SGI-JOURNAL OF SCIENCE AND TECHNOLOGY

VOLUME. 10
ISSUE. 1
FEB., 2026

GENERATIVE ADVERSARIAL NETWORK–BASED INTRUSION DETECTION FOR SECURING IN-VEHICLE COMMUNICATION IN ELECTRIC VEHICLES

[1]Bogineni Kasi Pavan Kalyan, [2]Mergu Siri Chandana, [3]Navya Karimalla & [4]Wisam Bukaita
[1,2,3&4]Math and Computer Science Department, Lawrence Technological University, South Field, USA.

Abstract
The increasing connectivity of in-vehicle electronic control systems has intensified the need for robust cyber security solutions, especially for the Controller Area Network (CAN) bus. This study proposes a deep learning–based Intrusion Detection System (IDS) utilizing Generative Adversarial Network (GAN) architecture to detect anomalous CAN bus traffic in real time. The GAN model is trained solely on legitimate CAN messages, enabling it to learn the underlying statistical patterns of normal communication without relying on predefined attack signatures. The proposed GAN-IDS demonstrate strong detection performance, achieving an accuracy of 98.7% and an F1-Score of 98.5%, outperforming conventional deep learning baselines. To assess deployment feasibility, the discriminator is optimized using Tensor Flow Lite (TFLite) and deployed on a Raspberry Pi 4 integrated with a PiCAN2 interface. Hardware evaluation confirms real-time operation with a low detection latency of 2.9 milliseconds per message sequence. System interpretability is further enhanced through SHapley Additive exPlanations (SHAP), which identify CAN ID, engine torque, and RPM as the most influential features contributing to anomaly classification. The proposed GAN-based IDS offer a scalable, manufacturer-independent, and non-intrusive cyber security solution for modern Electric Vehicles. Its combination of high detection performance, real-time hardware deployment, and interpretable decision-making marks a significant step toward more intelligent and resilient security mechanisms for future connected and autonomous vehicles.
Keywords: CAN Bus Security, Generative Adversarial Networks, Intrusion Detection System, Electric Vehicles, Real-time Anomaly Detection.

I. Introduction

The rapid advancement of automotive industry is characterized by the integration of networked electronic control systems, transforming modern vehicles, particularly Electric Vehicles (EVs), into complex cyber-physical systems. This transition, while enabling enhanced functionality, efficiency, and connectivity, simultaneously introduces significant cyber security vulnerabilities. At the core of in-vehicle communication is the Controller Area Network (CAN) bus, a proto- col critical for coordinating safety-essential functions such as braking, steering, and propulsion. Despite its operational efficiency and reliability, the CAN protocol was not designed with security in mind, fundamentally lacking native mechanisms for authentication, encryption, and message integrity verification. This inherent security weakness leaves the CAN bus highly susceptible to sophisticated cyber-attacks, including message spoofing, injection, and denial-of-service, which could lead to catastrophic functional failure. This vulnerability underscores the urgent requirement for robust and intelligent security solutions.

Traditional Intrusion Detection Systems (IDS) for vehicular networks typically rely on rule-based or statistical methods. While effective against known threats, these conventional approaches struggle with generalization to novel or zero-day attacks and often suffer from a high rate of false positives in dynamic operating environments. Furthermore, most existing deep learning–based IDS solutions are discriminative in nature, focusing primarily on classification and failing to fully capture the complex, underlying statistical distribution of legitimate network traffic. This limitation highlights a critical research gap: the need for an advanced IDS framework capable of providing non-signature-based anomaly detection in real time. Crucially, any such solution must also incorporate explain ability features to ensure transparency and trustworthiness in its decision-making, a prerequisite for integration into safety-critical automotive systems.

To address these challenges, this research proposes a novel Intrusion Detection System leveraging Generative Adversarial Network (GAN) architecture. The GAN framework is uniquely suited for anomaly detection as its training process enables the discriminator to learn a precise model of normal CAN traffic distribution, allowing it to accurately identify messages that deviate from legitimate communication pat- terns. The main contributions of this research are as follows:

GAN-Based Anomaly Detection: Design and implementation of GAN-based IDS specifically tailored for time-series CAN bus data, enabling robust, non-signature-based intrusion detection.

Real-Time Embedded Prototype: Validation through the development of a functional hardware prototype utilizing a Tensor Flow Lite discriminator model on a Rasp- berry Pi 4 with a PiCAN2 interface, demonstrating feasibility for real-time, on-vehicle deployment.

Explainable Decision-Making: Integration of SHapley Additive exPlanations (SHAP) analysis to interpret the model's classification outcomes, identifying the most influential CAN features (e.g., CAN ID, torque, RPM) and enhancing the system's trustworthiness.

II. Literature Review

The rapid integration of electronic control units and net- worked communication in modern vehicles has increased exposure to cyber-attacks, particularly within the Controller Area Network (CAN) bus, which lacks inherent authentication and encryption capabilities. Foundational studies first demonstrated how easily adversaries could manipulate CAN traffic. Hoppe et al. [1] showed that simple replay and injection attacks could disrupt safety-critical vehicle functions, prompting early interest in lightweight defensive mechanisms. In response, Groza et al. [13] developed LiBrA-CAN, a broadcast authentication protocol that strengthened message integrity without imposing significant bandwidth overhead. Together, these studies established that CAN traffic follows stable statistical and temporal patterns that intrusions typically disturb.

Statistical, entropy-based anomaly detection emerged as one of the earliest non-intrusive defensive strategies. Müter and Asaj [2] demonstrated that monitoring ID entropy could reveal sudden irregularities caused by malicious traffic, while Wu et al. [7] extended this work through a sliding-window entropy model capable of capturing short-lived anomalies in real time. Researchers also leveraged timing and sequential features of CAN communication. Song et al. [3] showed that abnormal inter-frame timing patterns often precede or accompany attacks, whereas Marchetti and Stabili [4] demonstrated that deviations in expected CAN ID sequences can serve as reliable intrusion indicators. These methods revealed that anomalies can manifest not only in message frequency but also in timing rhythms and ordering structures.

Beyond protocol-level analysis, physical-layer detection expanded the defensive landscape. Choi et al. [5] introduced VoltageIDS, which analyzed voltage waveforms to differentiate spoofed signals from authentic ones. Groza and Murvay [6] similarly proposed memory-efficient Bloom filter–based detection suitable for embedded automotive systems. While effective, such approaches require specialized hardware or strict assumptions about physical stability, limiting practicality across diverse vehicle platforms.

The introduction of deep learning substantially reshaped CAN intrusion detection by enabling automated feature ex- traction from complex vehicular data. Early work by Kang and Kang [8] demonstrated the superiority of deep neural networks over traditional statistical techniques. More recent studies employ context-aware architectures: Wei and Wang [14] utilized attention mechanisms within an auto encoder to model interdependencies among CAN packets, while Feng et al. [15] applied transformer-based encoders to capture long-range temporal structures in vehicle communication streams.

These models greatly improved accuracy but still rely heavily on labeled attack data an ongoing challenge due to the scarcity of real-world intrusion traces.

Generative modeling emerged as a promising alternative by learning only the distribution of benign traffic. Seo et al. [9] developed GIDS, GAN-based IDS capable of detecting anomalous messages as deviations from learned normal pat- terns. Broader generative modeling surveys by Xia et al. [11] and Creswell et al. [12] established GANs as strong candidates for unsupervised anomaly detection. Recent automotive applications include Bai et al. [16], who utilized conditional GANs for controlled anomaly synthesis, and Ren et al. [17], who employed a cycle-consistent GAN to improve anomaly reconstruction accuracy. Meanwhile, Zhang et al. [10] introduced a bit-level whitelist approach that learns permissible bit structures, achieving fine-grained detection with low false-positive rates.

Complementary advancements include hybrid and self-supervised methods. Kim et al. [18] combined auto encoding with contrastive learning to enhance detection robustness under limited data conditions; while Huang et al. [19] applied graph neural networks to represent relational structures among CAN IDs. Martínez et al. [20] used variational inference to incorporate uncertainty into anomaly scoring, and Liu et al. [21] introduced a multi-view learning framework that fuses timing, payload, and signal-level features. Wang et al. [22] further demonstrated that hierarchical feature fusion improves generalizability across vehicle platforms.

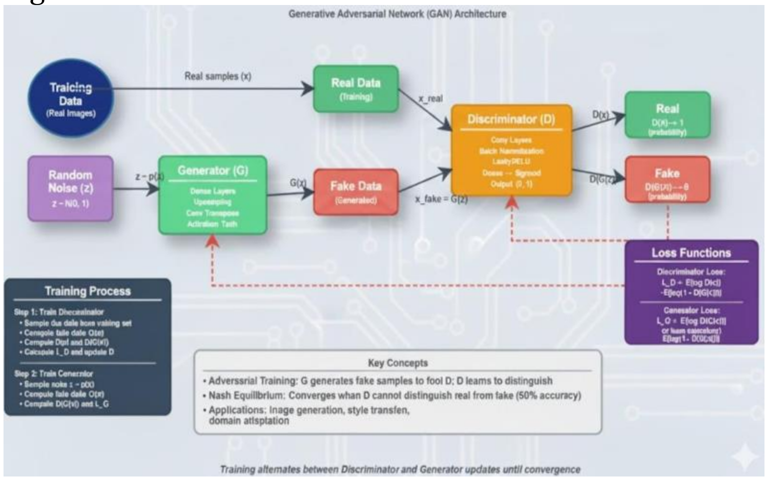
Despite substantial progress, key gaps remain. Most deep learning and GAN-based IDS models still lack real-time feasibility on resource-constrained automotive hardware. Few studies incorporate model interpretability, leaving anomaly sources opaque to developers and engineers. Additionally, cross-vehicle generalization is rarely evaluated, and hardware prototype implementations are limited. These limitations motivate the present study, which introduces a real-time, hardware-deployable GAN-based IDS enhanced with SHAP interpretability to address practical scalability, transparency, and deployment challenges.

