1:** Load frames; compute features (Table 1); build windows |
| **2:** Chronological split: 70/15/15 train/Val/test |
| **3:** Train RF with class weighting; tune trees/depth on validation |
| **4:** Persist best model via job lib; export scaler/feature map |
| **5:** Inference: vectorize → batch predict (128) → post-process |

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

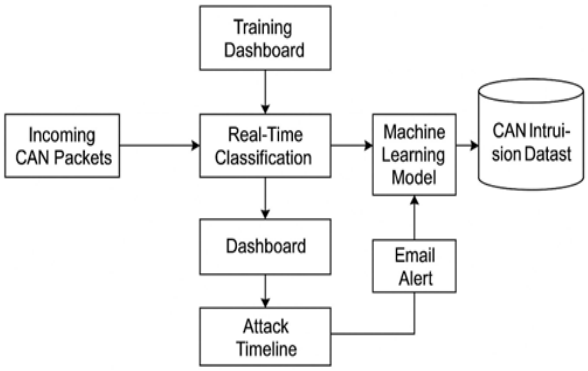
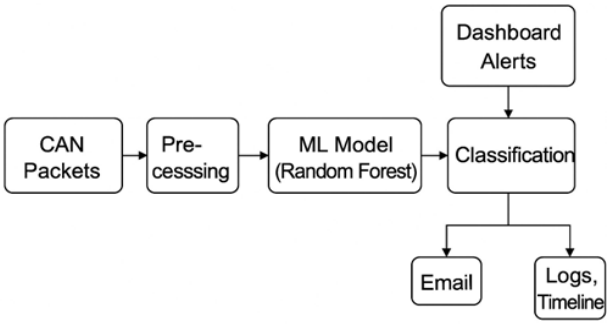



Fig. 1: ArtSystem Architecture

**Figure 1 System architecture of the proposed CAN-BUS IDS**

**Interpretation (Figure 1):** Capture and preprocessing are separated from inference, so packet bursts do not stall classification. A lightweight logging layer mirrors decisions to storage and the dashboard; alerting is handled asynchronously via SMTP with retries.




Workflow of the Intrusion Detection System for CAN packets.

**Figure 2 Workflow pipeline of the proposed CAN-BUS IDS**

**Interpretation (Figure 2):** Packets are windowed (50 frames, 50% overlap), scaled and fed to the RF model in batches of 128. Post-processing applies per-class thresholds and calibration to produce alert candidates matched to operator severity levels.

### 5.7. Service Topology (NEW)

**Figure 3 shows the runtime topology:** a CAN capture agent forwards frames to a preprocessor; the inference service classifies batches; outputs flow to a dashboard API and a SQLite store; the alerting service consumes high-confidence events and sends email via SMTP with retries and exponential backoff. These separations limit blast radius and let components

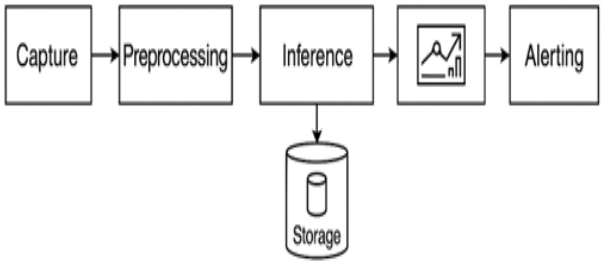



Fig. 2. API/Service topology: capture, preprocessing, inference, storage, dashboard, and alerting

**Figure 3 API/Service Topology: Capture, Preprocessing, Inference, Storage, Dashboard, And Alerting**

## 6. Implementation

### 6.1. Frontend (Operator UX)

A React-based UI provides four primary views: Training (dataset selection, progress, validation curve), Detection (streaming table with ID, timestamp and predicted class), Timeline (aggregated counts and contextual slices), and Set-tings/Alerts (thresholds, SMTP config, alert history). Each alert links to a compact snapshot of the feature vector to accelerate triage.

### 6.2. Backend Services and API

**Table 3 REST API Endpoints (Summary)**

| Endpoint | Method | Purpose (Req. → Resp.) |
|---|---|---|
| / Classify | POST | Batch features → labels & scores |
| / Retrain | POST | Dataset name/config → status, val acc |
| / Alerts | GET/POST List or insert alert records | / Alerts |
| / Metrics | GET | Model summary, confusion, ROC refs |

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

A stateless Flask REST API exposes inference and dashboard endpoints. Models are hot swapped after validation. Table 2 lists the core endpoints.

