symmetry

MDPI

Article

A Stacked Machine Learning-Based Intrusion Detection System for Internal and External Networks in Smart Connected Vehicles

Xinlei Zhou [1], Yujing Wu [1,*], Junhao Lin [1], Yinan Xu [1] and Samuel Woo [2]

[1] Department of Electronic & Communication Engineering, Yanbian University, Yanji 133002, China; 2023050059@ybu.edu.cn (X.Z.); 1234024514@ybu.edu.cn (J.L.); ynxu@ybu.edu.cn (Y.X.)
[2] The Department of Software Science, Dankook University, Jukjeon 16890, Republic of Korea; samuelwoo@dankook.ac.kr
* Correspondence: yjwu@ybu.edu.cn

Abstract: In response to the escalating threat of cyberattacks on smart connected vehicles, numerous Intrusion Detection Systems (IDSs) have emerged. However, existing IDSs often prioritize enhancing detection accuracy while overlooking the time needed for training and detection. Moreover, they may not fully leverage the combined utilization of CAN bus IDs and the data field with external network data. Consequently, these systems frequently struggle to meet the real-time demands and broader attack scenarios inherent in in-vehicle systems. To overcome these challenges, we propose a stacked-model IDS architecture deployed across the CAN bus and central gateway, capable of detecting both internal and external vehicular network attacks. The system extracts key features from in-vehicle and external network data, builds base learners (CART, LightGBM, XGBoost), and integrates them through stacking with a meta-learner. Feature selection and training efficiency are enhanced using information gain and maximal information coefficient algorithms. Experiments show that the proposed IDS achieves an average detection accuracy of 99.99% for internal CAN bus attacks and 99.81% for external network attacks, with fast detection times of 0.018 ms and 0.088 ms, respectively. These results highlight the system's real-time capability, high accuracy, and adaptability to complex attack scenarios.

Keywords: IDS; internal and external network; machine learning; smart connected vehicles; stacked model

check for updates

Academic Editor: Jie Yang

Received: 3 May 2025
Revised: 22 May 2025
Accepted: 29 May 2025
Published: 4 June 2025

Citation: Zhou, X.; Wu, Y.; Lin, J.; Xu, Y.; Woo, S. A Stacked Machine Learning-Based Intrusion Detection System for Internal and External Networks in Smart Connected Vehicles. *Symmetry* **2025**, *17*, 874. https://doi.org/10.3390/sym17060874

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Smart connected vehicles (SCVs) integrate internal (CAN bus, ECU-based) and external (V2X) networks, delivering advanced functionalities but introducing critical security vulnerabilities [1], as illustrated in Figure 1. The CAN bus's inherent lack of security (e.g., no authentication) exposes it to remote exploits, while V2X connectivity is susceptible to attacks, like Denial of Service (DoS), GPS spoofing, and brute-force [2,3]. As traditional security measures are ill-suited for vehicular constraints, Intrusion Detection Systems (IDSs) have become vital.

However, existing IDSs often fail to simultaneously address internal and external threats and typically prioritize detection accuracy over the crucial real-time performance (training and detection speed) required in vehicles. This paper proposes a novel stacked model-based IDS to overcome these limitations, providing comprehensive and rapid detection of diverse attacks (including DoS, fuzzing, spoofing, sniffing, and brute-force) across both internal and external vehicular networks. The system aims for high detection accuracy with minimal computational overhead, ensuring efficiency and real-time responsiveness.

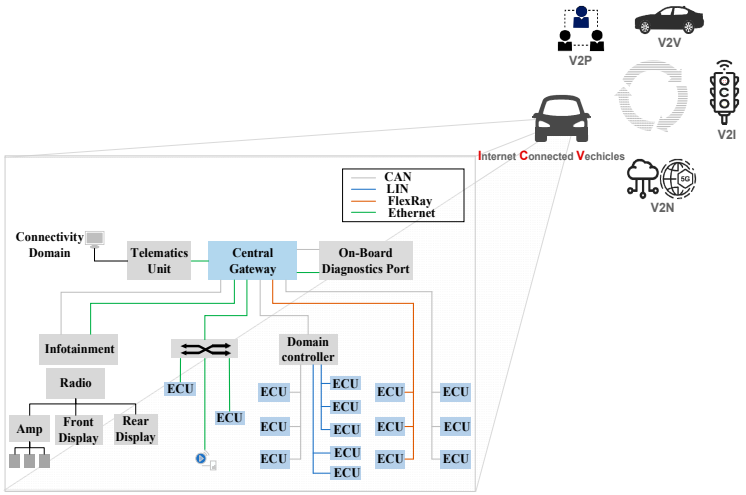


Figure 1. Internal and external vehicle network.

This paper further details vehicular network vulnerabilities and associated attack methods, the proposed stacked machine learning model for collaborative intrusion detection, and its performance validation using datasets encompassing various internal and external network attacks.

2. Introduction to Security Analysis and IDS for Smart Connected Vehicles

2.1. Vehicle Bus Internal Vulnerability Issues and Related Attacks

The bus technology of smart connected vehicles is the core of building the internal communication network of the vehicle. In this complex and sophisticated system, various sensors, controllers, and actuators need to communicate with each other to realize various functions of the vehicle, such as engine control, braking system, airbag, and entertainment system. In order to meet different communication needs, smart connected vehicles typically use a variety of bus technologies, including CAN, Local Interconnect Network (LIN), FlexRay, Media Oriented System Transmission (MOST), and Ethernet.

Among the many bus technologies, CAN bus was chosen as the main bus for smart connected vehicles because of its mature, reliable, efficient and cost-effective characteristics. It supports efficient and reliable data exchange between multiple ECUs, enabling smart connected vehicles to realize complex functions [4]. The security aspects of the CAN bus are usually described in terms of the identifier (ID), the data field, and the arbitration mechanism, so we will focus on these three components. The CAN bus 2.0 A standard format is shown in Figure 2.

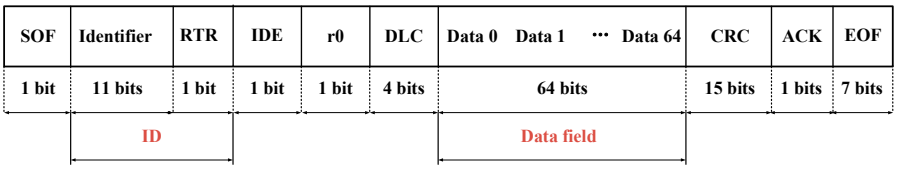


Figure 2. CAN 2.0 A bus frame structure.

ID: Used to determine the priority and sender of a message, there are two types of IDs: standard ID (11 bits) and extended ID (29 bits). The lower the ID value, the higher the priority. This means that high-priority messages can dominate the bus, ensuring the timely transmission of critical data.

Data field: The data length code (DLC) in the control segment dictates that the maximum data field length is 8 bytes. This data segment carries crucial ECU signals, their rate of change, and status information, like start, run, and stop states. Such data are vital for the braking system, as it enables precise prediction of the vehicle's dynamics and engine response. This, in turn, allows for the adjustment of braking force and strategy to maintain vehicle stability and safety during braking.

Arbitration mechanism: The internal communication process of the CAN bus is shown in Figure 3. The CAN bus uses an arbitration mechanism to manage message transmission among multiple nodes [5]. Nodes assess whether the bus is clear before sending messages by comparing their message's unique ID, which indicates priority. If a node's message ID is higher, it proceeds to transmission; otherwise, it waits for the bus to be free and retries.

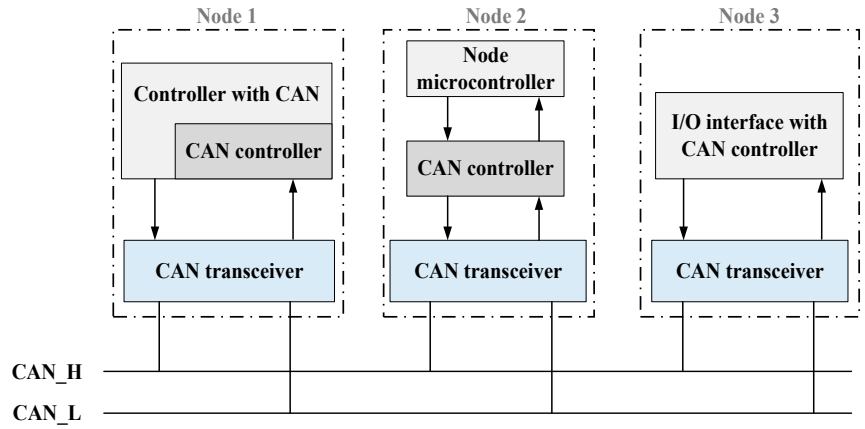


Figure 3. Multiple nodes sharing a bus in a CAN network.

The arbitration mechanism on the CAN bus ensures orderly message transmission, preventing conflicts and data loss. It prioritizes high-priority messages, ensuring timely delivery of critical information. This coordination enhances system efficiency and reliability, allowing the smart connected car's components to collaborate effectively and perform complex functions.

The CAN bus, central to smart connected vehicles, has inherent vulnerabilities due to its design. Firstly, its broadcast nature allows all devices to access all messages, reducing security and making it easier for attackers to intercept and alter data. Secondly, the lack of robust message prioritization can delay critical transmissions during network congestion, impacting system performance and stability. Lastly, the absence of a built-in error detection and re-transmission mechanism in the CAN protocol can lead to communication interruptions and reduced system reliability in the event of errors.

With the rapid advancement of telematics technology, these issues are becoming increasingly severe, posing greater security threats to the CAN bus system [6,7]. These attacks not only disrupt the normal operation of vehicles but also potentially endanger occupant safety. The following are several fundamental types of CAN bus attacks:

1. Message injection attack;

Attackers can compromise vehicle operations by injecting unauthorized CAN messages, either by directly accessing the bus or by exploiting other vulnerable vehicle systems [8].

This can lead to dangerous scenarios, such as: parking attacks: sending false brake commands to abruptly stop the vehicle, risking accidents; acceleration attacks: sends false acceleration commands, potentially resulting in unsafe driving or loss of control; steering attacks: sending false steering commands, causing the vehicle to veer off course or crash; door lock and window control attacks: sending commands to unlock doors or

control windows, increasing the risk of vehicle theft or intrusion [9,10]. These attacks demonstrate the serious threat to vehicle and occupant safety posed by the manipulation of CAN bus messages.

2. DoS attack;

A Denial of Service (DoS) attack on a vehicle's CAN bus deliberately disrupts its normal operation, affecting performance and security.

Common forms of DoS attacks include: flooding attacks: attackers overwhelm the CAN bus with invalid packets, consuming resources and blocking legitimate traffic [11]; error message injection: false error messages are sent to trick the system into halting operations due to perceived faults [12]; traffic overload: excessive packets or requests are sent, overwhelming the bus and causing delays and system failure [13]; continuous interference: persistent sending of malicious packets disrupts communication, incapacitating the system for extended periods [14]. These attacks highlight how vulnerabilities in the CAN bus can be exploited to flood it with malicious traffic, thereby compromising vehicle performance and safety.

3. Tampering Attack;

Gearbox and RPM attacks are prevalent forms of CAN bus tampering that target vehicle control systems.

Gearbox attacks manipulate CAN messages to alter vehicle functions, like speed and transmission, potentially forcing gear shifts and increasing accident risks [15]. RPM attacks alter parameters, like vehicle speed and engine RPM, affecting vehicle performance, potentially bypassing security measures, and threatening vehicle safety.