III. Research Methodology

GAN-based IDS Architecture

The methodological framework for this study follows a structured and logically connected workflow designed to develop, train, and evaluate a Generative Adversarial Net- work (GAN)–based Intrusion Detection System (IDS) for real-time CAN bus anomaly detection. As illustrated in Figure 1, the approach progresses through four major phases: dataset selection and preprocessing, temporal sequence construction and feature normalization, GAN architecture designs and adversarial training, and, evaluation of the discriminator as the final IDS detector.

Figure 1. Architecture of the GAN-based Intrusion Detection System.


Generative Adversarial Network (GAN) Architecture
Traicing Data (Real Images)
Real samples (x)
Real Data (Training)
x_real
Random Noise (z) z ~ N(0, 1)
z ~ p(z)
Generator (G) Dense Layers Upsampling Conv Transpose Activation Tanh
G(x)
Fake Data (Generated)
x_fake = G(z)
Discriminator (D) Conv Layers Batch Normalization LeakyReLU Dense → Sigmoid Output (0, 1)
D(x)
Real D(x) → 1 (probability)
D(G(z))
Fake D(G(z)) → 0 (probability)
Loss Functions
Training Process
Step 1: Train Discriminator
Step 2: Train Generator
Key Concepts
• Adverssrial Training: G generates fake samples to fool D; D leams to distinguish
• Nash Equilibrium: Converges whan D cannot distinguish real from fake (50% accuracy)
• Applications: Inage generation, style transfen, domain atlsptation
Training alternates between Discriminator and Generator updates until convergence

Dataset Description

The study utilizes a widely adopted, publicly available CAN intrusion dataset containing approximately 3 million CAN messages. The dataset includes both benign traffic and multiple injected attack types, at an approximate 70: 30 ratio of normal to malicious frames. Each CAN frame comprises a timestamp, CAN ID, Data Length Code (DLC), and eight data bytes (D0–D7). Attack scenarios include:

Spoofing: malicious insertion of fabricated CAN IDs
Flooding: rapid high-frequency message injection
Replay: resending previously captured valid messages at incorrect times

This dataset provides sufficient variability and volume to support deep temporal learning of normal communication patterns.

Data Preprocessing and Temporal Encoding

To prepare the raw data for deep learning, the following preprocessing stages were implemented:

Feature Encoding and Normalization

All categorical and numerical fields (CAN ID, DLC, data bytes) were encoded into tensors and normalized using Min-Max scaling to [−1, 1], aligning with the tanh activation used in the Generator and stabilizing adversarial training.

Temporal Windowing

Because CAN traffic is inherently sequential, normalized frames were grouped into fixed-length sequences of 50 consecutive messages, preserving time-dependent relationships critical for anomaly detection.

Dataset Partitioning

The processed sequences were divided into training, validation, and test sets using stratified sampling:

70% training
15% validation
15% testing

This ensured a consistent class distribution across all sub- sets.

In this research, each message collected from the CAN bus consisted of five key attributes, as summarized in Table 1.

Table 1. Structure of Collected CAN Messages.

Component	**Value (Sample)**	**Description**
Timestamp	1478191030	UNIX time when message was captured
CAN ID	316	Message type or sender identifier
Payload	8 5 22 68 9 22 20 0 75	Operational data (9 bytes total)
On/Off State	8	Indicates system ON (value = 8)
Flag	R	Regular message (not attack)

The timestamp (e.g., 1478191030) represents UNIX time, corresponding to November 3, 2016, 09:37:10 UTC, used for sequential ordering and temporal analysis. The CAN ID (316) is stored in hexadecimal to minimize memory space, later converted to decimal (316 → 790) for model processing using the formula:

Decimal = (Hex digit) × 16^n

The payload encapsulates operational data transmitted among ECUs, while the flag distinguishes regular (R) messages from injected (T) messages.

Payload Structure and Anomaly Patterns

Each CAN message contains attributes relevant to vehicular operation shown in Table 2, including onboard status flags, sensor readings (speed, RPM, temperature, pressure), telemetry values, and torque. Attack messages deliberately intro- duce inconsistent values within these fields, mimicking re- al-world malicious manipulations such as bogus RPM surges or falsified sensor outputs.

These anomalies serve as ground-truth labels for performance evaluation but are not used during GAN training, as the approach relies on learning the distribution of normal traffic only.

In This study, a total of 654,897 messages were logged, comprising both normal and injected traffic. Abnormal messages are deliberately crafted to exhibit inconsistent sensor readings compared to typical EV behavior, as illustrated in Table 2.

Table 2. Payload Range and Anomalies.

Injected Feature	Normal Range	Anomaly Value
Status Flag of Engine	0–255	69
Sensor Temperature	0–184	41
Vehicle Speed	0–252	36
RPM Sensor	0–255	255
Pressure Sensor	0–255	41
Telemetry Reading	0–255	36
Default/Null Value	0–209	0
Torque	0–255	0

These injected values emulate spoofed RPMs, false speed readings, and fabricated pressure or temperature signals, mimicking a hacker's attempt to mislead the ECU or disable key vehicle safety functions.

GAN Architecture for Unsupervised IDS

The proposed IDS is structured around a standard Generator–Discriminator GAN configuration.

Generator (G)

The Generator models the statistical distribution of normal CAN sequences.
Input: 100-dimensional noise vector
Layers: fully connected layers followed by stacked LSTM layers to capture sequential structure
Output: synthetic CAN sequences (length = 50) scaled by tanh

Its purpose is to produce realistic sequences that challenge the Discriminator and improve its sensitivity to deviations from learned distributions.

Discriminator (D)

The Discriminator acts as the anomaly detector during both training and deployment.
Input: real or synthetic sequence of 50 frames
Layers: two 1D CNN layers for spatial feature extraction, followed by a Bidirectional LSTM for temporal modeling
Output: sigmoid probability indicating whether the sequence is legitimate
The Discriminator ultimately becomes the trained IDS once the GAN converges.
An excerpt of the collected data is shown in Table 3, high- lighting both normal and abnormal entries.

Table 3. Sample Data from Dataset.

Timestamp	CAN ID (Dec)	On/ Off	Stamp Flag	Temp/Pressure	Speed/RPM	Sensor 1	Sensor 2	Telemetry	Default	Torque
9:37:10.45	704	8	21	0	0	0	0	0	0	0
9:37:10.46	790	8	69	41	36	255	41	36	0	255
9:37:10.47	1264	8	0	0	0	128	0	103	209	19
9:37:10.48	1349	8	216	0	0	134	0	0	0	0
9:37:10.49	790	2	69	41	36	255	41	36	0	255
9:37:10.50	790	8	69	41	36	255	41	36	0	255
9:37:10.51	790	2	69	41	36	255	41	36	0	255

9:37:10.52	790	8	69	41	36	255	41	36	0	255

On/Off state; The color green represents that the car is turned on as code is 8. When the color is red and code 2 appears that represent that the car is turned off.

Adversarial Training

The GAN follows the standard minimax optimization objective:
Training alternates between:
Updating D to maximize its ability to distinguish real and generated sequences
Updating G to minimize detection and produce more realistic samples
Hyperparameter Summary
Hyperparameters were optimized through grid search and empirical testing:
Optimizer: Adam ($\beta1 = 0.5$, $\beta2 = 0.999$)
Learning rate: 0.0002
Batch size: 64
Epochs: 100
Loss function: Binary Cross-Entropy
Dropout: 0.3
Activation functions: Leaky ReLU, tanh, sigmoid
These parameters achieved stable convergence without mode collapse and enabled deployment in constrained embedded environments.
The GAN model is trained through an alternating competitive process until a Nash Equilibrium is reached, at which point the discriminator becomes an expert anomaly detector.
Formal Training Objective: The adversarial loss function is the standard minimax objective:

$$\min_G \max D\ V(D, G) = \mathrm{E}_{x\sim \mathrm{pdata(x)}}[\log D(x)] + \mathrm{E}_{\mathrm{z}\sim \mathrm{pz(z)}}[\log(1 - D(G(z)))]$$

Where D(x) is the probability D assigns to a real sample x, and G(z) is the synthetic sample generated from noise z.

IV. Experimental Evaluation Setup

After training, the Discriminator was exported as a Ten- sorFlow Lite (TFLite) model and deployed on a Raspberry Pi 4 with a PiCAN2 interface. Evaluation metrics included:
Accuracy
Precision, Recall, F1-Score
Detection latency per message sequence
Robustness across attack types
This setup validates the real-time applicability of the pro- posed IDS in realistic automotive hardware environments.

Hardware and Software Environment

The experimental validation is conducted in two distinct environments: a high-performance workstation for model training and a bespoke embedded platform for real-time deployment and validation.