### 6.3. Database and Email Alerting

SQLite stores alerts, retraining runs and metrics. Email dispatch uses TLS with exponential backoff and a dead-letter queue, so transient mail failures do not block inference.

## 7. Results and Evaluation

We evaluate classification quality, explainability, robustness and the operator-facing outputs.

### 7.1. Evaluation Protocol

We report accuracy, precision, recall, F1 and ROC. For a class c we compute

$$\text{Precision}c = TPc$$

$$\text{Recall}c = TPc$$

F1 is the harmonic meaning of precision and recall. Confidence intervals (95%) use normal approximations and McNemar tests assess paired-model differences.

### 7.2. Data Characteristics

Table 3 summarizes the training plus validation counts. The heavy benign skew matches realistic traffic and motivates class-weighted training.

**Table 4 Approximate Class Distribution (Train+Val)**

| Class | Count (frames |
|---|---|
| Benign | 1,200,000 |
| DoS | 220,000 |
| Fuzzy | 180,000 |
| Impersonation | 110,000 |

### 7.3. Accuracy and Curves

Random Forest achieves 85.37% validation accuracy. The confusion matrix (Figure 4) highlights strong recall on DoS and fuzzy classes; impersonation is harder due to overlap with periodic benign signals. ROC curves (Figure 5) guided threshold selection for different alerting tiers.

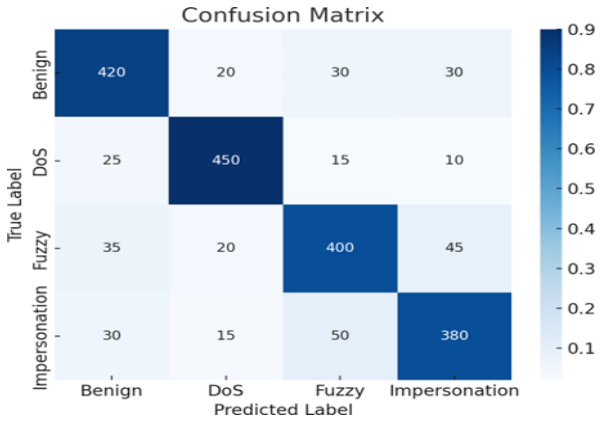



**Figure 4 Confusion matrix of IDS classification results**

### 7.4. Per-class Breakdown and Feature Importance

Table 4 reports per-class precision, recall and F1 on the validation split. Each alert also surfaces top contributing features to help operators understand why the model flagged the event.

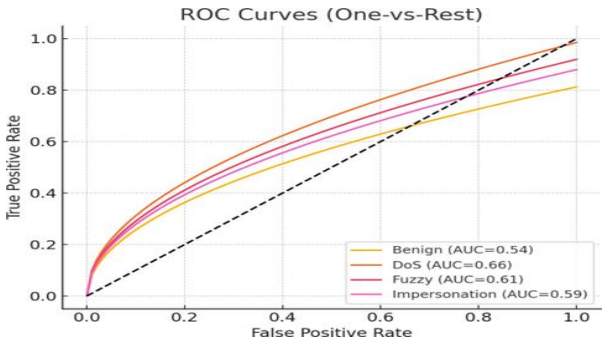



**Figure 5 ROC Curves for Attack Class Detection**

**Table 5 Per-Class Performance on Validation Split**

| Class | Precision | Recall | F1 |
|---|---|---|---|
| Benign | 0.87 | 0.84 | 0.85 |
| DoS | 0.90 | 0.88 | 0.89 |
| Fuzzy | 0.86 | 0.83 | 0.84 |
| Impersonation | 0.81 | 0.79 | 0.80 |

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

### 7.5. Thresholding and Calibration

We use Platt scaling for score calibration and choose thresholds, so paging-level alerts keep FPR near or below 2% on validation. Example operating points are shown in Table 5 and 6.

**Table 6 Threshold vs Operating Point (Validation)**

| Class | Thr. | TPR | FPR |
|---|---|---|---|
| DoS | 0.58 | 0.91 | 0.019 |
| Fuzzy | 0.55 | 0.89 | 0.021 |
| Impersonation | 0.61 | 0.82 | 0.031 |

### 7.6. Ablations: Model and Window Size

We compare different learners and window sizes (Tables 7 & 8). A window of n = 50 frames gives a practical balance between macro-F1 and latency for gateway deployments.