These attacks underscore how attackers can exploit CAN bus vulnerabilities to interfere with critical systems, heightening vehicle risks and security threats.

4. Fuzzing Attack;

Fuzzing attacks on the CAN bus involve sending random or semi-random data packets to induce system crashes or abnormal behavior, revealing vulnerabilities without requiring full knowledge of the system's functions [16].

This method includes: Fuzzy Message Injection: transmitting messages with invalid data to cause the control system to misinterpret and potentially crash; Fuzzy Parameter Modification: altering CAN message parameters, like engine or vehicle speed, with values outside normal ranges to confuse the control system. These attacks exploit the CAN bus's weaknesses, disrupting the vehicle's control system and increasing safety risks.

2.2. External Network Connections in Vehicles and Related Attacks

2.2.1. V2X Communication Protocols: IEEE 802.11p and VANET

Specifically, we now discuss the IEEE 802.11p protocol, which is widely adopted in V2X (vehicle-to-everything) communications, including vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) scenarios [17]. This protocol plays a critical role in enabling low-latency, high-reliability communication between vehicles and roadside infrastructure, which is essential for ensuring driving safety and supporting cooperative perception in intelligent transportation systems.

IEEE 802.11p is an amendment to the IEEE 802.11 standard (also known as Wireless Access in Vehicular Environments, WAVE) [18], designed specifically for vehicular environments. It operates in the 5.9 GHz frequency band (5.850–5.925 GHz) with a communication range of up to 1000 m and supports data rates ranging from 3 to 27 Mbps. One of its major advantages is its ability to operate without fixed infrastructure, enabling direct vehicle-to-vehicle or vehicle-to-infrastructure communication with minimal latency—typically

under 100 ms. These properties make it particularly suitable for time-sensitive applications, such as forward collision warnings, blind spot detection, and emergency electronic brake light notifications.

In addition, we introduce the background of Vehicular Ad-Hoc Networks (VANETs), which serve as the foundational communication framework for V2X systems [19]. VANETs are a subclass of Mobile Ad-Hoc Networks (MANETs) in which vehicles and roadside units (RSUs) act as mobile network nodes. VANETs are characterized by high mobility, frequent topology changes, and intermittent connectivity, all of which introduce unique vulnerabilities that significantly complicate intrusion detection.

Key features of VANETs include:

- Dynamic topology resulting from vehicle movement.
- Distributed communication, typically without centralized infrastructure.
- Broadcast-based message dissemination.
- Strict latency requirements for safety-critical information exchange.

Security challenges in VANETs encompass a variety of threats, such as:

- Identity spoofing, where malicious nodes impersonate legitimate vehicles.
- Message tampering and replay attacks, aimed at misleading nearby vehicles.
- Denial-of-Service (DoS) attacks, which aim to saturate the network.
- Location spoofing (e.g., GPS spoofing), used to manipulate a vehicle's position awareness.

Given these complexities, VANETs demand robust and specialized Intrusion Detection Systems (IDSs) capable of operating under high mobility, decentralized control, and heterogeneous network structures. Therefore, the integration of IEEE 802.11p-based V2X data into intrusion detection modeling—alongside in-vehicle communication protocols, like CAN—represents an important direction for future research and system design.

2.2.2. External Network Related Attacks

In modern intelligent transportation systems (ITSs), the vehicle external network is pivotal. It serves as the backbone for vehicle communication and is vital for intelligent traffic management, road safety, and enhanced driving experiences. As vehicles connect with pedestrians, infrastructure, other vehicles, and the Internet, the external network becomes a complex information exchange hub, as illustrated in Figure 4. This connectivity is expanding ITS capabilities, propelling the shift from the Internet of Things (IoT) to the Internet of Vehicles (IoV) [20–22].

The swift advancement of in-vehicle external network technology has endowed vehicles with enhanced sensing and communication abilities, allowing for tight integration with the surrounding environment, other vehicles, infrastructure, and smart devices. However, this progress has also introduced a range of security threats and challenges, primarily including:

1. Rogue Networks;

Rogue networks, like fake base stations and malicious Wi-Fi hotspots, can disrupt vehicle communication by spreading false messages or blocking real ones, impacting vehicle navigation and functions [23,24]. Attackers might also impersonate legitimate vehicles to spread misinformation, affecting traffic and communication integrity [25].

2. Data Eavesdropping

Data eavesdropping attacks involve attackers intercepting sensitive vehicle communications through various means, such as wireless, Bluetooth, CAN bus, and GPS signal interception [26]. This allows them to steal vehicle identification, position, and sensor data [27].

For example, by eavesdropping on the CAN bus, attackers can access engine parameters and speed, potentially manipulating the vehicle. GPS signal interception can also enable tracking or planning theft.

3. Information Forgery, Tampering and Replay

Information forgery, tampering, and replay attacks deceive, disrupt, or control vehicles by altering communication data. Attackers might spread false traffic info, fake vehicle IDs [28], or tamper with sensor data to confuse a vehicle's systems [29] or replay intercepted data to mimic legitimate messages [30].

4. Privacy Leakage

Smart connected vehicles risk exposing sensitive data, like user and vehicle IDs, through open communication interfaces, which can be intercepted by attackers, leading to privacy breaches and personal safety threats [31]. For instance, personal information obtained from vehicle-server communications could be used for identity theft [32].

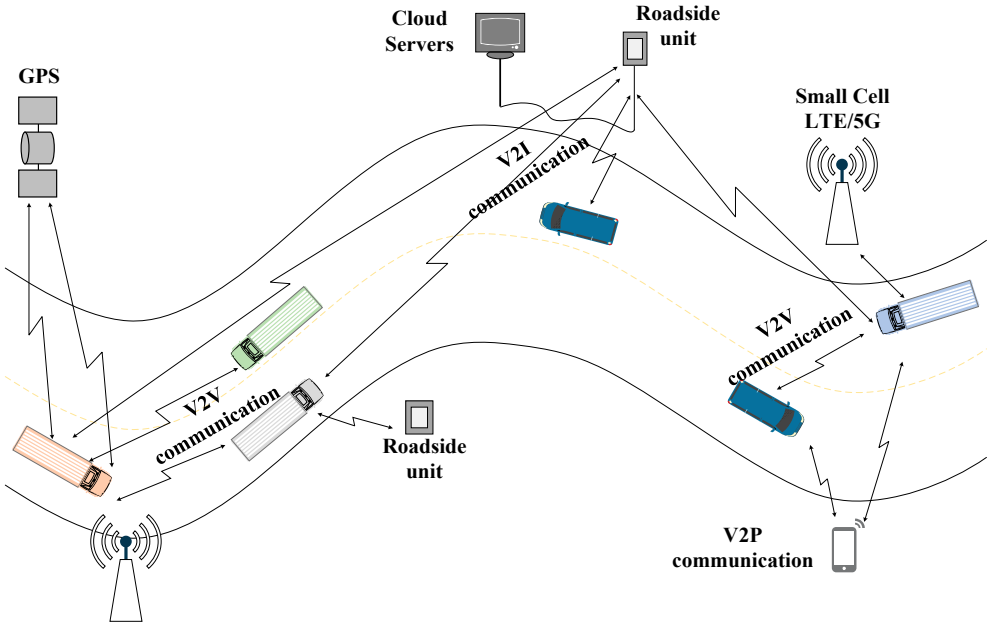


Figure 4. IoV communication technology.

Cyber-attacks exploiting V2X technology and wireless vulnerabilities include DoS, GPS spoofing, jamming, sniffing, brute-force, botnets, infiltration, and network assaults, posing significant challenges to vehicle security and personal privacy [33]. Table 1 summarizes these external network attacks and their respective categories, providing a comprehensive reference for understanding these threats.

Table 1. External network attacks and respective categories.

Type of Attack	Rogue Networks	Data Eavesdropping	Information Forgery, Tampering and Replay	Privacy Leakage
DoS			√	
GPS Spoofing	√			
Jamming	√			
Sniffing		√		
Brute-force			√	
Botnets				√
Infiltration			√	
Web Attack				√

2.3. IDS for Smart Connected Vehicles

The research community is increasingly focusing on Intrusion Detection Systems (IDSs) for in-vehicle networks due to rising cyber threats. Researchers are exploring four main areas for IDS in these systems: advanced detection algorithms based on features, rules, statistics, machine learning, and deep learning.

2.3.1. Feature-Based IDS (FIDS)

The FIDS detects anomalies and potential attacks by analyzing network traffic features, such as message IDs, time intervals, packet lengths, and signal strengths. Feature selection is key to improving detection accuracy and efficiency. A baseline model is created using normal traffic data during training, and incoming traffic is compared against this model for real-time intrusion detection. D'Angelo et al. [34] developed an algorithm that categorizes CAN bus messages in real-time as legitimate or illegitimate, alerting when malicious messages are detected.

Through analysis of in-vehicle networks, researchers have identified features, like device fingerprints, clock offsets, and frequency characteristics, that can enhance detection. Khan et al. [35] introduced an IDS for CAN networks that uses thresholds and statistical attack characteristics to improve detection efficiency. In related literature [36], an RNN-LSTM classifier was used to construct ECU fingerprint signals based on frequency domain features, effectively identifying flooding attacks and offering new protection strategies.

However, feature-based detection methods face challenges in long-term vehicle use and evolving networks. They are often specific to certain attack types, may not fully reflect real-time conditions, and have limitations in assessing response times and analyzing functional security risks. To address these challenges and ensure functional security in smart connected vehicles, particularly in Automotive Cyber–Physical Systems (ACPSs), it is crucial to optimize intrusion detection models and algorithms. This involves implementing improved algorithms and optimized feature combinations [37].

2.3.2. Rule-Based IDS (RIDS)

RIDS use predefined rules to detect patterns and behaviors, often based on expert knowledge, heuristics, or statistical analysis. This method is popular for its simplicity and speed in safety monitoring and anomaly detection. However, the specification-based approach can miss threats if an attacker sends a compliant but disguised message, which is particularly challenging in resource-limited in-vehicle networks.

Wang et al. proposed a method combining CNNs with anomaly detection and Kalman filtering to detect anomalous messages in vehicles [38]. Another approach involves a real-time system that constructs a timing model by monitoring CAN bus traffic, identifying anomalies without predefined specifications [39].

Rule-based methods are simple and fast but have limitations. They are not adaptable to new threats, require frequent rule updates, and are vulnerable to evasion and incompleteness. Rule formulation can be subjective, leading to missed detections or false alarms. To enhance detection, combining rule-based methods with other techniques is necessary.

2.3.3. Statistical-Based IDS (SIDS)

SIDS are effective at extracting statistics from network traffic, using entropy-based methods to measure the uncertainty of CAN IDs in CAN buses [40]. They analyze data through statistical attributes, like distribution and correlation, to detect anomalies [41]. Unlike rule-based methods, SIDS focus on data's statistical properties, revealing hidden patterns. However, data quality and sample representativeness are critical, and selecting appropriate statistical methods and parameters can be challenging.

Olufowobi et al. detected CAN bus data injection attacks using cumulative and change-point detection algorithms on a real dataset [42]. They analyzed bus messages under various driving conditions. Another study analyzed message timing intervals in the frequency domain, achieving 99.9% detection accuracy in experiments on two vehicles [43].