Training Environment

The GAN model is trained and validated on a high-performance workstation to manage the computational demands of the adversarial process.
Processor: Intel® Core™ i9-13900K @ 3.0 GHz.
GPU: NVIDIA RTX 4080 (16 GB GDDR6X VRAM).
RAM: 64 GB DDR5.
Operating System: Ubuntu 22.04 LTS (64-bit).
Software Frameworks: Python 3.10, TensorFlow 2.14, Keras, NumPy, and Pandas.
Training Procedure
The GAN model was trained using an adversarial training loop consisting of alternating updates between the generator and discriminator:
Step 1 – Discriminator Update:

The discriminator was trained on batches containing both real CAN sequences and synthetic sequences generated by the generator.
Step 2 – Generator Update:
The generator's parameters were updated to minimize the discriminator's ability to distinguish fake samples, thereby learning to approximate the distribution of real CAN traffic.
Step 3 – Convergence Monitoring:
Training was continued until both networks reached equilibrium, observed through stabilization of generator and discriminator loss curves.
Step 4 – Model Selection and Conversion:
The best-performing discriminator model on the validation set was saved and converted into TFLite format for prototype deployment.

Real-time Prototype

To confirm the practical feasibility and low-latency performance of the IDS in an automotive context, a functional hardware prototype is developed.
Main Processing Unit: Raspberry Pi 4 Model B (4 GB RAM), serving as the edge computing platform.
CAN Interface: PiCAN2 Interface Board, which pro- vides the physical layer connection to the Controller Area Network (CAN) bus, enabling real-time message capture and transmission.
Model Deployment: The final trained Discriminator model was converted into the TensorFlow Lite (TFLite) format. TFLite is a specialized framework that optimizes model inference for resource-constrained embedded devices, ensuring minimal computational overhead and maximal processing throughput.
User Interface: A 7-inch Capacitive Touchscreen Dis- play is integrated to provide a Graphical User Interface (GUI) for real-time monitoring and visual/auditory alert notifications upon intrusion detection.
Table 4 illustrate the physical components and their interconnections, including the OBD-II to DB9 cable with power switch, car USB power adapter, Raspberry Pi 4 Model B, PiCAN2 HAT, Micro-HDMI to HDMI Cable and 7-inch touchscreen display. These figures collectively demonstrate the compact and modular nature of the prototype, allowing easy dashboard installation forreal-time testing.

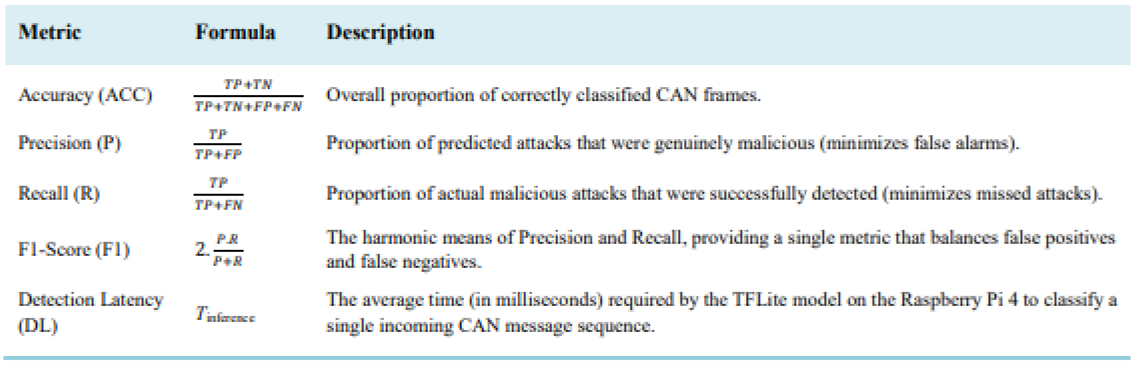
Table 4. Hardware Prototype Components.

Unit	image	Purpose
OBD-II to DB9 Cable with Power Switch		This cable links the OBD-II diagnostic port of the car to the DB9 connector on the PiCAN2 board. It makes it possible for the Raspberry Pi and the car's CAN system to exchange data in real time.
Car USB Power Adapter (Anker Dual Port Charger)		Car USB is used to supply the Raspberry Pi with power from the 12V cigarette lighter plug in the car. It ensures converting 12V DC from the car into 5V/3A USB output.
7-Inch Touchscreen Display		The user interface component of the IDS prototype. Displays real-time CAN message logs, attack alerts, and system diagnostics for driver visibility and interaction during testing.
PiCAN2 HAT Board		A CAN bus interface card that is directly connected to the GPIO header of the Raspberry Pi. It sends and receives CAN messages using the MCP2551 transceiver and MCP2515 CAN controller. With the help of this board, the Raspberry Pi may interact with the car's may network to detect and notify anomalies.
Raspberry Pi 4 Model B (4GB)		The intrusion detection system's central processing unit. It analyzes CAN messages in real time using the trained TensorFlow Lite GAN model. The Raspberry Pi 4 has the processing power to manage data processing, visualization, and alarm production all at once.
Micro-HDMI to HDMI Cable		IT connects the 7-inch touchscreen display to the micro-HDMI connector on the Raspberry Pi. The user interface can display real-time CAN statistics, status messages, and alarm notifications thanks to this cable's ability to send both audio and video signals.

Evaluation Metrics

The performance of the proposed Intrusion Detection Sys- tem (IDS) is quantitatively evaluated using a combination of classification and time-based performance metrics. These metrics provide a comprehensive understanding of the system's detection capability, classification reliability, and real-time feasibility when deployed in embedded vehicular environments. The evaluation was conducted on an unseen test dataset to ensure unbiased performance validation and assess the model's generalization ability. Table 5 summarizes the assessment metrics and their corresponding mathematical formulations. These include Accuracy (ACC), Precision (P), Recall (R), F1-Score (F1), and Detection Latency (DL). Each metric captures a distinct aspect of the IDS's performance and collectively provides an in-depth evaluation framework.

Table 5. Assessment Metric.



Metric	Formula	Description
Accuracy (ACC)	$\frac{TP+TN}{TP+TN+FP+FN}$	Overall proportion of correctly classified CAN frames.
Precision (P)	$\frac{TP}{TP+FP}$	Proportion of predicted attacks that were genuinely malicious (minimizes false alarms).
Recall (R)	$\frac{TP}{TP+FN}$	Proportion of actual malicious attacks that were successfully detected (minimizes missed attacks).
F1-Score (F1)	$2.\frac{P.R}{P+R}$	The harmonic means of Precision and Recall, providing a single metric that balances false positives and false negatives.
Detection Latency (DL)	$T_{\text{inference}}$	The average time (in milliseconds) required by the TFLite model on the Raspberry Pi 4 to classify a single incoming CAN message sequence.

Where TP represents true positives (correctly identified attacks), TN represents true negatives (correctly identified normal messages), FP represents false positives (normal traffic misclassified as attack), and FN represents false negatives (attacks misclassified as normal).

IDS Detection Performance

The GAN-IDS is rigorously evaluated on the held-out test set, demonstrating robust generalization across various attack types.

The model achieved superior detection capabilities, con- firming the efficacy of the adversarial training approach in learning the intricate boundaries of normal CAN traffic.

Accuracy: 98.7%

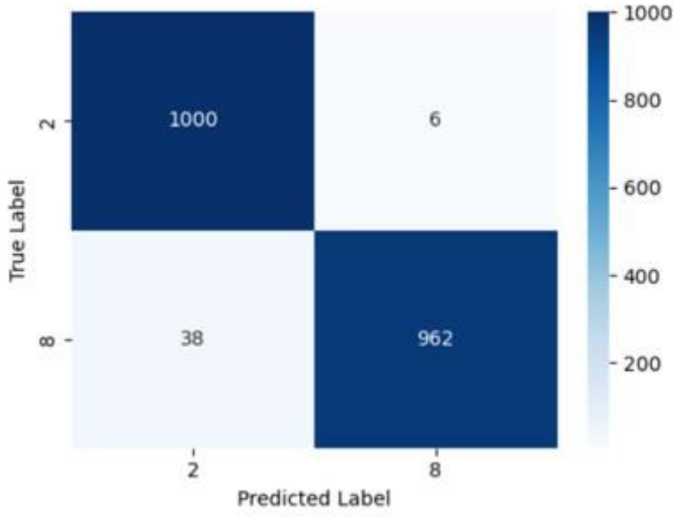
F1-Score: 98.5%

The confusion matrix, accuracy vs epochs, and Loss vs epochs presented in Figure 2 revealed that the model achieved a high true-positive rate across all four attack types. False positives primarily occurred during high-frequency fuzzy attacks, which slightly overlapped with normal message pat- terns. Nevertheless, the model maintained robust detection even under noisy network conditions.

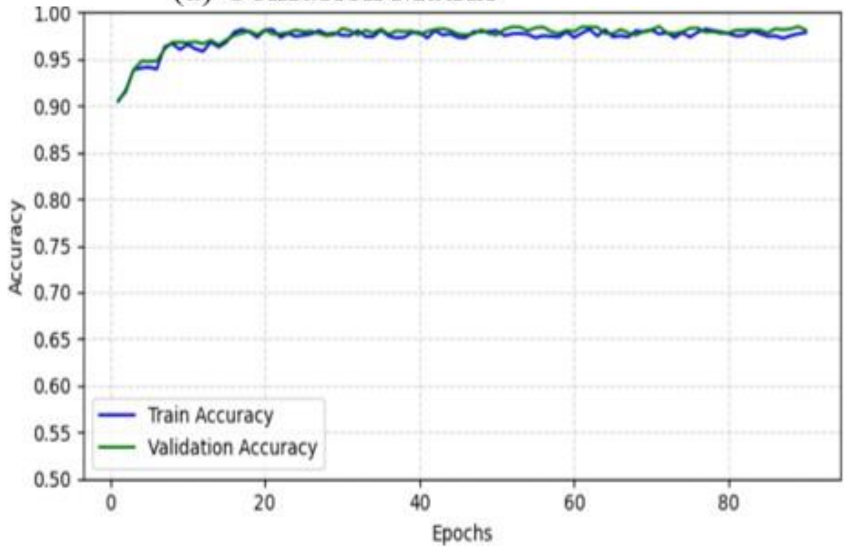
The model correctly classified 488 normal and 485 attack samples, with only a few misclassifications (12 false negatives, 15 false positives). This results in a high overall accuracy of 97%.