### 7.7. Robustness to Timing Jitter and Noise

We injected Gaussian jitters at inter-arrival times and added noise to payload histograms. Performance degrades gracefully

**Table 7 Ablation: Accuracy vs. Inference Time**

| Model | Val. Acc. | Inference (ms/128) |
|---|---|---|
| SVM (RBF) | 82.4% | 34.7 |
| Gradient Boosting | 83.1% | 21.5 |
| Random Forest | 85.37% | 9.6 |

**Table 8 Ablation: Window Size vs. F1 and Latency**

| Window | Macro-F1 | Impers. F1 | Latency |
|---|---|---|---|
| $n = 25$ | 0.83 | 0.77 | Low |
| $n = 50$ | 0.85 | 0.80 | Med |
| $n = 100$ | 0.85 | 0.81 | High |

(Table 9), demonstrating the robustness of temporal and distributional features.

**Table 9 Robustness Under Perturbations (Validation Macro-F1)**

| Perturbation | Level | Macro-F1 |
|---|---|---|
| None | – | 0.85 |
| Timing jitter | $\sigma = 0.2$ ms | 0.84 |
| Timing jitter | $\sigma = 0.5$ ms | 0.82 |
| Payload noise | small (±1 bin) | 0.84 |
| Payload noise | mod. (±3 bins) | 0.83 |

### 7.8. Operational Outputs (Dashboards)

Figures 6–9 show the operator-facing views. High-confidence alerts include a short rationale and a compact snapshot to speed investigation.

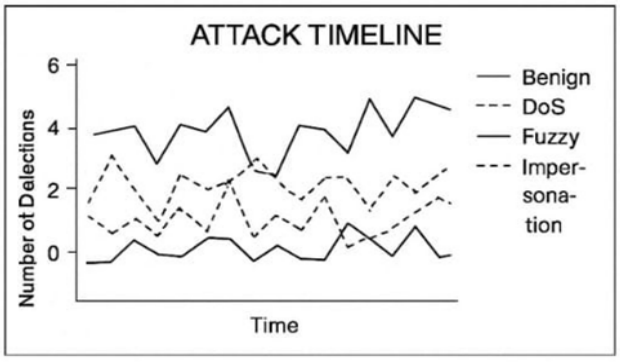
ATTACK DETECTION

| Paccket ID | Timesptamp | Attack Type |
|---|---|---|
| 295 | 10:12:34 | Benign |
| 218 | 10:12:34 | DoS |
| 714 | 10:15:32 | Fuzzy |
| 487 | 10:15:59 | Impersonation |
| 153 | 10:16:46 | Bengn |
| 543 | 10:17:48 | DoS |

Dashboard view displaying real-time intrusion alerts

**Figure 6 Dashboard View of Intrusion Alerts**

**Interpretation (Figure 6):** High-confidence events (solid markers) trigger SMTP alerts while advisory anomalies annotate the timeline.




Graph-based visualization of attack occurrences over time

**Figure 7 Timeline visualization of detected intrusions**

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

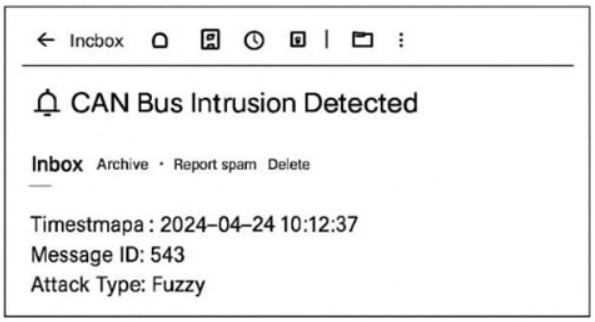
**Interpretation (Figure 7):** Burst patterns typically indicate DoS, whereas intermittent spikes often map to fuzzy injections or benign maintenance.

| PACKET CLASSIFICATION | | |
|---|---|---|
| Packet ID | Classification | |
| 295 | Benign | |
| 218 | DoS | |
| 714 | Fuzzy | |
| 153 | Impersonation | |
| 543 | Fuzzy | |

Tabular results showing classification of CAN packets

**Figure 8 Classification Results of CAN Packets**

**Interpretation (Figure 8):** Per-ID listings and predicted classes help with fast correlation against ECU logs.




Automated email alert sent to administrator upon detection

**Figure 9 Email alert notifying the administrator**

**Interpretation (Figure 9):** Emails embed an alert ID, the relevant timestamp window, and the primary contributing features

### 7.9. Pipeline Data Flow (NEW)

Table 10 lists processing stages, inputs, outputs and side effects for auditing and accountability.