Statistical methods require high-quality, representative data; inadequate data can lead to inaccurate results. Selecting and tuning these methods is complex and requires expertise. They may overlook nonlinear or non-normally distributed relationships and can be slow for large datasets, affecting real-time performance.

2.3.4. Machine Learning/Deep Learning-Based IDS (M/DIDS)

Machine learning and deep learning methods use algorithms and models to learn from data and make predictions or decisions. They identify patterns by training models to classify or predict unknown data.

For instance, Hyun Min Song et al. developed an IDS using a deep convolutional neural network (DCNN) to protect the CAN bus [44]. These methods can handle large-scale data and complex features but require significant computational resources and labeled data for training, and often lack interpretability. Deep belief networks (DBN) can pre-train feature extraction layers unsupervised, but their training is time-consuming [45]. References [46,47] propose an LSTM-based bus intrusion detection method that improves detection of new attacks through transfer learning but performs less well on known attacks.

Duan et al. introduced an improved data volume isolation forest method (MS-iForest) for detecting data tampering attacks, using data volume for anomaly scoring [48]. Maode Ma et al. developed a system for detecting location forgery attacks in the Internet of Vehicles (IoV) [49]. Devaraj et al. used support vector machine classifiers to build a multi-SVM detection model for intrusion detection [50]. Li et al. investigated the security vulnerabilities of both in-vehicle and external networks, proposing a multi-layered hybrid intrusion detection system. This system integrates various detection techniques to enhance security by identifying and mitigating different types of network threats within vehicle networks. The hybrid approach improves both the accuracy and efficiency of detection [51]. While deep learning methods typically achieve high accuracy, they are computationally expensive due to their complex models and lack flexibility in adapting to diverse network conditions.

Table 2 presents the detection targets and types of existing IDS algorithms.

The existing IDS algorithms have the following issues:

- Current intrusion detection systems are typically limited to either internal or external detection, and fail to implement a comprehensive detection mechanism for both internal and external threats simultaneously. Even when internal and external detection systems are present, the types of attacks that can be detected are usually limited to no more than four.
- If the training data are limited, deep learning models may suffer from overfitting, leading to poor generalization on new data. Additionally, data bias can cause the model to produce unfair results, while poor or insufficient data quality can cause the model to learn incorrect patterns, thereby reducing its predictive accuracy and reliability.
- Deep learning models for vehicle intrusion detection may introduce a range of issues, including poor real-time performance, high resource consumption, increased energy usage, deployment challenges, and difficulties in maintenance.

Table 2. Comparison of Intrusion Detection Algorithms.

IDS Types	Reference	Internal Network	External Network	Detection Targets	Types of Attacks
FIDS	[34]	√		ID	DOS, Tampering, Fuzzy
	[35]	√		ID	DOS, Fuzzy
	[36]	√		ID	Tampering
	[37]	√		ID	DOS, Tampering, Fuzzy
RIDS	[38]		√	ID	Web attack
	[39]	√		ID	DOS
SIDS	[40]		√	Data field	Web attack
	[41]	√		ID	DOS, Tampering, Fuzzy
	[42]	√		Data field	Inject
	[43]	√		ID	Inject, DOS
M/DIDS	[44]	√		ID	Inject
	[45]	√		Data field	Tampering, DOS
	[46]	√		Data field	Tampering, DOS
	[47]		√	ID	Tampering
	[48]		√	Data field	Forgery
	[49]	√		Data field	Tampering
	[50]	√		ID	Tampering, DOS
	[51]	√	√	ID, Data field	DOS, Fuzzy, Gear, RPM

3. Stacked Machine Learning Based Intrusion Detection System

To address these issues, we developed a stacked machine learning model that simultaneously detects internal and external threats, dynamically selects the best-performing model, ensures real-time adaptability, reduces resource consumption, and optimizes detection accuracy while mitigating overfitting and data bias.

To enhance comprehensive security, we optimized our stacked machine learning model to be more lightweight, ensuring that it maintains high detection capabilities while requiring fewer resources. Additionally, we strategically placed our Intrusion Detection System (IDS) at the junction of the CAN bus and the central gateway, enabling efficient deployment in intelligent connected vehicles, as illustrated in Figure 5. The IDS on the CAN bus is responsible for identifying malicious messages, leveraging the bus's broadcast nature to monitor signal transitions and analyze CAN IDs and data fields for signs of intrusion. Meanwhile, the IDS embedded within the gateway acts as a vigilant guard, filtering external network traffic to ensure network integrity and stability. By detecting anomalies as they pass through the gateway, this dual-layered detection system provides robust protection for the vehicle's network security.

The proposed IDS is composed of three modules: the data preprocessing module, the feature extraction module, and the internal–external network intrusion detection module. To address the current limitations of insufficient normal and attack data, as well as the limited types of attacks, we employed a variational autoencoder (VAE) to generate both normal and attack data internally. This approach effectively mitigates the class imbalance issue and expands the range of attack types. In the feature extraction module, we optimized the algorithm to enhance the speed of feature extraction. In the detection module, we employed an ensemble learning approach that leverages multiple underlying models to enhance the accuracy and robustness of detection. Details for each module are outlined below.

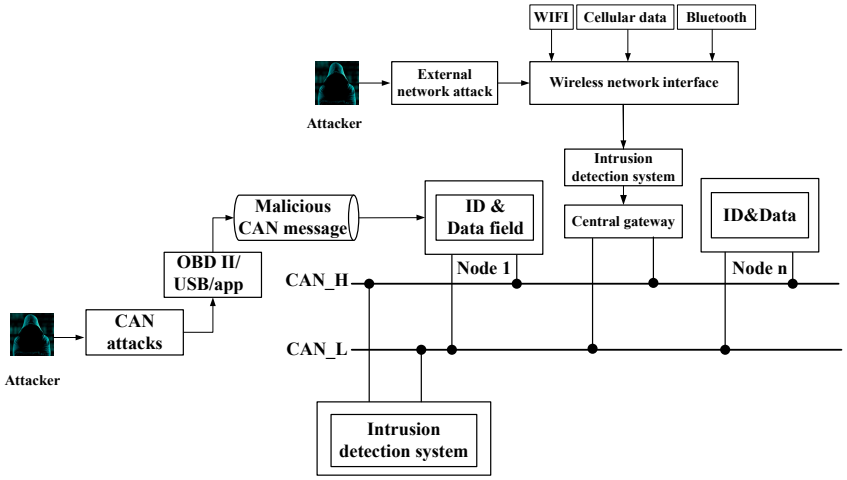


Figure 5. Distributed topology architecture for in-vehicle and external network intrusion detection.

3.1. Data Preprocessing Module

The data preprocessing module receives both normal and abnormal data from internal and external networks. Normal data include CAN bus IDs and data fields for the internal network, while for the external network, it encompasses network attribute data, like packet length, data transmission rate, throughput, inter-arrival time, TCP flags with their counts, subnet size, and active/idle time. Anomaly data involves simulating various attacks using CANoe within the internal network and extracting post-attack data, whereas attacks originating from external networks are analyzed using the CICIDS2017 dataset.

To address the class imbalance issue in both internal and external data, we applied VAE for internal data and the k-means clustering method for external data.

3.1.1. Solution for Handling Class Imbalance in Internal Data

Currently, the extracted in-vehicle bus data typically cover only a few hours of driving data, failing to represent the full data range across the vehicle's entire lifecycle, including potential attack data. To generate internal bus data that closely approximate real-world data and to enable the model to not only learn the low-dimensional representation of the data but also generate new data by inferring the distribution of the latent space, we introduced the VAE algorithm.

The input data x are first mapped to the latent space through the encoder, yielding the mean μ and standard deviation σ, from which the latent variable z is generated. The latent variable z is then used by the decoder to generate the data x. The loss function for the variational autoencoder is shown in Theorem 1 [52]:

$$L = E_q[\log p(x|z)] - D_{KL}[q(z|x)||P(z)] \quad (1)$$

In this context, $E_q[\log p(x|z)]$ represents the reconstruction loss, and $D_{KL}[q(z|x)||P(z)]$ is the Kullback–Leibler (KL) divergence.

3.1.2. Solution for Handling Class Imbalance in External Data

The flowchart of the class imbalance handling process for external data is depicted in Figure 6. Firstly, external data undergo K-means sampling to generate a highly representative subset. Secondly, we assessed the dataset for class imbalance; if identified, the system applies oversampling techniques to balance the dataset. Lastly, we normalized the messages to enhance subsequent data analysis and modeling.

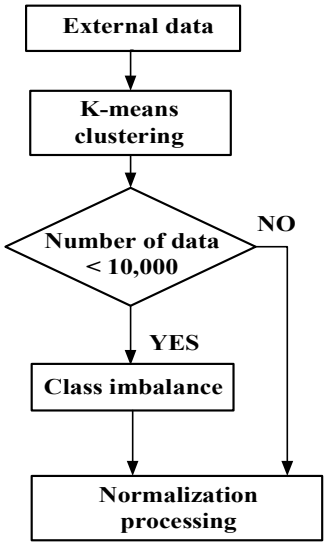


Figure 6. Flowchart for handling class imbalance in external data.

3.1.3. K-Means Clustering

K-means clustering is a widely used technique that partitions data points into K clusters. The data samples within each cluster are relatively close to each other in some measure, making them similar. Data sampling using K-means clustering can effectively reduce the size of the dataset without losing essential information, thereby enhancing the training efficiency of machine learning models.

The steps of the K-mean clustering algorithm are as follows:

- Data points are assigned to K cluster centers based on their Euclidean distance, Manhattan distance, or Mahalanobis distance;
- Each data point is assigned to the cluster whose center is closest to it;
- Recalculate the center of each cluster by computing the mean of all data points in the cluster;
- Repeat steps 2 and 3 until the cluster centers no longer change or the preset number of iterations is reached.

The objective of K-means is to minimize the sum of squared distances between all data points and their respective cluster centroids. The specific objective function is shown in Theorem 2 [53].

$$J = \sum_{i=1}^{K} \sum_{x \epsilon C_i} \|x - \mu_i\|^2 \tag{2}$$

where J is the objective function of clustering, which represents the sum of squared distances from all data points to their respective cluster centroids. K denotes the number of clusters, C_i represents the set of data points in the i-th cluster, and μ_i is the centroid of the i-th cluster.

3.1.4. Class Imbalance

Class imbalance often occurs in network traffic data, where the percentage of normal samples is typically much higher than that of attack samples, leading to biased models and lower detection rates. To address this issue, oversampling techniques, like the Synthetic Minority Over-sampling Technique (SMOTE), are commonly employed. SMOTE enhances dataset diversity and mitigates overfitting risks by synthesizing minority class samples. This method balances the dataset, enabling models to better capture features of minority classes, thereby improving their recognition capabilities and reducing overfitting to noise and unnecessary details in training data.

For each instance $x_{minority}$ in the minority class, where $x_{nearest}$ is a randomly selected sample from the knearest neighbors of $x_{minority}$, the new synthetic instance x_{new} can be represented by Theorem 3 [54].

$$x_{new} = x_{minority} + rand(0,1) \times \left(x_{nearest} - x_{minority}\right) \tag{3}$$

where and (0,1) denotes a random number generated from the uniform distribution between 0 and 1.