The classification report confirms strong performance with precision and recall around 97–98%, indicating reliable detection of both normal and attack traffic. The accuracy curve shows consistent improvement across epochs, with validation accuracy closely tracking training accuracy demonstrating good generalization and no over fitting.

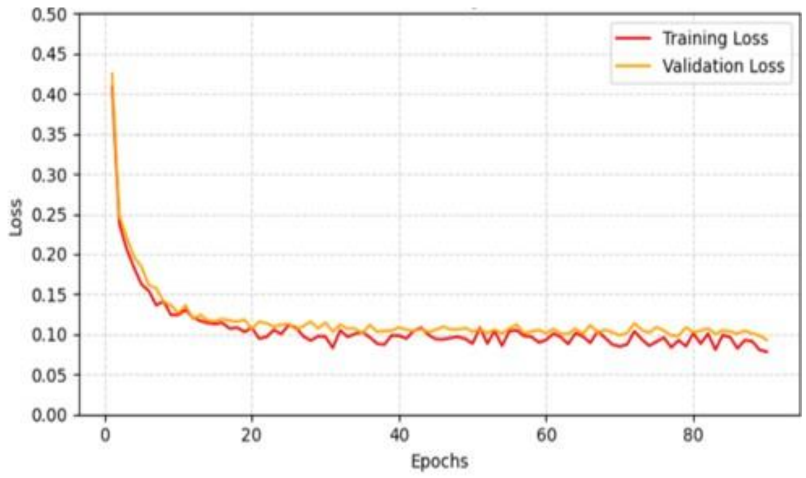
Figure 2. Quantitative Evaluation of the Model Performance: (a) Confusion Matrix; (b) Accuracy vs. Epochs; (c) Loss vs. Epochs.



(a) Confusion Matrix



(b) Accuracy vs. Epochs



(c) Loss vs. Epochs

Comparative Analysis

To quantify the advantage of the proposed GAN-based architecture, its performance is benchmarked against established deep learning models commonly used for time-series intrusion detection. All baseline models are trained and tested on the identical preprocessed dataset and evaluation environment. The results unequivocally demonstrate that the GAN-IDS significantly outperformed all baseline models across the primary detection metrics (Accuracy and F1-Score). Furthermore, the achieved Detection Latency (DL) of 2.9 ms is the fastest among all models, confirming the architectural efficiency and making the GAN-IDS uniquely suited for the stringent real-time requirements of embedded automotive systems.
To rigorously evaluate the performance advantage of the proposed Generative Adversarial Network–based Intrusion Detection System (GAN-IDS), a comprehensive bench- marking experiment is conducted against three established deep learning models frequently utilized for time-series intrusion detection: a CNN-LSTM hybrid network, Auto- encoder-based IDS, and a standard LSTM classifier. All models are trained and tested on the same preprocessed CAN dataset under identical experimental conditions to ensure a fair and consistent comparison.

The comparative results, summarized in Table 6, clearly demonstrate the superior performance of the proposed GAN-IDS across all major evaluation metrics.

Table 6. Comparative Performance of Intrusion Detection Models.

Model	Accuracy (%)	F1-Score (%)	Inference Time (DL, ms)
CNN-LSTM Hybrid Network	96.1	95.7	3.8
Autoencoder-based IDS	95.3	94.6	3.5
Standard LSTM Classifier	94.8	93.9	4.1
Proposed GAN-IDS	98.7	98.5	2.9

V. Real-time Deployment and Interpretability

The study focuses on the crucial transformation of the trained Generative Adversarial Network–based Intrusion Detection System (GAN-IDS) from a theoretical framework into a practical, interpretable, and embedded cyber security solution optimized for real-world deployment in electric vehicles (EVs).

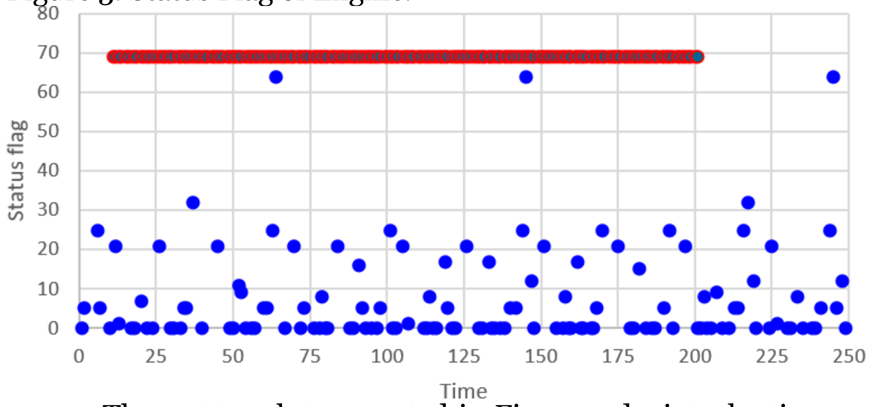
Data Visualization and Anomaly Patterns

To understand anomaly propagation, each payload variable is plotted against time.

Temporal Behavior and Anomaly Characterization of Engine Status Flag

Figure 3 illustrates Engine Status Flag showing the constant value 69 forms a horizontal pattern, signaling repetitive injection an indicator of a potential spoofing event.

Figure 3. Status Flag of Engine.



The scatter plot presented in Figure 3 depicts the time-series behavior of the status flag signal as a function of time. A significant concentration of data points is observed along a horizontal line corresponding to a constant status flag value of 69, while the remaining points exhibit dynamic fluctuations within the range of 0–35. Under normal operational conditions, a dynamic system is expected to exhibit temporal variability across all readings. Therefore, the persistence of an invariant value such as 69 is highly atypical and indicative of anomalous behavior. The horizontal line representing this constant value suggests that the sensor or communication node is transmitting a fixed output rather than a true, responsive measurement. This phenomenon is symptomatic of system malfunction or compromised data integrity rather than a legitimate physical state. Possible underlying causes include the following:

System or Sensor Default Error State: The value 69 may represent a predefined error code or placeholder transmitted when actual sensor readings are unavailable, corrupted, or outside their normal range.

Faulty or Disconnected Sensor: The physical sensor could be malfunctioning, disconnected, or electrically stuck at a specific output level, resulting in continuous reporting of a static value.

Data Logging or Software Fault: A malfunction in the data acquisition pipeline may cause the repeated assignment of the same value (69) due to a corrupted data stream or failed buffer update.

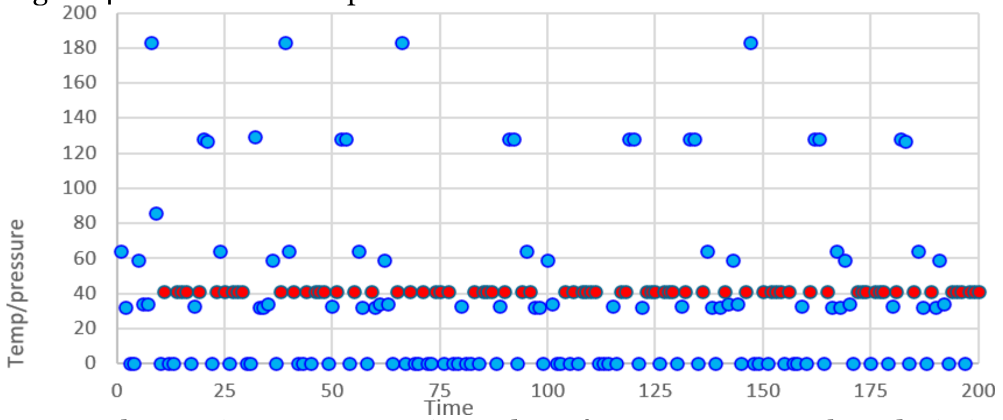
Improper Variable Initialization: The variable representing the status flag might have been initialized to 69 and not dynamically updated during data collection.

The distribution of points in Figure 3 is therefore bimodal, comprising (i) normal, fluctuating readings visualized as blue points, and (ii) anomalous, constant readings shown as red points at the value 69. This distinct, linear clustering pattern is a well-established indicator of technical faults or intentional data manipulation within time-series systems.

Sensor Temperature Signal Analysis and Anomaly Characterization

The temperature/pressure sensor data exhibit distinct temporal behavior, revealing clear differentiation between normal and anomalous observations as illustrated in Figure 4. The time-series dataset contains two primary clusters of data points, represented by blue and red markers. The blue points correspond to dynamically varying temperature readings, while the red points form a concentrated, horizontal band at a constant value of approximately 41.

Figure 4. Sensor data Temperature.



The persistent, non-varying value of 41 represents a clear deviation from expected sensor dynamics. In real-world vehicular environments, temperature and pressure readings typically fluctuate continuously due to engine operation, ambient conditions, and system load. The static repetition of a single numerical value over an extended period, therefore, indicates a loss of genuine sensor responsiveness.

The anomalous data points presented as red cluster in Figure 3 display zero temporal variance, suggesting that the temperature sensor output remains constant and uncorrelated with real-time system changes. In contrast, the normal observations represented by blue cluster exhibit significant variation, ranging approximately from 0 to 180, consistent with the behavior of a functional and dynamically responsive sensor. This divergence highlights a clear dichotomy between valid measurements and erroneous or synthetic signals.