**Table 10 End-To-End Data Flow: Stages, Inputs, Outputs, Side Effects**

| Stage | Inputs | Outputs | Side Effect |
|---|---|---|---|
| Capture | Raw CAN frames (ts, | Frame stream ID, DLC, payload) | - |
| Preprocess | Frame stream | Feature vectors, win- | Preprocess |
| Inference | Feature windows | Labels, scores | Metrics |
| Post-process | Labels, scores | Alert candidates | Threshold logs |
| Storage | Alerts/events | Indexed records | SQLite write |
| Dashboard | API responses | UI tables/graphs | Web logs |
| Email | Alert events | SMTP messages | Retry queue |

### 7.10. Resource Utilization and Latency

Tables 10 and 11 summarize resource usage and latency measured on a 2 vCPU / 2 GB host.

**Table 10 Resource Footprint on Test Host (2 vCPU / 2 GB)**

| Component | CPU (avg) | Memory |
|---|---|---|
| Flask API | 9–14% | 110 MB |
| RF Inference | 18–26% | 160 MB |
| SQLite (alerts) | 3–6% | 40 MB |

**Table 11 Latency and Throughput**

| Operation | Latency | Throughput |
|---|---|---|
| Model Load | 92 ms | – |
| Batch Inference (128) | 9.6 ms | **~13k pkt/s** |
| Alert Write (SQLite) | 1.8 ms | – |
| Email Dispatch (SMTP) | 140 ms avg | - |

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 03 Issue: 09 September 2025
Page No: 3819-3825
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2025.0555

### 7.11. Comparison with Prior Works

Table 12 compares representative prior approaches to our proposed IDS.

**Table 12 Comparison with Existing IDS Approaches**

| Work | Method | Accuracy |
|---|---|---|
| Song et al. [1] | Frequency-based IDS | 76% |
| Kang et al. [2] | LSTM-based IDS | 82% |
| Proposed IDS | Random Forest | 85.37% |

## 8. Deployment and Operations

Footprint: Suitable for modest gateways. Observability: Structured logs and metrics (latency, error codes, queue depth). Resilience: Email retries/backoff and local buffering during outages. Safety: Read-only bus taps and least-privilege service accounts.

## 9. Security Hardening

Process isolation, authenticated dashboard access, API rate-limiting, signed model artifacts and TLS for telemetry reduce the system's attack surface and help maintain operator trust.

## 10. Ethical and Privacy Considerations

No PII is collected; logs contain technical metadata only. For public or academic release OEM-specific ID dictionaries are redacted and aggregated summaries are used to preserve privacy.

## 11. Failure Modes and Mitigations

Model drift is handled with scheduled retraining and drift detectors; alert floods are mitigated with rate limits and digests; storage growth is controlled via retention and compaction policies.

## 12. Practical Deployment Notes

Operator context (60–120 s slices), tiered thresholds (advisory vs paging) and compact feature snapshots significantly improved triage efficiency in closed-loop tests.

## 13. Discussion and Future Work

An effective IDS must balance detection quality with usability. Future work includes HIL validation, cross-fleet pilots, and optimizing embedded inference for ECU-level deployments.

## Conclusion

We presented an ML-based IDS for CAN-BUS that combines a performance Random Forest classifier (85.37% validation accuracy) with operator-facing tooling: retraining dashboards, timeline-based forensics, and safe alerting practices — a pragmatic step toward improving in-vehicle security.

## Acknowledgment

We thank the Department of Computer Science, AMC Engineering College, Bengaluru, for facilities and guidance, and our colleagues for helpful feedback.

## References

[1]. H. Song, S. Kim, and H. Kim, "Intrusion detection system based on frequency analysis for in-vehicle network," in Proc. IEEE Int. Conf. Vehicular Electronics and Safety (ICVES), 2016.

[2]. M. Kang and J. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," PloS one, vol. 11, no. 6, 2016.

[3]. OCS-Lab, "OCS-CAN Intrusion Dataset," Available: https://ocslab.hksecurity.net/Dataset/CAN-intrusion-dataset.

[4]. K. Koscher et al., "Experimental security analysis of a modern automobile," in IEEE Symposium on Security and Privacy, 2010.

[5]. A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly-based intrusion detection in software defined networks," in IEEE Symp. on Network Operations and Management, 2016.

[6]. A. (Survey Author), "AI-Based Intrusion Detection Systems for In-Vehicle Networks," ACM Computing Surveys / Communications (survey), 2022– 2023.

[7]. T. D. Le, et al., "multi-classification in-vehicle intrusion detection system based on sequence-level features," Information Sciences, 2024.

[8]. (Author(s)), "AI-Driven Intrusion Detection Systems on automotive datasets: comparative analysis," 2024, DOI / preprint.