3.1.5. Normalization Procession

After implementing the SMOTE method to obtain a representative and balanced dataset, several additional data preprocessing steps are necessary. Initially, categorical features are converted into numerical features using a label encoder to facilitate their compatibility with machine learning algorithms. Subsequently, normalization of the network dataset is performed using the Z-score algorithm. This step is crucial because features collected from network data often exhibit disparate ranges, and machine learning models typically perform optimally on normalized datasets. The Z-score method standardizes features to have a mean of 0 and a standard deviation of 1, thereby enhancing model stability and effectiveness. The Z-score normalization formula is represented as shown in Theorem 4 [55].

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

where x represents the original feature value, μ denotes the mean of the feature, and σ signifies the standard deviation of the feature.

3.2. Feature Extraction Module

Evaluating features using information gain in in-vehicle internal and external network data may encounter challenges, like high dimensionality, sparsity, diverse feature types, data imbalance, and complex feature relationships. Hence, we optimized the feature selection method by integrating the maximum information coefficient algorithm with relevance-based selection techniques to suit the characteristics of in-vehicle external network data, aiming to decrease training time for intrusion detection models.

The process of feature evaluation and selection is as follows:

For a single variable X, the calculation formula for entropy is shown by Theorem 5 [56], where $P(x_i)$ denotes the prior probability of the variable X.

$$H(X) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i) \tag{5}$$

After introducing another variable Y, the conditional entropy $H(X/Y)$ is calculated, as shown in Theorem 6 [56].

$$H(X \mid Y) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j)\log_2 P(x_i \mid y_j) \tag{6}$$

where $P\ (xi \mid yj)$ is the posterior probability of the variable X given the value of Y. In the field of intrusion detection, datasets often contain high-dimensional features, some of which may carry minimal information, potentially affecting classification results. For feature selection, the more information a feature carries, the greater its information gain, making it more important. The calculation of information gain is shown by Theorem 7 [56].

$$IG(X, Y) = H(X) - H(X \mid Y) \tag{7}$$

Using the formulas mentioned above, we can calculate the information gain between each feature and the target variable, thereby assessing the importance of each feature.

Although the feature selection method using the maximum information coefficient helps eliminate unimportant features and reduce time complexity, many redundant features may still exist. To address this issue, we propose a method that combines the maximum information coefficient with fast correlation-based selection to remove redundant features (referred to as the IG-NF algorithm), thereby enhancing the model's performance and efficiency. First, the range of values for mutual information is shown in Theorem 8 [56].

$$0 \leqslant I(X:Y) \leqslant min\{H(X), H(Y)\} \tag{8}$$

This indicates that the mutual information is non-negative, i.e., it is at least 0. When the mutual information is 0, it indicates that the two variables are independent of each other, i.e., the information of one variable does not contain any information about the other variable. This means that the mutual information between X and Y does not return more than their respective entropies.

Due to the large variations in entropy values of external network data, uncertain entropy values can lead to unreasonable *I (X: Y)* values. Therefore, it is necessary to process *I (X: Y)* using the maximum information coefficient. The maximum information coefficient can compensate for the bias in mutual information among multiple features and restrict its value range to [0, 1]. Thus, for the maximum information coefficient of the random variables X and Y, it can be determined by the minimum value of *H(X)* and *H(Y)*. The specific formula is shown in Theorem 9 [56]:

$$I_{max}(X:Y) = \frac{I(X:Y)}{\min\{H(X), H(Y)\}} \tag{9}$$

The importance of each feature is calculated using (9), and these importance values are normalized to reflect the relative importance among the features.

To effectively remove redundant features, we improved the Fast Correlation-Based Filter (FCBF) algorithm. The symmetric uncertainty SU_{max} *(X, Y)*, measures the direct correlation between features, normalized by the maximum information coefficient, as shown by Theorem 10 [56].

$$SU_{max}(X,Y) = 2\left[\frac{I_{max}(X:Y)}{H(X)+H(Y)}\right] \tag{10}$$

The SU_{max} $(X,\ Y)$ ranges from 0 to 1, where a value of 1 indicates perfect correlation between two features X and Y, and a value of 0 signifies complete independence. By setting a correlation threshold α, if the correlation between two features exceeds this threshold, only the one with higher feature importance is retained, while the other is excluded. This process is repeated until no pair of features in the feature set exceeds the threshold α, ensuring that all pairwise correlations between features are below α.

3.3. Internal and External Network Intrusion Detection Module

In order to detect various attacks on both internal and external networks of the vehicle, the designed intrusion detection system needs to handle multiple classification tasks. We used three base-learner and constructed a core intrusion detection model by combining meta-learner through stacking techniques. This results in a stacked model-based intrusion detection system for in-vehicle networks that can effectively identify and classify various types of attacks.

3.3.1. Selecting Base-Learner and Optimizing Parameters

In-vehicle bus network data transmission requires efficiency, real-time responsiveness, and reliability, while handling external network data demands algorithms with high robustness and generalization capabilities. Light Gradient Boosting Machine (LightGBM) and Extreme Gradient Boosting (XGBoost) algorithms excel in rapid processing and analysis of large-scale real-time data, ensuring timely detection of attack behaviors within constrained time frames. These ensemble models reduce the model training time in resource-limited vehicle network environments, enhancing efficiency. LightGBM and XGBoost effectively manage nonlinear complexities in network data, performing better with lower complexity and lighter deployment suitability for vehicles. We used Classification and Regression Trees (CART) as the third base learner. The tree structure generated by this algorithm is simple, and feature selection naturally occurs during the tree-building process, which aligns with the lightweight deployment requirements of this paper. Moreover, the CART algorithm is highly robust in handling missing values and outliers.

The process of parameter adjustment and optimization is as follows:

When using the DT algorithm, setting the splitting criterion to Gini impurity and constructing a Classification and Regression Tree (CART) model often yields better performance than using information gain. By employing the cost function *C(S)* defined by Theorem 11 [57], CART selects the tree within S that minimizes empirical risk.

$$C(S) = \hat{L}_n(S) + \alpha|S| \tag{11}$$

where $\hat{L}_n(S)$ represents the empirical risk of the tree S, $|S|$ denotes the cardinality of the tree, which is the number of nodes, and α is a constant. Therefore, when selecting the optimal tree, it is necessary to consider both the empirical risk and the complexity of the tree.

The XGBoost algorithm is based on decision trees, and its performance is influenced by the number of trees T. The objective function of XGBoost is illustrated in Theorem 12 [58].

$$Obj = -\frac{1}{2}\sum_{j=1}^{t}\frac{G_j^2}{H_j + \lambda} + \gamma|T| \tag{12}$$

where G_j and H_j represent the sum of the first-order and second-order gradient statistics for the j-th leaf, respectively, while T denotes the number of trees. The parameters λ and γ are regularization coefficients used to control the model's complexity. Since the gradient statistics are based on the sum of the prediction scores from T trees, and the number of leaves increases with the tree depth D, both T and D directly affect the value of XGBoost's objective function.

During model training, we used a grid search method to find the optimal parameter combination. In the initial stage of training, we started with a small number of trees and shallower depths, gradually increasing the values of key parameters while evaluating the changes in model accuracy. We continued this process until signs of overfitting appeared, indicated by a decline in accuracy. At this point, the final tree depth D was set to 8 and the number of trees T was set to 200. Similarly, the minimum sample split and the minimum sample leaf were also tuned and ultimately set to 8 and 3, respectively.

3.3.2. Stacked Modeling

To enhance the model's accuracy, this paper employs an ensemble learning strategy known as stacking. This technique constructs a powerful classifier through three levels of model training. The process of the stacked model is depicted in Figure 7.

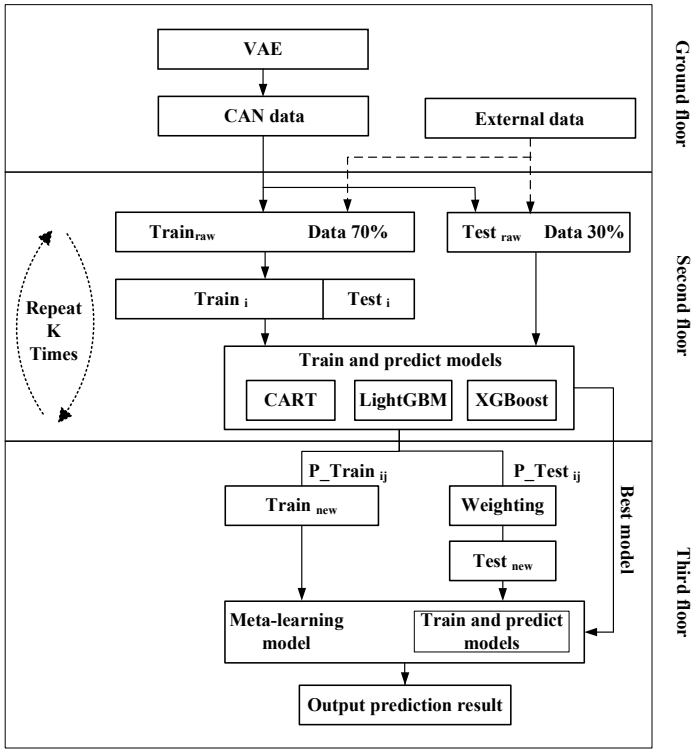


Figure 7. The process of the stacked model.

In the first layer, VAE is used to generate internal CAN bus data, including both normal and attack data. In the second level, three base algorithms are individually trained and evaluated. Their output results serve as input features for the meta-learner in the three level. In these three level, the base algorithm with the best performance in the second layer is chosen as the base model for the meta-learner. By training the meta-learner, we fully leverage and integrate the strengths of the base models from the previous layer, further improving the prediction accuracy and generalization capability of the entire ensemble model. This hierarchical stacking technique effectively combines the advantages of multiple base models to build a more accurate and robust intrusion detection system. Compared to a single machine learning algorithm, the stacking model can more effectively capture the complex patterns in vehicle network data, thereby enhancing detection performance.

1. First Layer

Due to the scarcity of internal CAN bus data, with only a few million driving data points available, it is insufficient to comprehensively cover the entire vehicle lifecycle, including potential attack data. To enhance the generalization ability and robustness of the detection system against both known and unknown attacks, we utilized the VAE algorithm to generate different types of normal and attack data. This approach not only expands the diversity of the dataset but also enables the model to better learn the underlying distribution of the data, thereby improving its ability to accurately identify and respond to various attacks in practical applications.

2. Second Layer

We divide the original data into a training set $Train_{orig}$ (70% of the original data) and a test set $Test_{orig}$ (30% of the original data). We applied k-fold cross-validation on the

original training set train, using a 10-fold cross-validation approach where train is split into 10 subsets. The specific calculation formula is shown in Theorem 13 [59].

$$Train_{orig} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n/k} \end{Bmatrix} = \begin{Bmatrix} x_{1+\frac{n}{k}} \\ x_{2+\frac{n}{k}} \\ \vdots \\ x_{2*\frac{n}{k}} \end{Bmatrix} + \ldots + \begin{Bmatrix} x_{i*\frac{n}{k}+1} \\ x_{i*\frac{n}{k}+2} \\ \vdots \\ x_{(i+1)*\frac{n}{k}} \end{Bmatrix} + \ldots + \begin{Bmatrix} x_{(k-1)*\frac{n}{k}+1} \\ x_{(k-1)*\frac{n}{k}+2} \\ \vdots \\ x_n \end{Bmatrix}, i \in [1,k] \tag{13}$$

where x is the feature data corresponding to each request, n is the total number of requests and k is the number of fold crossings.