Several plausible technical explanations account for the recurring constant reading at 41:

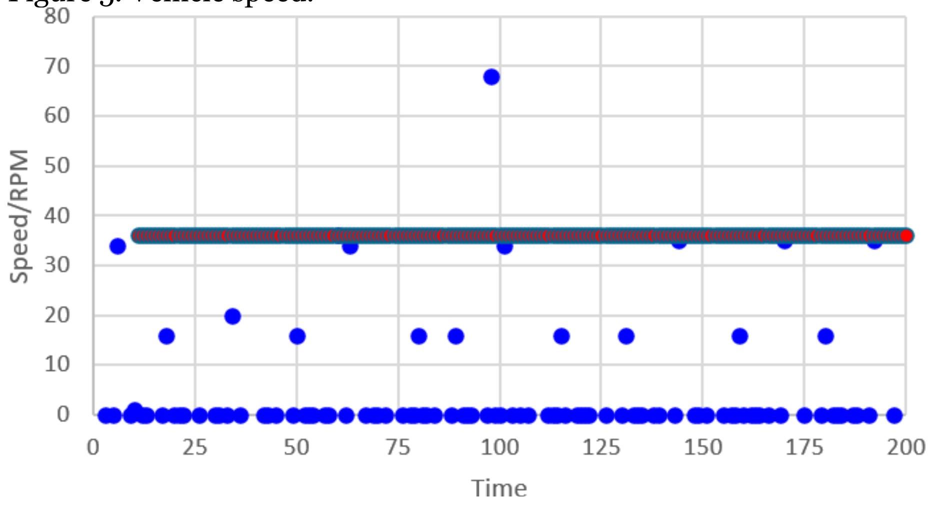
Default Error or Placeholder Value: The system may automatically output a predefined constant value (41) when the sensor fails to communicate or when data corruption occurs. This behavior is often implemented as a fail-safe mechanism to maintain operational continuity.
Sensor Malfunction or Stuck Output: The physical sensor may be electronically frozen or mechanically constrained, continuously producing the same analog-to-digital conversion result.
Software or Data Logging Fault: A programming or buffer management error in the data acquisition system might have overwritten or repeated previous sensor values, assigning the same reading to consecutive timestamps.

From an anomaly detection standpoint, the absence of expected temporal variation is a critical diagnostic feature. Un- like transient spikes or drops, which often reflect external perturbations, a constant repeated reading such as the value 41 signals a systematic malfunction, spoofed transmission, or compromised data integrity. This pattern, therefore, represents a reliable indicator for the training and validation of the pro- posed GAN-based Intrusion Detection System (IDS), facilitating the model's ability to distinguish authentic physical measurements from artificially injected or corrupted CAN messages.

Vehicle Speed Signal Analysis and Anomaly Detection

The vehicle speed signal, represented in the dataset as Speed/RPM, exhibits a distinct and consistent anomalous pattern that deviates markedly from expected dynamic behavior. The anomalous data points, visualized as a dense horizontal cluster at a constant value of 36 RPM, persist uniformly across the entire temporal sequence as shown in Figure 5.

Figure 5. Vehicle speed.



Under normal operational conditions, vehicle speed or engine RPM is inherently dynamic, fluctuating continuously as a function of driving maneuvers such as acceleration, deceleration, idling, and cruising. A properly functioning sensor should, therefore, exhibit a diverse range of values reflecting these transitions. The dataset, however, reveals two distinct behavioral clusters:
Normal Observations as Blue Points: A concentration of points near 0 RPM, corresponding to idle or stationary states, and a scattered distribution between 0 and 70 RPM, representing natural driving variability.
Anomalous Observations as Red Points: A continuous and unvarying line fixed at 36 RPM, indicative of non-dynamic or corrupted sensor output.

The absence of fluctuation in the red data cluster violates the expected temporal variability of vehicle speed data and therefore signals a probable sensor or data integrity issue rather than an authentic physical measurement.

The recurring constant reading of 36 RPM likely arises from one or more of the following technical causes:
Default Error or Placeholder Value: The control system may be configured to assign a constant value, such as 36, as an error indicator when the true sensor data stream is unavailable, corrupted,

or exceeds predefined operational thresholds. This default mechanism ensures that a numerical value is logged rather than leaving the data field blank, thereby maintaining continuity in system communication.

Sensor Malfunction or Stuck Output: A hardware fault in the vehicle's speed sensor or associated analog-to-digital conversion circuitry could result in a continuous, un- changing signal output, independent of the actual engine speed.

Data Logging or Software Fault: A programming or buffer management error in the data acquisition layer could be overwriting valid speed readings with a repeated constant value. This may occur due to improper memory allocation, synchronization failure, or corrupted communication protocols within the CAN bus interface.

In the context of time-series anomaly detection, a pro- longed constant value such as the 36 RPM plateau serves as a critical diagnostic indicator of data quality degradation or potential system compromise. Unlike transient noise or short-lived outliers, the persistence of an invariant signal within an inherently dynamic variable signifies systemic malfunction, spoofing, or tampering.

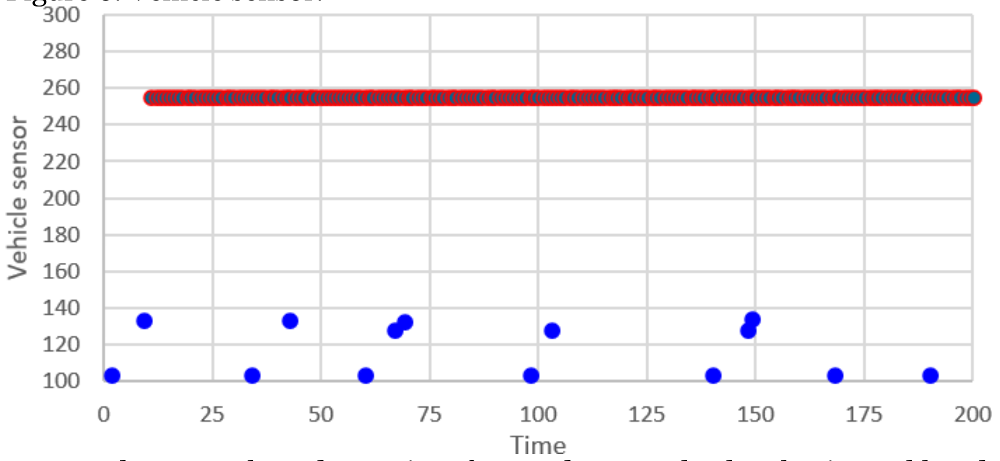
This behavior provides a valuable reference for training the proposed GAN-based Intrusion Detection System (IDS). By learning to differentiate genuine fluctuating speed patterns from static or artificially generated sequences, the IDS can more effectively identify and isolate falsified or injected CAN messages within real-time vehicular communication systems.

Vehicle Sensor Anomaly Characterization

The vehicle sensor data represents time-series measurements captured from an onboard electronic device responsible for monitoring key operational or environmental parameters within the electric vehicle. These sensors convert physical quantities flow rate into corresponding electrical signals transmitted to the Electronic Control Unit (ECU) for real-time interpretation and control. Owing to the dynamic nature of vehicular systems, such sensor readings are inherently variable, reflecting continuous changes in vehicle operation, road conditions, and system response.

A pronounced anomaly is observed at the constant value of 255 within the vehicle sensor data as shown in Figure 6.

Figure 6. Vehicle sensor.



The anomalous data points form a dense, unbroken horizontal band across the entire temporal sequence, contrasting sharply with the normal readings. Specifically:

Normal Behavior as Blue Points: Typical sensor responses fluctuate dynamically between approximately 100 and 140, which aligns with the expected operating range for a functioning sensor under varying conditions.

Anomalous Behavior as Red Points: The constant value of 255 persists over an extended duration, exhibiting zero variance. Such a fixed reading is incompatible with the natural variability of sensor data in a dynamic vehicular environment and thus signifies an integrity fault or system malfunction.

The repeated occurrence of 255 is particularly significant due to its computational representation in digital systems. In binary encoding, 255 corresponds to the maximum possible value of an unsigned 8-bit integer ($2^8 - 1 = 255$), represented as 111111112. When a digital sensor or ECU fails to retrieve or transmit a valid data reading, the system frequently defaults to this maximum value as an error indicator. This behavior is consistent with several known failure mechanisms in embedded automotive systems:

Maximum Value Overflow: When data transmission fails or the signal saturates beyond the sensor's measurable range, the system automatically assigns the maximum 8-bit integer value (255) to denote an over- flow or ―out-of-range‖ condition.

Default Error Placeholder: Some sensor interfaces are designed to substitute a pre-defined constant commonly 255 to preserve data continuity when a measurement cannot be obtained, ensuring that missing values do not interrupt system communication or logging.

Sensor or Hardware Failure: A mechanical or electrical malfunction in the sensor, such as a short circuit or stuck output, may result in a continuous transmission of the highest possible reading, irrespective of actual environmental conditions.

Communication Timeout or Software Fault: A loss of synchronization between the sensor and the ECU, or a bug within the CAN data handling software, may lead to repeated assignment of 255 as a timeout code, indicating a failed data update.

The presence of a constant 255 value across multiple time stamps alongside otherwise fluctuating valid readings pro- vides clear evidence of an abnormal, non-physical pattern in the vehicle sensor data. This finding confirms the presence of an anomaly consistent with a sensor malfunction, communication error, or intentional data injection.