Training of base learners: In $Train_{orig}$, select one portion of data as the test set $Test_i$, and the remaining as the training set $Train_i$, as shown in Theorems 14 and 15. Train three base algorithms on the training set $Train_i$, and then use the trained models to predict on $Test_i$ and the original test set $Test_{orig}$, as shown in Theorems 16 and 17. Repeat this process 10 times, each time using different randomly selected splits of $Train_i$ and $Test_i$.

$$Train_i = \begin{Bmatrix} x_1 \\ \vdots \\ x_{i*\frac{n}{k}} \\ x_{(i+1)*\frac{n}{k}+1} \\ \vdots \\ x_n \end{Bmatrix}, i \in [2,k-1] \tag{14}$$

$$Train_1 = \begin{Bmatrix} x_{1+\frac{n}{k}} \\ x_{2+\frac{n}{k}} \\ \vdots \\ x_n \end{Bmatrix};\ Train_k = \begin{Bmatrix} x_1 \\ \vdots \\ x_{(k-1)*\frac{n}{k}-1} \\ x_{(k-1)*\frac{n}{k}} \end{Bmatrix}$$

$$Test_i = \begin{Bmatrix} x_{i*\frac{n}{k}+1} \\ x_{i*\frac{n}{k}+2} \\ \vdots \\ x_{(i+1)*\frac{n}{k}} \end{Bmatrix}, i \in [0,k-1] \tag{15}$$

$$Train_i \overset{Model_j\ train}{\rightarrow} Test_i \overset{Model_j\ forecast}{\rightarrow} P_{Train\,i,j} \tag{16}$$

$$Train_i \overset{Model_j\ train}{\rightarrow} Test_{orig} \overset{Model_j\ forecast}{\rightarrow} P_{Test\,i,j} \tag{17}$$

where $j \in$ [1,m], and in this part m = 3, m denotes the number of base learners in the first layer.

3. Third Layer

The predictions obtained after training in the first layer will be used as inputs to train and predict using the meta-learner.

The training set for the second layer: Concatenate $P_Train_{i,j}$, row-wise to obtain the new training set $NTrain_i$, as shown in Theorem 18.

$$NTrain_i = \begin{Bmatrix} P_{Train1,1} & \cdots & P_{Train1,3} \\ \vdots & \ddots & \vdots \\ P_{Train10,1} & \cdots & P_{Train10,3} \end{Bmatrix} \tag{18}$$

The test set for the second layer: Weight $P_Test_{i,j}$ row-wise to obtain the new test set $NTest_i$, as shown in Theorem 19.

$$NTest_i = weights(P_Test_{i,1} \quad P_Test_{i,2} \quad P_Test_{i,3}) \tag{19}$$

Training of meta-learner: After evaluating and analyzing the performance of the three base algorithms, the model selects the optimal algorithm as the meta-learner. This approach enables the system to adaptively choose the best base learner based on different datasets and attack types, thereby continuously optimizing and enhancing its accuracy.

The new training set and test set are inputted into the third-layer model for training. After training, the model predicts on the newly formed test set, and the resulting predictions constitute the final detection results.

Internal CAN bus data provides detailed information about the vehicle's internal state and behavior, while external data reflects the surrounding environment of the vehicle. Through the stacked model, both types of data can be trained simultaneously, enabling the model to acquire more comprehensive and diverse information, thereby enhancing its comprehensiveness and diversity.

The stacked model improves prediction accuracy and decision-making capabilities, enhances generalization, and mitigates risks associated with single-model approaches. By integrating and analyzing both internal and external data, the model can intelligently adapt to different driving scenarios, thereby enhancing vehicle safety and intelligence.

4. Experimentation and Analysis

In this chapter, we first introduce the construction of the internal datasets and the selection of the external datasets. Next, we evaluate the performance of the stacked model. Finally, we explain the hardware implementation of the model and its performance metrics.

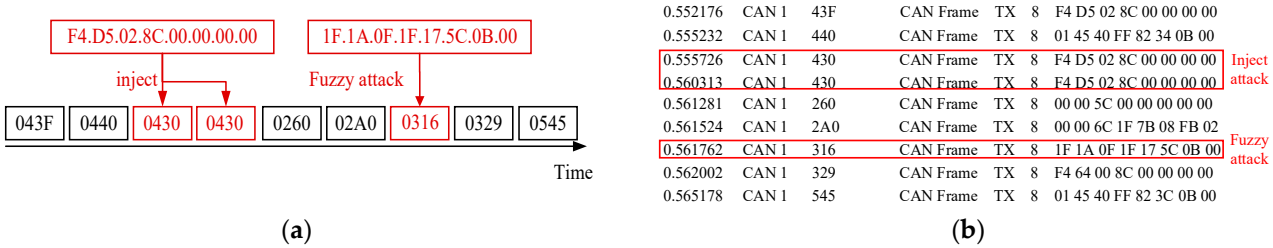
4.1. Constructing the Datasets

4.1.1. Internal Datasets

We divided the internal CAN bus data into normal datasets and attack datasets. The normal datasets are extracted from vehicles in motion, while the attack data were obtained by simulating a series of attack behaviors in CANoe. Below are the implementations of various attack types and the attack test results in CANoe; the black part in the figure represents the normal data of the CAN bus, and the red part is the attack data we simulated through CANoe:

1. Fuzzy Attack

We explored two scenarios in our fuzzy attack experiments. First, we conducted an injection attack, a type of fuzzing attack aimed at intentionally disrupting or manipulating vehicle functions by sending unauthorized CAN messages. For instance, as depicted in Figure 8a, we injected a forged data segment into a message identified by ID 0430H and transmit it onto the CAN bus. Sending these invalid data segments disrupts the vehicle's normal operation.



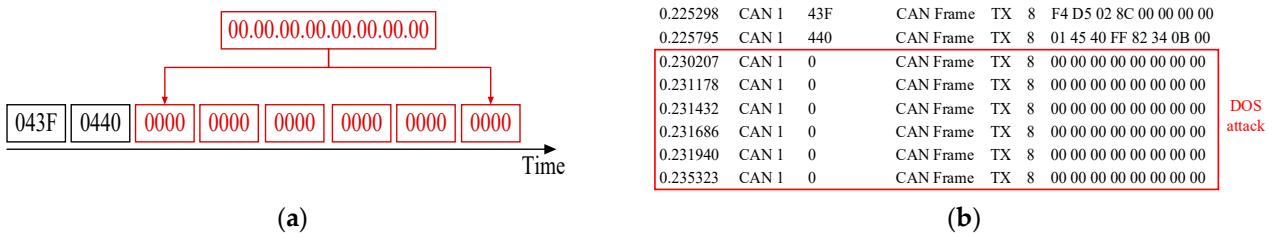
(**a**) (**b**)

Figure 8. (**a**) Inject and Fuzzy attack node transmission graph; (**b**) CANoe monitoring result.

Secondly, in another fuzzy attack scenario, we used an already authorized ID, such as 0316H, as a node for attack. In this instance, we injected a message {1F, 1A, 0F, 1F, 17, 5C, 0B, 00} onto the CAN bus. This message involves altering the vehicle's start button and speed data, thereby interfering with its regular operation. Figure 8b illustrates the monitoring results transmitted on the CAN bus following a real-time fuzzing attack conducted using CANoe.

2. Dos Attack

DoS attacks are achieved by injecting high frequency, high priority packets in a short period of time. This requires targeting the CAN bus using multiple IDs with high priority as attack nodes. As shown in Figure 9a, we used the highest priority ID 0000H as an attack node to send a large number of messages with data 0, which results in delayed transmission of low-priority packets, Figure 9b shows the result after monitoring in CANoe.



(**a**) (**b**)

Figure 9. (**a**) DoS attack node transmission graph; (**b**) CANoe monitoring result.

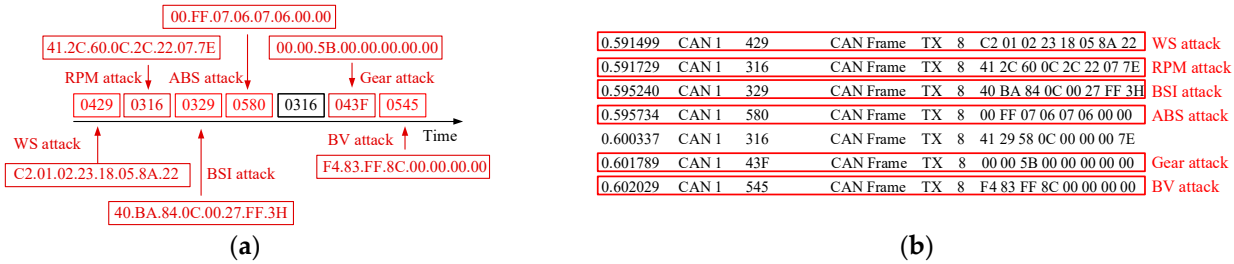
When using 200 messages as a DoS attack on the CAN bus, the bus load is 9.87%. When using 800 messages for the attack, the bus load increases to 17.33%. DoS attacks can threaten the safety of the vehicle by causing CAN bus congestion, bandwidth limitation, data loss, and system crash. Therefore, it is crucial to identify and respond to such attacks in a timely manner.

3. Tampering Attack

Tampering attacks involve the intentional alteration of critical vehicle data to disrupt normal operations. In our experiment, we simulated such attacks by modifying specific ID messages in the vehicle's communication system, such as engine speed (RPM), gear shifting (Gear), wheel steering (WS), brake switch indicator (BSI), anti-lock braking system (ABS), and battery voltage (BV). These alterations cause erroneous data readings and potential malfunctions, highlighting the risk of vehicle data anomalies resulting from tampering, as shown in Figure 10.

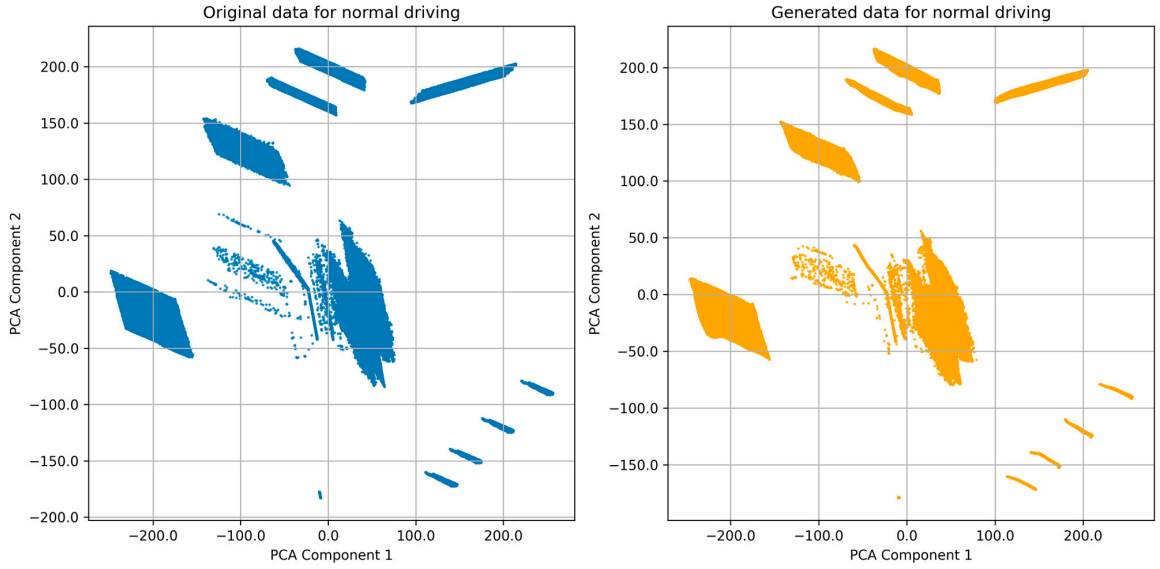
The datasets of extracted data are limited, so we used VAE to augment and generate both normal and abnormal data. The original data for normal driving are shown in Figure 11a, which is a two-dimensional distribution plot generated after dimensionality reduction, containing over a million messages. Figure 11b displays the two-dimensional distribution plot of the over five million normal data points generated using VAE. By

comparing the principal component analysis (PCA) plots of the original data and the generated data, it can be seen that the feature distribution of the generated data is highly similar to that of the original data, indicating that the data generated by VAE can simulate CAN bus data. Generating data through VAE can solve the problem of class imbalance in the dataset, and VAE, which is different from the pure copy–paste of smote, can improve the generalization ability of the model.



(**a**) (**b**)

Figure 10. (**a**) Tampering attack node transmission graph; (**b**) CANoe monitoring result.



(**a**) (**b**)

Figure 11. (**a**) The original data for normal driving (**b**) The generated data for normal driving.

Similarly, we used VAE to generate attack datasets for DoS, Fuzzy, RPM/Gear, WS, ABS, BSI, and BV, as shown in Table 3. The datasets are increased by an average of five times. The experimental results indicate that VAE can generate multi-distribution data similar to vehicle behavior, which helps improve the model's ability to recognize different situations.

Table 3. Types and sizes of internal datasets.

Data Types	Original Data Size	Generated Data Size
Normal	1,048,575	5,242,875
DoS	447,978	2,239,890
Fuzzy	259,676	1,798,380
RPM	384,300	1,921,500
Gear	499,020	2,495,100
WS	499,425	2,497,125
ABS	744,385	3,721,925
BSI	434,375	2,171,875
BV	555,625	2,778,125

4.1.2. External Datasets

In this study, we utilized the CICIDS2017 datasets created by Iman Sharafaldin from the Canadian Institute for Cybersecurity as our external network datasets. In the field of cybersecurity, many research projects and papers utilize standard network security datasets to develop IDS for external vehicular networks. These datasets include KDD-99, NSL-KDD, Kyoto2006+, UNSW-NB15, and CICIDS2017, among others. Among these datasets, CICIDS2017 stands out as one of the most representative external network datasets currently available, encompassing advanced and diverse data features, instances, and types of network attacks. To integrate the CICIDS2017 dataset with applications in intelligent connected vehicles, we conducted a detailed analysis of the CICIDS2017 dataset and correlated it with external threats to vehicular networks.

In the data preprocessing stage of our intrusion detection system, we employed class imbalance handling techniques to increase the number of samples, particularly for attack types with fewer instances, such as Bot, DoS, Probe, and Network Attacks. Our objective was to ensure that each attack category dataset had at least 10,000 samples. By addressing class imbalance, we aimed to avoid biased models and achieve more accurate attack detection results. Table 4 illustrates that each dataset has at least 10,000 samples after addressing the class imbalance issue.

Table 4. External network datasets.

Data Types	Original Data Size	After Preprocessing Size
Normal	2,273,097	2,273,097
DoS	380,699	380,699
Port-Scan	158,930	158,930
Brute-Force	13,835	13,835
Web-Attack	2180	10,000
Botnet	1966	10,000
Infiltration	36	10,000

4.2. Evaluation Metrics for Proposed IDS

In this paper, we used the following metrics to evaluate the performance of the intrusion detection system.

1. Accuracy

The accuracy of a classifier is defined as the ratio of the number of correctly classified samples to the total number of samples, as shown in Theorem 20 [60]. A higher accuracy indicates better performance of the classifier. Accuracy is a crucial metric for evaluating classifier performance, especially in cases where sample classes are balanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{20}$$

where TP is the number of samples that the model correctly predicts as positive cases, and TN is the number of samples that the model correctly predicts as negative cases. FP is the number of samples where the model incorrectly predicts negative cases as positive cases, and FN is the number of samples where the model incorrectly predicts positive cases as negative cases. FN is the number of samples where the model incorrectly predicts a positive case as a negative case.

2. Precision

Precision measures the proportion of true positive samples among all samples predicted as positive by the classifier. It is calculated as the ratio of the number of true positive

samples to the total number of samples predicted as positive, as shown in Theorem 21 [60]. A higher precision indicates that the classifier correctly identifies a larger proportion of positive samples among those predicted as positive, reflecting better classifier performance.

$$Precision = \frac{TP}{TP + FP} \tag{21}$$

3. Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive samples that are correctly predicted as positive by the classifier among all samples that are actually positive. It is calculated as the ratio of the number of true positive samples to the total number of samples that are actually positive, as shown in Theorem 22 [60]. A higher recall indicates that the classifier correctly identifies a larger proportion of positive samples among all samples that are truly positive, indicating better classifier performance.

$$Recall = \frac{TP}{TP + FN} \tag{22}$$

4. *F*1-Score

The *F1* score is the harmonic mean of precision and recall, combining their strengths and weaknesses into a single metric, as shown in Theorem 23 [60]. A higher *F*1 score indicates better classifier performance. The *F*1 score considers both precision and recall simultaneously, making it suitable for scenarios with class imbalance. When the sample classes are balanced, the *F*1 score is equivalent to the arithmetic mean of precision and recall.

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{23}$$

4.3. Comparative Analysis of Training Time

While high accuracy is important in in-vehicle system intrusion detection, the diversity of in-vehicle bus data, the sheer volume of data, and the impact of different driver habits are sometimes overlooked. In addition, as the number of ECUs increases, the number of unknown attack types also increases. Using fixed models is difficult to update in time for new attacks after deployment to real vehicles, and even updating via the cloud requires a long training time.

In real vehicle deployments, the models need to be able to quickly adapt to new attacks and data pattern changes. With a hierarchical training and updating strategy, stacked models can quickly update some sub-models locally without re-training the entire model, thus greatly reducing training time. Combining on-board computing power and cloud resources, stacked models can complete initial training and updates locally, and for major updates can leverage cloud resources to improve real-time performance and response time.

The results of testing algorithms on the internal bus datasets are presented in Table 5. Using original feature extraction algorithm, our intrusion detection system achieved 100% accuracy and detection rate, indicating detection of all trained attacks. However, the stacked model required longer training times. To reduce this training time, we proposed the IG-NF algorithm within the feature extraction module to expedite feature extraction. Test results show that with the IG-NF feature selection method, training the model on the internal bus datasets containing over 2 million records took 331.6 s. This represents a 73.1% reduction in training time for the stacked model, while maintaining high accuracy.

Table 5. Comparative analysis of feature selection of proposed system on internal CAN bus network.

Proposed IDS	Accuracy (%)	Precision (%)	Recall (%)	*F*1-Score	Training Time (S)
using the original feature extraction	100	100	100	1	1237.1
Using IG-NF feature extraction	99.99	99.99	99.99	0.999	331.6

The test results on the external network datasets are shown in Table 6. Utilizing our proposed IG-NF algorithm, we rapidly selected 20 important features out of 78 features. After feature selection, the training time of the stacked model reduced by 21.2% compared to using a conventional feature algorithm. This indicates that our IG-NF feature selection method can maintain high accuracy in IDS while saving training time, achieving good results on both internal and external datasets.

Table 6. Comparative analysis of feature selection of the proposed system on external networks.

Proposed IDS	Accuracy (%)	Precision (%)	Recall (%)	*F*1-Score	Training Time (S)
using the original feature extraction	99.86	99.8	99.77	0.9978	3519.3
Using IG-NF feature extraction	99.83	99.75	99.76	0.9977	2786.8

4.4. Comparative Analysis of IDS

Most IDSs utilize deep learning for training. In this study, we compared our proposed IDS with a publicly available DCNN model. As shown in Table 7, our proposed system achieves high accuracy of 99.99% on the internal bus dataset. While the DCNN method achieves a high average *F*1 score of 0.949, it requires a graphics processing unit (GPU) for model training, making it challenging to deploy in vehicle systems under budget constraints. In contrast, our proposed system not only detects IDs but also data field, and it offers advantages in deployment cost and training time over the DCNN method, thus demonstrating higher practical applicability.

Table 7. Comparison of detection rates of the algorithms in this paper on internal buses.

IDS	Accuracy (%)	Precision (%)	Recall (%)	*F*1-Score
DCNN	96.62	93.38	96.62	0.949
Proposed	99.99	99.99	99.99	0.999

For intrusion detection of external network data, we selected the MTH-IDS system from the same CICIDS2017 dataset for comparison. As shown in Table 8, the simulation results, MTH-IDS achieved an accuracy of 99.88% with a training time of 1563.4 s, while our proposed IDS model attained an accuracy of 99.83% with a significantly shorter training time of 352.8 s, meeting the lightweight requirements of vehicles. Therefore, the experimental results indicate that the proposed intrusion detection system can effectively distinguish between normal and malicious network traffic data and successfully detect various known types of network attacks in vehicle systems.

Table 8. Comparison of the detection rate of this paper's algorithm on external networks.

IDS	Accuracy (%)	Precision (%)	Recall (%)	*F*1-Score	Time (s)
MTH-IDS	99.88	99.81	98.25	0.998	1563.4
Proposed	99.83	99.75	99.76	0.997	352.8

4.5. Hardware Experimental Analysis

The experimental environment consists of the Final Shell service manager and Raspberry Pi 3B+ development boards. To evaluate the real-world deployment feasibility of the proposed model, we implemented and tested it on a Raspberry Pi 3 Model B+ platform. The hardware specifications of the Raspberry Pi 3B+ are as follows:

- Processor: Broadcom BCM2837B0, Quad-core Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz.
- Memory: 1 GB LPDDR2 SDRAM.
- Storage: 32 GB microSD card (Class 10) used as the primary storage.
- Networking: 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, Gigabit Ethernet (over USB 2.0).
- Operating System: Raspbian OS (Debian-based Linux distribution).
- Power Supply: 5 V/2.5 A DC via micro USB connector.

An intrusion detection model is built in PyCharm and uploaded to Raspberry Pi 3B+ via the Final Shell service manager to simulate the detection system within a vehicle. The experimental setup is shown in Figure 12.



Figure 12. Experimental environment.

We tested the proposed intrusion detection system on Raspberry Pi to assess its feasibility for deployment in vehicle environments. Based on experimental results from the test set, the system achieved accuracies of 99.97% on internal and 99.81% on external network datasets. Table 9 demonstrates the system's potential for deployment in actual vehicles.

Table 9. Detection results following hardware testing.

Datasets Type	Accuracy (%)	Precision (%)	Recall (%)	*F*1-Score
Internal	99.97	99.97	99.97	0.9997
External	99.81	98.33	91.95	0.9467

In addition, we evaluated the proposed IDS confusion matrices for the internal bus dataset and the external network dataset, as shown in Figure 13 and Figure 14, respectively. For the internal dataset and external dataset, the proposed IDS accurately detects various attacks. This demonstrates the robust generalizability of the stacked model proposed, capable of addressing a diverse range of attacks, including future updates.