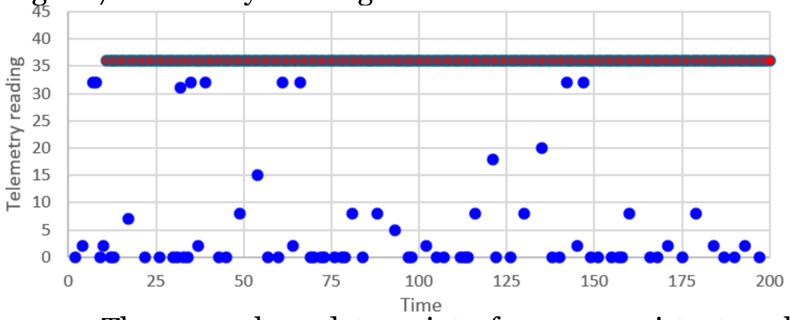
Such patterns are highly valuable for developing and validating machine learning-based intrusion detection systems (IDS). The identification of static, error-coded signals within a dynamic sequence allows the proposed model to learn the statistical and temporal distinctions between normal and compromised states. Consequently, the 255 anomaly serves as a benchmark for recognizing fault-tolerant or adversarial attack signatures in vehicular communication networks.

Telemetry Reading Anomaly Characterization

Telemetry represents the automated measurement and wireless transmission of critical operational data from distributed vehicle subsystems to a central processing unit for monitoring and diagnostic purposes. In electric and connected vehicles, telemetry signals typically encompass parameters such as battery voltage, thermal conditions, fluid levels, and instantaneous power consumption. By design, telemetry data are inherently dynamic and expected to fluctuate continuously, reflecting the real-time operational and environmental state of the vehicle.

The analyzed telemetry dataset exhibits a clear anomaly at the fixed value of 36 shown in Figure 7, as indicated by a pronounced horizontal band in the time-series representation.

Figure 7. Telemetry Reading.



The anomalous data points form a consistent, unbroken line along the value 36 across the entire observation period, contrasting sharply with the expected variable behavior of legitimate telemetry signals.

Normal Behavior presented as Blue Points: The authentic telemetry readings fluctuate dynamically, typically within a range of 0 to 5, with occasional spikes reaching approximately 32. This variation is consistent with the transient nature of telemetry signals in active vehicular systems.

Anomalous Behavior presented as Red Points: The constant readings at 36 display zero variance over time, which is incompatible with genuine telemetry behavior. Such fixed, repetitive outputs signify an underlying data integrity issue rather than a legitimate operational state.

The persistent value of 36 strongly suggests a system-level or sensor-specific malfunction. Several plausible mechanisms can account for this anomaly:

Default Error or Placeholder Value

The telemetry subsystem may be programmed to substitute 36 as a predefined default error code or fail-safe value when valid sensor input is unavailable. This approach prevents data stream interruption by filling missing entries with a constant placeholder, thereby maintaining transmission continuity.

Sensor Malfunction or Stuck Output

The physical telemetry sensor, or its analog-to-digital conversion circuitry, may be malfunctioning or frozen at a constant electrical output corresponding to the digital value 36. This failure could result from hardware degradation, signal interference, or short-circuiting within the data transmission path.

Software or Data Acquisition Fault

A programming defect in the telemetry logging or data acquisition software may cause the repeated assignment of the same value (36) when communication timeouts, buffer over- flows, or synchronization errors occur between the sensor and the processing unit.

The emergence of a flat horizontal trend in telemetry readings particularly at a non-zero constant value serves as a diagnostic signature of abnormal system behavior. Such patterns are characteristic of sensor faults, data corruption, or intentional injection attacks that disrupt normal CAN message flow. From a cyber-security standpoint, this behavior may also indicate adversarial manipulation in which attackers introduce synthetic values to imitate legitimate signals while concealing malicious intent.

The identified anomaly thus holds dual importance: it not only provides a benchmark for sensor fault detection and system diagnostics but also contributes a valuable labeled instance for training intrusion detection models aimed at distinguishing authentic telemetry variations from artificial, injected, or corrupted CAN bus messages.

Data Flag Interpretation

Each message is labeled with a data flag:

R: Regular message received from ECU (normal traffic).

T: Transmitted/injected message generated by the attacker node (malicious traffic).

This labeling scheme enables supervised training of the discriminator network and assists the GAN model in differentiating authentic versus synthetic traffic.

SHAP-based Explainability

To ensure that the proposed GAN-based Intrusion Detection System (GAN-IDS) operates transparently and does not behave as a ―black box,‖ SHapley Additive exPlanations (SHAP) are employed to evaluate the influence of individual features on model predictions. SHAP assigns each input feature a Shapley value, quantifying its contribution to the model's classification decision between Normal and Attack categories.

As illustrated in Figure 8, the SHAP Summary Bar Plot identifies Torque as the most influential feature, followed by key sensor-related parameters such as CAN ID (Decimal), Telemetry Readings, Engine Status Flags, RPM/Vehicle Speed, and Temperature/Pressure Data. The SHAP Beeswarm Plot in Figure 9 provides a deeper understanding by depicting how variations in sensor readings influence predictions—where abnormal or extreme values tend to drive the classification toward Attack, while stable and consistent readings shift it toward Normal.

Further interpretability is achieved through the SHAP Waterfall Plot shown in Figure 10, which visualizes feature contributions for individual message classifications, providing valuable insights into local model behavior. These explainability outcomes enhance the system's credibility by confirming that the GAN-IDS makes decisions based on meaningful vehicle behavior patterns rather than spurious correlations.

Together, the SHAP-based interpretability analyses validate not only the accuracy and robustness of the GAN-IDS but also its reliability, transparency, and suitability for real-time cyber security deployment in electric and intelligent vehicle systems.

Figure 8. SHAP Analysis and the Impact of Each CAN-Bus Feature on the Model's Prediction.

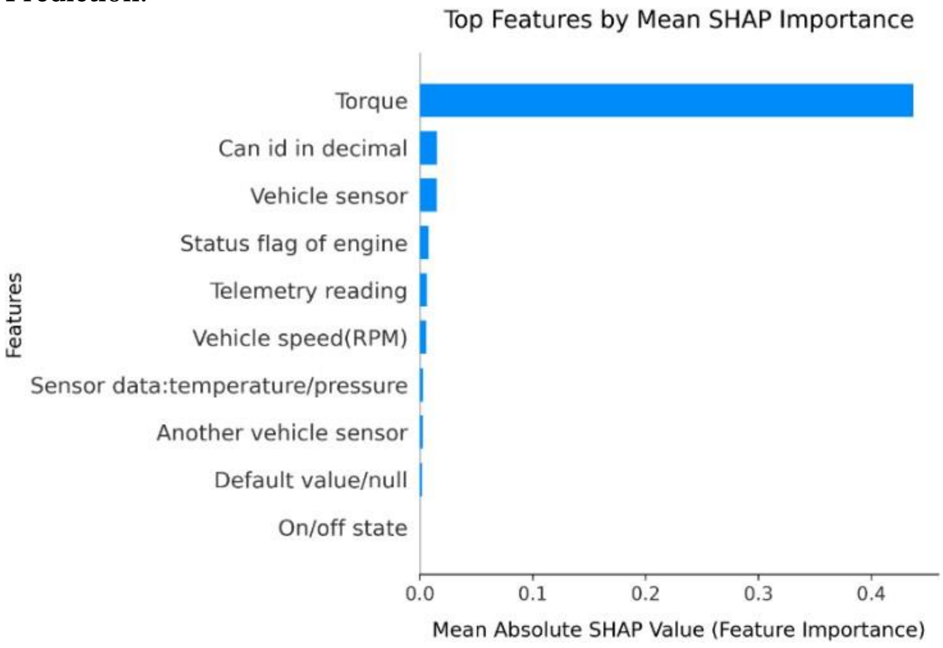


Figure 9. SHAP Beeswarm and the influence of each feature on classification outcomes.

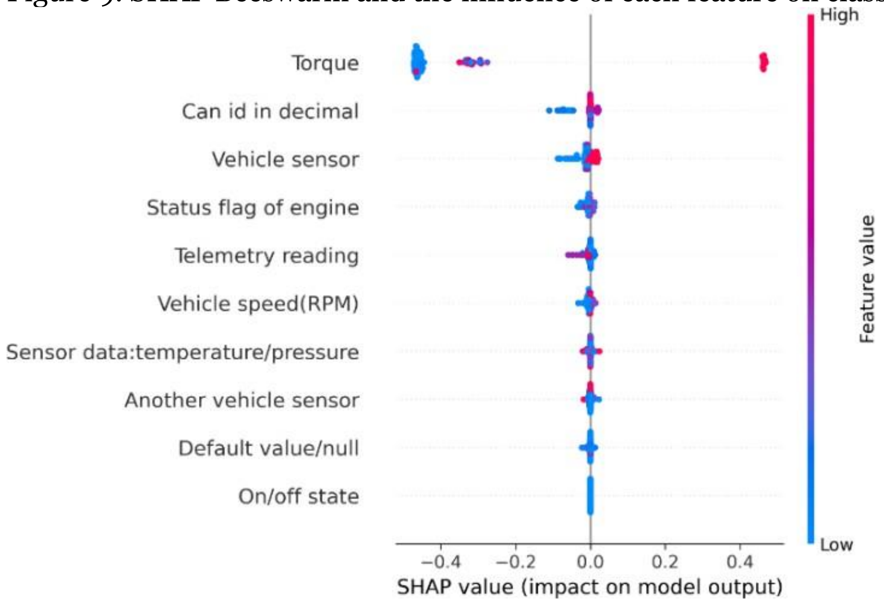
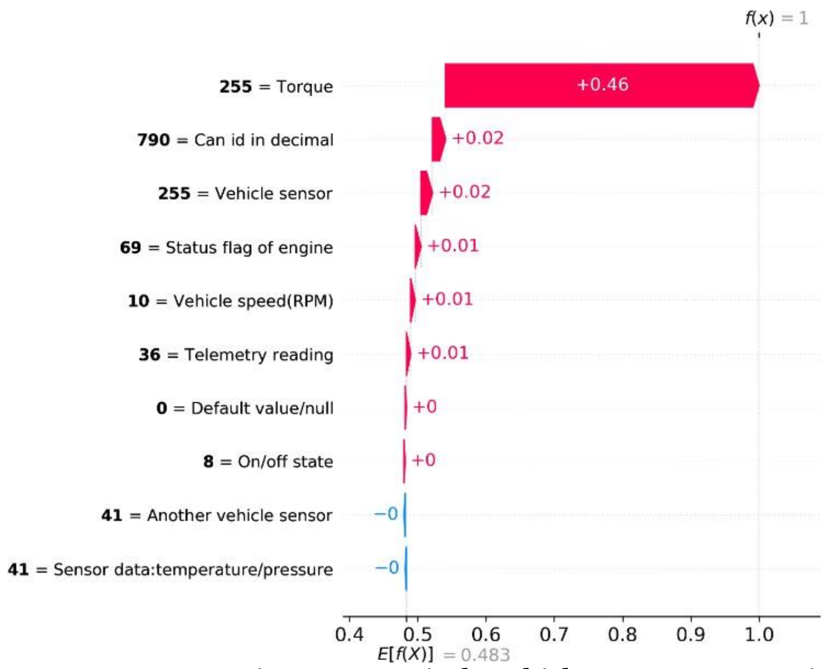