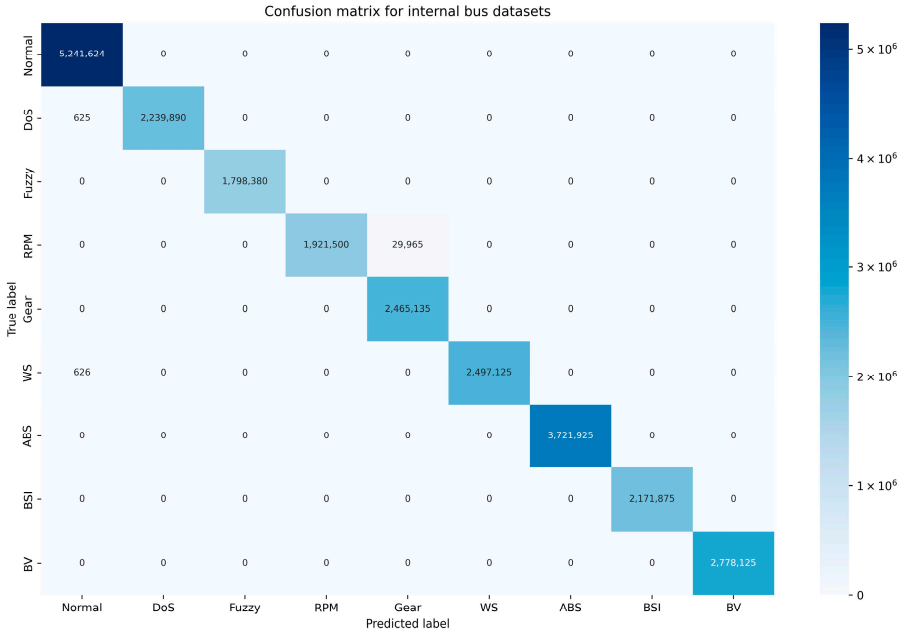


Figure 13. Confusion matrix for internal bus datasets.

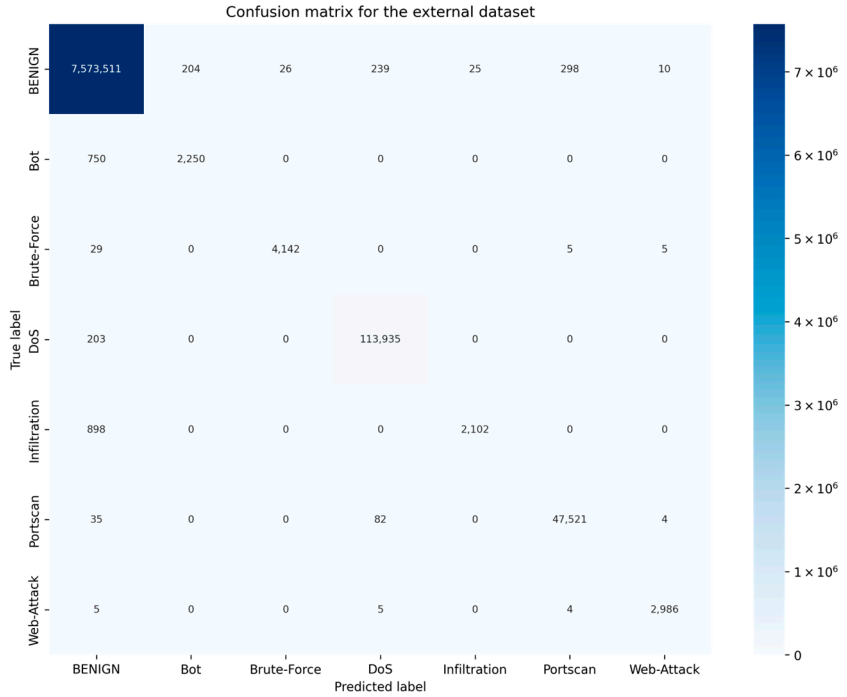


Figure 14. Confusion matrix for the external dataset.

Current IDSs using deep learning algorithms typically require training times in hours, making it challenging to meet the real-time updating needs of vehicle data in dynamic and multi-attack environments. In contrast, our proposed stacked machine learning algorithm can dynamically train and update in real-time to better adapt to practical applications.

To verify real-time training effectiveness and detection rates, we extracted a subset from the original datasets, comprising 818,441 internal network data and 56,662 external network data samples. Our proposed IDS can detect all internal network data within 15 s (Figure 15), while external network data only requires 5 s (Figure 16). The system processes each internal CAN bus data sample in just 0.018 ms and each external network data sample in 0.088 ms. This demonstrates that our proposed intrusion detection system operates

well below the latency requirements of vehicle network security, ensuring both real-time performance and latency compliance.

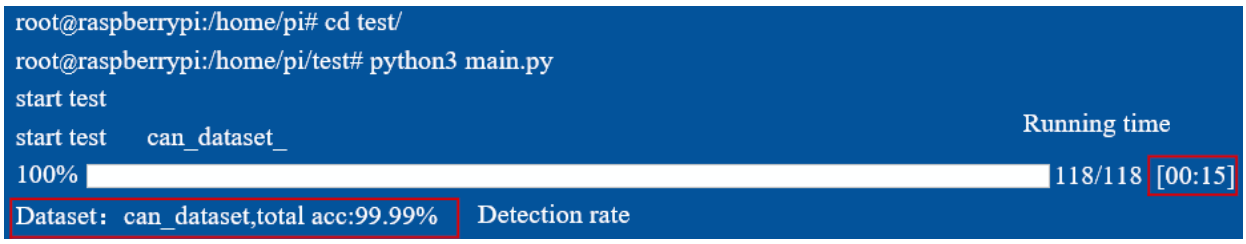


Figure 15. Intrusion detection results for internal data.

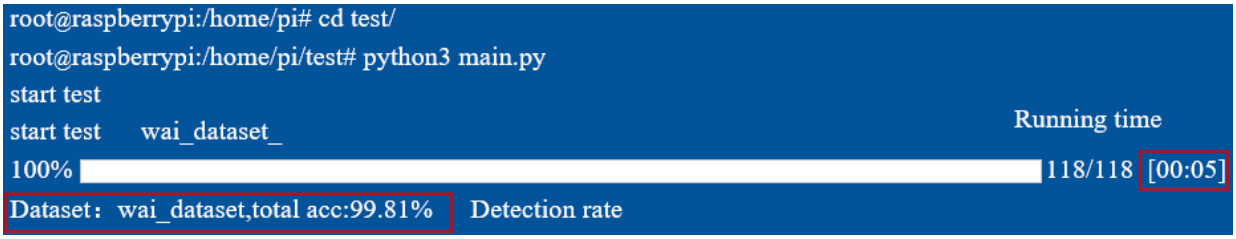


Figure 16. Intrusion detection results for external data.

5. Conclusions

With the rapid development of connected vehicles and autonomous driving technologies, concerns about automotive network security have become increasingly critical. This paper proposes a vehicular intrusion detection system (IDS) based on a stacked model approach. The system first identifies foundational models suited to the characteristics of both internal and external vehicle networks. It then employs stacked ensemble techniques to construct a detection model capable of accurately identifying anomalies in both internal and external network data. Unlike traditional single-model methods, this system provides several advantages: it tailors detection models for both internal and external network environments, enhancing specificity; it improves detection accuracy through ensemble integration, achieving over 99.92% accuracy for both internal and external detections; it demonstrates robust real-time performance, meeting the stringent requirements of vehicular environments; and it is primed for future algorithmic optimizations to enhance detection speed and expand coverage to more automotive network attack scenarios, thereby continually strengthening security for intelligent connected vehicles.

The proposed stacked machine learning-based IDS shows significant advantages over traditional deep learning methods in terms of both accuracy and training time. The system is able to quickly adapt to dynamic, evolving environments and effectively detect various internal and external threats. Testing on both internal and external datasets demonstrates that the system not only achieves high accuracy and efficiency but also supports real-time detection in vehicular environments, meeting the practical deployment requirements. By optimizing the lightweight design of the stacked model, the system strikes a balance between real-time responsiveness and accuracy. This offers an efficient and flexible security solution for intelligent connected vehicles, enhancing both their safety and operational reliability.

Author Contributions: Conceptualization, Y.W. and X.Z.; methodology, X.Z. and Y.W.; software, X.Z.; validation, Y.W. and X.Z.; resources, Y.X.; data curation. J.L.; writing—original draft preparation, Y.W. and X.Z.; writing—review and editing, Y.W. and X.Z.; visualization, S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant Nos. 62201492 and 62161049; and was supported by the Jilin Province Natural Science Foundation, grant Nos. YDZJ202301ZYTS409 and YDZJ202501ZYTS641.

Data Availability Statement: Data are contained within this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, C.; Zhang, B.; Liu, M.; Wei, S.; Zhang, J.; Shen, L. Research on intelligent networking automotive technology and information security based on CAN bus. *IOP Conf. Ser. Mater. Sci. Eng.* **2019**, *688*, 044058. [CrossRef]
2. Bendiab, G.; Hameurlaine, A.; Germanos, G.; Kolokotronis, N.; Shiaeles, S. Autonomous Vehicles Security: Challenges and Solutions Using Blockchain and Artificial Intelligence. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3614–3637. [CrossRef]
3. Obaidat, M.; Khodjaeva, M.; Holst, J.; Ben Zid, M. Security and Privacy Challenges in Vehicular Ad Hoc Networks. In *Connected Vehicles in the Internet of Things*; Mahmood, Z., Ed.; Springer: Cham, Switzerland, 2020; pp. 223–251. [CrossRef]
4. Al-Areqi, A.; Szakács, T. Can Bus Communication Demonstration Tool for Education. In Proceedings of the 2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 19–21 May 2021; pp. 000299–000304. [CrossRef]
5. Voss, W. *A Comprehensible Guide to Controller Area Network*; Copperhill Media: Amherst, MA, USA, 2008.
6. Zhang, X.; Mu, L.; Zhao, J.; Xu, C. An efficient anonymous authentication scheme with secure communication in intelligent vehicular ad-hoc networks. *KSII Trans. Internet Inf. Syst. (TIIS)* **2019**, *13*, 3280–3298. [CrossRef]
7. Mwanje, M.D.; Kaiwartya, O.; Aljaidi, M.; Cao, Y.; Kumar, S.; Jha, D.N.; Naser, A.; Lloret, J. Cyber security analysis of connected vehicles. *IET Intell. Transp. Syst.* **2024**, *18*, 1175–1195. [CrossRef]
8. Boström, A.; Wotawa, F. Wireless Threats Against V2X Communication. In Proceedings of the 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS), Chiang Mai, Thailand, 22–26 October 2023; pp. 529–540. [CrossRef]
9. Staat, P.; Jansen, K.; Zenger, C.; Elders-Boll, H.; Paar, C. Analog Physical-Layer Relay Attacks with Application to Bluetooth and Phase-Based Ranging. In Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), San Antonio, TX, USA, 16–19 May 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 60–72. [CrossRef]
10. Ju, Z.; Zhang, H.; Li, X.; Chen, X.; Han, J.; Yang, M. A Survey on Attack Detection and Resilience for Connected and Automated Vehicles: From Vehicle Dynamics and Control Perspective. *IEEE Trans. Intell. Veh.* **2022**, *7*, 815–837. [CrossRef]
11. Karthikeyan, H.; Usha, G. Real-time DDoS flooding attack detection in intelligent transportation systems. *Comput. Electr. Eng.* **2022**, *101*, 107995. [CrossRef]
12. Derhab, A.; Belaoued, M.; Mohiuddin, I.; Kurniawan, F.; Khan, M.K. Histogram-Based Intrusion Detection and Filtering Framework for Secure and Safe In-Vehicle Networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 2366–2379. [CrossRef]
13. Twardokus, G.; Rahbari, H. Vehicle-to-Nothing? Securing C-V2X Against Protocol-Aware DoS Attacks. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications, London, UK, 2–5 May 2022; pp. 1629–1638. [CrossRef]
14. Abdullah, M.; Raza, I.; Zia, T.; Hussain, S.A. Interest flooding attack mitigation in a vehicular named data network. *IET Intell. Transp. Syst.* **2021**, *15*, 525–537. [CrossRef]
15. Farivar, F.; Haghighi, M.S.; Jolfaei, A.; Wen, S. Covert Attacks Through Adversarial Learning: Study of Lane Keeping Attacks on the Safety of Autonomous Vehicles. *IEEE/ASME Trans. Mechatron.* **2021**, *26*, 1350–1357. [CrossRef]
16. Swessi, D.; Idoudi, H. Comparative Study of Ensemble Learning Techniques for Fuzzy Attack Detection in In-Vehicle Networks. In *Advanced Information Networking and Applications*; AINA 2022. Lecture Notes in Networks and Systems; Barolli, L., Hussain, F., Enokido, T., Eds.; Springer: Cham, Switzerland, 2022; Volume 450, pp. 598–610. [CrossRef]
17. Singh, A.; Singh, B. A Study of the IEEE802.11p (WAVE) and LTE-V2V Technologies for Vehicular Communication. In Proceedings of the 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 9–11 January 2020; pp. 157–160. [CrossRef]
18. *IEEE 802.11p*; IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. IEEE: New York, NY, USA, 2010.
19. Santamaria, A.F.; Sottile, C.; Lupia, A.; Raimondo, P. An efficient traffic management protocol based on IEEE802.11p standard. In Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2014), Monterey, CA, USA, 6–10 July 2014; pp. 634–641. [CrossRef]
20. Qureshi, K.N.; Din, S.; Jeon, G.; Piccialli, F. Internet of Vehicles: Key Technologies, Network Model, Solutions and Challenges with Future Aspects. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 1777–1786. [CrossRef]