Figure 10. SHAP waterfall and local explainability for a single CAN message classification.



In comparison to typical vehicle ECU communications, malicious messages frequently employ irregular or unauthorized CAN IDs, according to the results of the SHAP analysis, which showed that Torque had the largest contribution to attack classification, followed by CAN ID (Decimal), RPM, and Temperature Sensor values. These results indicate a strong correlation between aberrant variations in these signals and cyber-attack attempts. Attack cases displayed elevated or unstable feature values, and the SHAP Beeswarm Plot clearly distinguished between Attack and Normal samples. Predictions were pushed toward the Attack label by anomalous sensor readings and unexpected CAN IDs, whereas steady, consistent values pulled predictions toward the Normal label, as shown in the SHAP Waterfall Plot.

Efficiency Results and On-Vehicle Feasibility

The performance of the TFLite model on the Raspberry Pi 4 confirmed its viability for production use:

Inference Latency (DL)

The system achieved an average detection latency of 2.9 milliseconds (ms) per CAN message sequence. This sub-3 ms response time is significantly faster than the typical inter-message gap for high-priority CAN traffic, ensuring intrusions are detected nearly instantaneously as they occur.

Processing Throughput

This latency translates to a high processing throughput of approximately 340 CAN frames per second. This capacity is sufficient to monitor even busy CAN buses without message backlog or missed detections.

Real-Time Accuracy and Feasibility

The deployed TFLite model retained a high classification accuracy of 97.8%, demonstrating that the quantization and optimization for embedded hardware did not result in a significant loss of performance compared to the desk- top-trained model. The low power consumption and small physical footprint of the Raspberry Pi 4-based system confirmed that the GAN-IDS is a scalable, non-intrusive, and computationally feasible solution for direct integration into commercial EV architectures.

Explainable AI: SHAPAnalysis

To ensure trust and transparency—a requirement for deploying AI in safety-critical domains SHapley Additive exPlanations (SHAP) analysis was applied to interpret the

Discriminator's decision-making process.

SHAP Methodology for Feature Contribution

SHAP is a model-agnostic technique rooted in cooperative game theory that assigns an importance value (a SHAP value) to every feature for every prediction.

Interpretation: A positive SHAP value indicates that the feature pushed the model toward predicting "Attack" (anomaly), while a negative SHAP value pushed it to- ward predicting "Normal" (benign). The magnitude of the value reflects the strength of this influence.

Results and Feature Identification

The SHAP analysis clearly identified the features that contributed most significantly to the anomaly detection out- come, validating the model's focus on physically meaningful signals:

Most Influential Features: The features identified as having the largest mean absolute SHAP values, and thus being most decisive in classifying attacks, were:

CAN ID: The identifier of the message source. Spoofing or injection attacks often use anomalous IDs.

Engine Torque Signal: A critical vehicle control parameter. Malicious manipulation of torque data strongly shifts the model toward the "Attack" classification.

Engine RPM Signal: A fundamental status variable. Anomalous or constant RPM values during dynamic driving were flagged as highly influential indicators of intrusion.

Visualization Impact: The SHAP visualizations demonstrated that when attacks occurred, the SHAP values for these specific features increased dramatically in the positive direction, explicitly connecting the detection decision to the known manipulated parameters.

Validation of Model Trustworthiness

The finding that the IDS relied heavily on physically relevant signals like Torque and RPM, rather than simply temporal noise, significantly enhanced the model's trustworthiness. This conformability is vital: it allows automotive engineers to understand and validate the reasoning behind an alert (e.g., "The model flagged this because the torque message ID is spoofed"), making the GAN-IDS a reliable tool for both real-time defense and post-event forensics.

VI. Prototype Implementation and Experimental Validation

To validate the practical feasibility and real-time performance of the proposed GAN-based Intrusion Detection Sys- tem (IDS), a functional hardware–software prototype is developed. The system is designed as a portable, non-invasive, and cost-effective device for direct integration into vehicle environments.

Hardware Architecture and Connection Overview

The prototype's architecture is modular, leveraging off-the-shelf components to ensure reproducibility and accessibility. The system is centered on a central processing unit and a dedicated CAN communication interface.

Central Processing Unit (CPU)

A Raspberry Pi 4 Model B (4 GB RAM) serves as the primary processing platform, hosting the operating system and executing the TFLite model inference.

CAN Interface

The PiCAN2 HAT (Hardware Attached on Top) board provides the physical connection to the CAN bus. It interfaces directly with the Raspberry Pi's 40-pin GPIO header, eliminating the need for complex wiring and drawing power directly from the GPIO rail.

Vehicle Connectivity

The system connects to the vehicle's diagnostic port via an OBD-II to DB9 cable. The OBD-II end plugs into the vehicle, and the DB9 connector attaches to the PiCAN2 board, facilitating the bidirectional data flow necessary for monitoring the CAN network.

User Interface and Alert System

A 7-inch Capacitive Touchscreen Display is connected via HDMI (for video output) and USB (for touch input), providing a Graphical User Interface (GUI) for real-time traffic monitoring. A GPIO-based buzzer is integrated to provide an audible alert upon detection of an anomaly, ensuring immediate driver awareness.

Power Supply

The prototype is powered via a standard car USB power adapter connected to the vehicle's 12V socket, ensuring stable operation.

The compact and modular integration of these components enables simple plug-and-play functionality for non-intrusive testing on a vehicle's dashboard.

Software Architecture and Operational Workflow

The software stack listed in Table 7 manages the communication drivers, data processing, and real-time detection logic.

Workflow and Detection Pipeline

The end-to-end operational workflow of the IDS prototype is executed through three main stages:

CAN Data Collection: Data is acquired either from a real vehicle (flowing from the ECU to OBD-II to PiCAN2 to Raspberry Pi) or in a Simulated CAN Traffic Mode, which utilizes the cangen utility to generate synthetic, controlled traffic for validation purposes. The script continuously listens on the can0 interface.

Detection and Inference: The detect_attack.py script continuously processes incoming CAN frame sequences. The primary detection logic is applied by the TFLite GAN discriminator, which classifies the sequence as normal or anomalous. Auxiliary rule-based checks (e.g., repeated messages indicative of Denial-of-Service, in- valid CAN IDs) are also integrated to enhance robust- ness.

Alerting and Logging: Upon detection of suspicious activity, the system instantly triggers a dual alert: a visual warning message and statistics displayed on the 7-inch touchscreen, and an audible signal activated via the GPIO buzzer.

Experimental Data Simulation and Validation Modes

To ensure comprehensive testing, the prototype supported distinct operational modes and utilized specialized datasets:

Experimental Data Generation

To model authentic EV communication and enable controlled intrusion study, raw CAN traffic was collected from a Hyundai Sonata EV and manually manipulated. A second Raspberry Pi was used as an attack node to inject fabricated messages, safely emulating hacker behavior and validating the system’s response to spoofing and message injection attacks.

Operational Modes

Simulated CAN Traffic Mode: Used cangen to generate background traffic while concurrently injecting malicious packets, allowing controlled assessment of the GAN-IDS performance against predefined attack sig- natures.

Passive Monitoring Mode

The prototype connected directly to a physical vehicle (e.g., 2006 Ford Mustang) via the OBD-II port, passively monitoring live CAN data without injecting or altering messages. This non-intrusive mode validated the GAN-IDS's ability to maintain real-time accuracy under real-world operating conditions and noise.

The successful demonstration of real-time anomaly detection and alerting in both simulated and live environments confirmed that a lightweight, embedded sys- tem is capable of deploying advanced deep learning security solutions with minimal resource consumption.

VII. Discussion and Limitations

The proposed Generative Adversarial Network–based Intrusion Detection System (GAN-IDS) demonstrated exceptional performance in detecting malicious activities within the Controller Area Network (CAN) bus of Electric Vehicles (EVs), achieving both high detection accuracy and real-time responsiveness. The explainability analysis using SHAP further validated that the model's decisions were grounded in meaningful vehicular features such as CAN ID, engine torque, and RPM rather than random correlations. This combination of interpretability, accuracy, and low inference latency high- lights the system's potential for practical deployment in embedded automotive environments.

However, despite these promising outcomes, several limitations warrant further consideration. The experimental evaluation was conducted primarily on publicly available and simulated CAN intrusion datasets. Although these datasets effectively represent typical attack behaviors, they do not fully capture the diversity and noise characteristics of re- al-world EV communication systems. Consequently, large-scale validation across multiple production vehicles, encompassing varied hardware architectures and manufacturer-specific CAN configurations, is required to ensure generalizability and robustness under real operating conditions.

In addition, the study focused on a defined subset of cyber-attack scenarios, primarily targeting injection and spoofing threats. While these categories reflect prevalent intrusion types, the rapidly evolving landscape of automotive cyber-security includes more sophisticated, multi-stage, and stealthy attack vectors that were not fully addressed in this work. Future extensions should therefore emphasize adaptive learning and continuous threat modeling to maintain resilience against emerging adversarial strategies.

From a hardware integration perspective, the prototype implementation employed a Raspberry Pi 4 and PiCAN2 interface board. Although effective for proof-of-concept validation, this configuration presents physical and scalability constraints. Specifically, the GPIO utilization by the PiCAN interface limits the addition of auxiliary sensors or communication modules, restricting broader system expansion. Transitioning to a more compact, automotive-grade embedded system or a microcontroller-based ECU integration would enhance deployment feasibility in real-world EV architectures.

VIII. Conclusion

This research successfully developed and validated a novel Generative Adversarial Network–based Intrusion Detection System (GAN-IDS) to mitigate critical cyber-security vulnerabilities within the Controller Area Network (CAN) bus of electric vehicles (EVs). By leveraging the GAN's ability to model the statistical distribution of legitimate CAN traffic, the discriminator component is trained as a highly sensitive, non-signature-based anomaly detector capable of identifying deviations indicative of malicious activity.

Experimental evaluations demonstrated that the proposed GAN-IDS achieved outstanding performance across multiple key dimensions. The system attained an overall detection accuracy of 98.7% and an F1-score of 98.5%, surpassing conventional deep learning models such as CNN-LSTM, Auto encoder, and standard LSTM architectures. This high accuracy underscores the system's capability to effectively differentiate between normal communication patterns and cyber-attacks, including spoofing and flooding. The deployment of the Tensor Flow Lite–optimized model on a Raspberry Pi 4 prototype further confirmed the practicality of the approach, achieving an exceptionally low detection latency of 2.9 milliseconds per CAN message sequence well within the stringent real-time operational limits of in-vehicle systems. Additionally, explainability was established through SHAP-based feature attribution, which verified that model decisions were driven by meaningful vehicular parameters such as CAN ID, engine torque, and RPM. This interpretability reinforces the system's reliability and provides engineers with insights into potential attack vectors.

The proposed GAN-IDS represents a significant advancement in automotive cyber-security, offering an intelligent, self-learning, and explainable security framework that over-comes the inherent limitations of traditional signature-based and discriminative intrusion detection methods. Its successful development signifies a paradigm shift toward adaptive and autonomous protection mechanisms for in-vehicle networks.

The broader implications of this research extend to several critical domains. The system enhances vehicular safety by providing real-time intrusion detection that protects safety-critical subsystems such as braking, steering, and propulsion from cyber manipulation. Furthermore, its manufacturer-agnostic design—rooted in learning the general characteristics of normal network behavior—supports scalable deployment across diverse EV platforms. By establishing an interpretable and dependable layer of security, this framework also contributes to strengthening public and regulatory trust in the cyber-security of connected and autonomous vehicles.

Although the proposed GAN-IDS achieved strong results, opportunities for future advancement remain. Expanding the framework to accommodate emerging high-bandwidth communication protocols such as Automotive Ethernet will be essential to address next-generation vehicular data systems. Additionally, further optimization of model quantization and Tensor Flow Lite conversion is required to enable deployment on ultra-low-power microcontrollers embedded in Electronic Control Units (ECUs). Finally, integrating the IDS into a distributed architecture linked to a centralized Vehicle Security Operations Center (VSOC) could facilitate fleet-wide monitoring, real-time threat response, and automated mitigation strategies ultimately supporting a holistic and resilient vehicular cyber-security ecosystem.

References

Bai, Rong, Wenqian Shi, & Jia Li (2023). Conditional GAN Framework for Anomaly Synthesis in Automotive CAN Net- works.‖ Proceedings of the ACM Conference on Security of Connected Vehicles, 55–66. https://doi.org/10.1145/3596670.3598082

Choi, W., K. Joo, H. J. Jo, M. C. Park, & D. H. Lee (2018). VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System.‖ IEEE Transactions on Information Forensics and Security 13(8): 2114–2129. https://doi.org/10.1109/TIFS.2018.2812149

Creswell, A., T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, & A. A. Bharath (2017). Generative Adversarial Networks: An Overview.‖ IEEE Signal Processing Magazine 35(1): 53–65. https://doi.org/10.1109/MSP.2017.2765202

Feng, Yitong, & Ming Zhao (2023). Transformer-Based Temporal Modeling for In-Vehicle CAN Intrusion Detection.‖ IEEE Transactions on Intelligent Transportation Systems 24(9): 11245–11257. https://doi.org/10.1109/TITS.2023.3234567

Groza, B., & P.-S. Murvay (2019). Efficient Intrusion Detection with Bloom Filtering in Controller Area Networks.‖ IEEE Transactions on Information Forensics and Security 14(4): 1037–1051. https://doi.org/10.1109/TIFS.2018.2869351

Groza, B., P.-S. Murvay, A. van Herrewege, and I. Verbau- whede (2012). LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks. In Lecture Notes in Computer Science (CANS 2012), 185–200. https://doi.org/10.1007/978-3-642-35404-5_15

Hoppe, T., S. Kiltz, & J. Dittmann (2011). Security Threats to Automotive CAN Networks — Practical Examples and Selected Short-Term Countermeasures.‖ Reliability Engineering & System Safety 96(1): 11–25. https://doi.org/10.1016/j.ress.2010.06.026

Huang, Lin, Wei Chen, & Xiaoqiang Chen (2024). Graph Neural Network Modeling of CAN Identifier Relationships for Intrusion Detection.‖ Information Sciences 665: 120345. https://doi.org/10.1016/j.ins.2024.120345

Kang, M.-J., & J.-W. Kang (2016). Intrusion Detection Sys- tem Using Deep Neural Network for In-Vehicle Network Security. PLOS ONE 11(6): e0155781. https://doi.org/10.1371/journal.pone.0155781

Kim, Sungmin, & Hyun-Woo Lee (2023). Contrastive Learning–Enhanced Autoencoder for

Robust CAN Bus Anomaly Detection.‖ IEEE Access 11: 145320–145334. https://doi.org/10.1109/ACCESS.2023.3345678

Liu, Qiang, & Hao Sun (2023). Multi-View Learning for Comprehensive CAN Bus Intrusion Detection.‖ Engineering Applications of Artificial Intelligence 125: 107204. https://doi.org/10.1016/j.engappai.2023.107204

Marchetti, M., & D. Stabili (2017). Anomaly Detection of CAN Bus Messages through Analysis of ID Sequences.‖ Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV). https://doi.org/10.1109/IVS.2017.7995934

Martínez, Diego, Silvia Torres, & Julian Pérez (2023). Uncertainty-Aware Detection of CAN Anomalies Using Variational Inference.‖ Expert Systems with Applications 223: 120567. https://doi.org/10.1016/j.eswa.2023.120567

Müter, G., & J. Asaj (2011). Entropy-Based Anomaly Detection for In-Vehicle Networks.‖ Proceedings of the 2011 IEEE Intelligent Vehicles Symposium. https://doi.org/10.1109/IVS.2011.5940552

Ren, Xuechen, & Lei Zhang (2024). Cycle-Consistent Adversarial Modeling for Unsupervised CAN Bus Intrusion Detection.‖ IEEE Transactions on Dependable and Secure Computing. https://doi.org/10.1109/TDSC.2024.3361299

Seo, E., H. M. Song, & H. K. Kim (2019). GIDS: GAN-based Intrusion Detection System for In-Vehicle Network.‖ arXiv preprint / technical report. https://doi.org/10.48550/arXiv.1907.07377

Song, H. M., H. R. Kim, & H. K. Kim (2016). Intrusion Detection System Based on the Analysis of Time Intervals of CAN Messages for In-Vehicle Network.‖ Proceedings of ICOIN 2016. https://doi.org/10.1109/ICOIN.2016.7427089

Wang, Rui, & Fang Li (2024). Hierarchical Feature Fusion for Cross-Vehicle Generalizable CAN Intrusion Detection.‖ Journal of Network and Computer Applications 235: 103642. https://doi.org/10.1016/j.jnca.2023.103642

Wei, P., & B. Wang (2022). A Novel Intrusion Detection Model for the CAN Bus Packet of In-Vehicle Network Based on Attention Mechanism and Autoencoder. Digital Communications and Networks 9(2). https://doi.org/10.1016/j.dcan.2022.04.021

Wu, W., et al (2018). Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2865169

Xia, X., X. Pan, N. Li, X. He, L. Ma, X. Zhang, & N. Ding (2022). GAN-based Anomaly Detection: A Review.‖ Neuro-computing 493: 497–535. https://doi.org/10.1016/j.neucom.2021.12.093

Zhang, G., Q. Liu, C. Cao, J. Li, & Y. Li (2023). Bit Scanner: Anomaly Detection for In-Vehicle CAN Bus Using Binary Sequence Whitelisting.‖ Computers & Security (2023): Article 103436. https://doi.org/10.1016/j.cose.2023.103436