21. Liu, Z.; Weng, J.; Ma, J.; Guo, J.; Feng, B.; Jiang, Z.; Wei, K. TCEMD: A Trust Cascading-Based Emergency Message Dissemination Model in VANETs. *IEEE Internet Things J.* **2020**, *7*, 4028–4048. [CrossRef]
22. Guo, J.; Liu, Z.; Tian, S.; Huang, F.; Li, J.; Li, X.; Igorevich, K.K.; Ma, J. TFL-DT: A Trust Evaluation Scheme for Federated Learning in Digital Twin for Mobile Networks. *IEEE J. Sel. Areas Commun.* **2023**, *41*, 3548–3560. [CrossRef]
23. Sedar, R.; Kalalas, C.; Vázquez-Gallego, F.; Alonso, L.; Alonso-Zarate, J. A Comprehensive Survey of V2X Cybersecurity Mechanisms and Future Research Paths. *IEEE Open J. Commun. Soc.* **2023**, *4*, 325–391. [CrossRef]
24. Gupta, S.; Maple, C.; Passerone, R. An Investigation of Cyber-Attacks and Security Mechanisms for Connected and Autonomous Vehicles. *IEEE Access* **2023**, *11*, 90641–90669. [CrossRef]
25. Zhu, P.; Zhu, K.; Zhang, L. Security Analysis of LTE-V2X and A Platooning Case Study. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, 6–9 July 2020; pp. 532–537. [CrossRef]
26. Bajracharya, C. Performance Evaluation for Secure Communications in Mobile Internet of Vehicles with Joint Reactive Jamming and Eavesdropping Attacks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 22563–22570. [CrossRef]
27. da Silva, L.B.; Fernández, E.M.G.; Camponogara, Â. Physical Layer Security Techniques Applied to Vehicle-to-Everything Networks. *arXiv* **2023**, arXiv:2301.05123. [CrossRef]
28. Dibaei, M.; Zheng, X.; Jiang, K.; Abbas, R.; Liu, S.; Zhang, Y.; Xiang, Y.; Yu, S. Attacks and defences on intelligent connected vehicles: A survey. *Digit. Commun. Netw.* **2020**, *6*, 399–421. [CrossRef]
29. Liu, X.; Yang, L. Analysis of Typical Attacks on Intelligent and Connected Vehicle Cyber Security. In *Green, Smart and Connected Transportation Systems: Proceedings of the 9th International Conference on Green Intelligent Transportation Systems and Safety*; Springer: Singapore, 2020; pp. 1039–1048. [CrossRef]
30. Xu, X.; Li, X.; Dong, P.; Liu, Y.; Zhang, H. Robust Reset Speed Synchronization Control for an Integrated Motor-Transmission Powertrain System of a Connected Vehicle Under a Replay Attack. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5524–5536. [CrossRef]
31. Panda, S.; Panaousis, E.; Loukas, G.; Kentrotis, K. Privacy Impact Assessment of Cyber Attacks on Connected and Autonomous Vehicles. In Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES '23), Benevento, Italy, 29 August–1 September 2023; Association for Computing Machinery: New York, NY, USA, 2023; pp. 1–9. [CrossRef]
32. Khan, S.; Sharma, I.; Aslam, M.; Khan, M.Z.; Khan, S. Security Challenges of Location Privacy in VANETs and State-of-the-Art Solutions: A Survey. *Future Internet* **2021**, *13*, 96. [CrossRef]
33. Ghimire, B.; Rawat, D.B. Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 8229–8249. [CrossRef]
34. D'Angelo, G.; Castiglione, A.; Palmieri, F. A Cluster-Based Multidimensional Approach for Detecting Attacks on Connected Vehicles. *IEEE Internet Things J.* **2021**, *8*, 12518–12527. [CrossRef]
35. Khan, J.; Lim, D.-W.; Kim, Y.-S. Intrusion Detection System CAN-Bus In-Vehicle Networks Based on the Statistical Characteristics of Attacks. *Sensors* **2023**, *23*, 3554. [CrossRef] [PubMed]
36. Yang, Y.; Duan, Z.; Tehranipoor, M. Identify a spoofing attack on an in-vehicle CAN bus based on the deep features of an ECU fingerprint signal. *Smart Cities* **2020**, *3*, 17–30. [CrossRef]
37. Islam, R.; Refat, R.U.D.; Yerram, S.M.; Malik, H. Graph-Based Intrusion Detection System for Controller Area Networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 1727–1736. [CrossRef]
38. van Wyk, F.; Wang, Y.; Khojandi, A.; Masoud, N. Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 1264–1276. [CrossRef]
39. Liu, S.; Liu, L.; Tang, J.; Yu, B.; Wang, Y.; Shi, W. Edge Computing for Autonomous Driving: Opportunities and Challenges. *Proc. IEEE* **2019**, *107*, 1697–1716. [CrossRef]
40. Yu, Z.; Liu, Y.; Xie, G.; Li, R.; Liu, S.; Yang, L.T. TCE-IDS: Time Interval Conditional Entropy- Based Intrusion Detection System for Automotive Controller Area Networks. *IEEE Trans. Ind. Inform.* **2023**, *19*, 1185–1195. [CrossRef]
41. Markovitz, M.; Wool, A. Field classification, modeling and anomaly detection in unknown CAN bus networks. *Veh. Commun.* **2017**, *9*, 43–52. [CrossRef]
42. Olufowobi, H.; Ezeobi, U.; Muhati, E.; Robinson, G.; Young, C.; Zambreno, J.; Bloom, G. Anomaly Detection Approach Using Adaptive Cumulative Sum Algorithm for Controller Area Network. In *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec '19)*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 25–30. [CrossRef]
43. Young, C.; Olufowobi, H.; Bloom, G.; Zambreno, J. Automotive Intrusion Detection Based on Constant CAN Message Frequencies Across Vehicle Driving Modes. In *Proceedings of the ACM Workshop on Automotive Cybersecurity (AutoSec '19)*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 9–14. [CrossRef]
44. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]
45. Ayyagari, M.R.; Kesswani, N.; Kumar, M.; Kumar, K. Intrusion detection techniques in network environment: A systematic review. *Wirel. Netw.* **2021**, *27*, 1269–1285. [CrossRef]

46. Tariq, S.; Lee, S.; Woo, S.S. CANTransfer: Transfer learning based intrusion detection on a controller area network using convolutional LSTM network. In Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20), Brno, Czech Republic, 30 March–3 April 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1048–1055. [CrossRef]
47. Longari, S.; Valcarcel, D.H.N.; Zago, M.; Carminati, M.; Zanero, S. CANnolo: An Anomaly Detection System Based on LSTM Autoencoders for Controller Area Network. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1913–1924. [CrossRef]
48. Duan, X.; Yan, H.; Tian, D.; Zhou, J.; Su, J.; Hao, W. In-Vehicle CAN Bus Tampering Attacks Detection for Connected and Autonomous Vehicles Using an Improved Isolation Forest Method. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 2122–2134. [CrossRef]
49. Ilango, H.S.; Ma, M.; Su, R. A misbehavior detection system to detect novel position falsification attacks in the Internet of Vehicles. *Eng. Appl. Artif. Intell.* **2022**, *116*, 105380. [CrossRef]
50. Vijayanand, R.; Devaraj, D.; Kannapiran, B. Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Comput. Secur.* **2018**, *77*, 304–314. [CrossRef]
51. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A Multitiered Hybrid Intrusion Detection System for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 616–632. [CrossRef]
52. Odaibo, S. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv* **2019**, arXiv:1907.08956.
53. Ahmed, M.; Seraj, R.; Islam, S.M.S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* **2020**, *9*, 1295. [CrossRef]
54. Pradipta, G.A.; Wardoyo, R.; Musdholifah, A.; Sanjaya, I.N.H.; Ismail, M. SMOTE for Handling Imbalanced Data Problem: A Review. In Proceedings of the 2021 Sixth International Conference on Informatics and Computing (ICIC), Jakarta, Indonesia, 3–4 November 2021; pp. 1–8. [CrossRef]
55. Curtis, A.; Smith, T.; Ziganshin, B.; Elefteriades, J. The mystery of the Z-score. *Aorta* **2016**, *4*, 124–130. [CrossRef] [PubMed]
56. Azhagusundari, B.; Thanamani, A.S. Feature selection based on information gain. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2013**, *2*, 18–21.
57. Breiman, L.; Friedman, J.; Olshen, R.A.; Stone, C.J. *Classification and Regression Trees*; Routledge: London, UK, 2017.
58. Mitchell, R.; Frank, E. Accelerating the XGBoost algorithm using GPU computing. *PeerJ Comput. Sci.* **2017**, *3*, e127. [CrossRef]
59. Yoshua, B.; Grandvalet, Y. No unbiased estimator of the variance of k-fold cross-validation. *J. Mach. Learn. Res.* **2004**, *5*, 1089–1105.
60. Alpaydin, E. *Machine Learning*; MIT Press: Cambridge, MA, USA, 2021.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.