Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2026.075959

ARTICLE

EDESC-IDS: An Efficient Deep Embedded Subspace Clustering-Based Intrusion Detection System for the Internet of Vehicles

Lixing Tan[1,2], **Liusiyu Chen**[1], **Yang Wang**[1], **Zhenyu Song**[1,*] and **Zenan Lu**[1,3,*]

[1]College of Information Engineering, Taizhou University, Taizhou, 225300, China
[2]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China
[3]School of Computer Science, Fudan University, Shanghai, 200433, China

*Corresponding Authors: Zhenyu Song. Email: songzhenyu@tzu.edu.cn; Zenan Lu. Email: luzenan@tzu.edu.cn

Received: 11 November 2025; Accepted: 23 December 2025; Published: 12 March 2026

ABSTRACT: Anomaly detection is a vibrant research direction in controller area networks, which provides the fundamental real-time data transmission underpinning in-vehicle data interaction for the internet of vehicles. However, existing unsupervised learning methods suffer from insufficient temporal and spatial constraints on shallow features, resulting in fragmented feature representations that compromise model stability and accuracy. To improve the extraction of valuable features, this paper investigates the influence of clustering constraints on shallow feature convergence paths at the model level and further proposes an end-to-end intrusion detection system based on efficient deep embedded subspace clustering (EDESC-IDS). Following the standard learning approach, continuous messages are encoded into two-dimensional data frames via a frame builder, which are then input into an extended convolutional autoencoder for extracting shallow features from high-dimensional data. On this basis, the dual constraints of these output features and the embedding clustering module facilitate end-to-end training of the EDESC-IDS in various attack scenarios. Extensive experimental results show that such a system exhibits significant detection performance on four types of attack datasets, including DoS, Gear, Fuzzy, and RPM, with precision, recall, and F1 scores consistently above 97.79%, while maintaining a false negative rate (FNR) and an error rate (ER) below 2.22%.

KEYWORDS: Internet of vehicles; control area network; anomaly detection; unsupervised learning; deep embedded subspace clustering; extended convolutional autoencoder

1 Introduction

The 5G technology enables advanced applications such as autonomous driving and high-definition video streaming, while automotive electronics facilitate the replacement of traditional mechanical linkages by electronic control units (ECUs) in modern vehicular systems [1,2]. This dual evolution has driven the emergence of diverse in-vehicle network (IVN) architectures, with the controller area network (CAN), local interconnect network, and FlexRay emerging as predominant protocols [3].

In the ECU communication paradigm, a CAN represents a meticulously engineered message broadcast system optimized for operational efficiency, transmission reliability, and cost efficiency. Its original specification omitted critical security provisions including cryptographic protection, access control mechanisms, and message authentication protocols [4]. This design deficiency inherently restricts authentication and encryption capabilities within the CAN protocol stack [5]. Such vulnerabilities allow malicious actors to conduct network intrusions, data exfiltration, and unauthorized vehicle control through CAN buses [6].

These systemic weaknesses fundamentally undermine IVN security, necessitating proactive defence strategies [7]. Therefore, anomaly detection for CAN bus traffic has become a critical research focus in automotive cybersecurity.

In terms of traffic anomaly detection, current research methodologies have exhibited diverse developmental trends that can be categorized into signature-based detection [8], statistics-based detection [9], physical feature-based detection [10], and machine learning approaches [11]. However, these detection technologies present varying degrees of limitations in real-world engineering applications. Signature-based intrusion detection systems (IDSs) rely on known attack signature databases, which limits their detection capabilities. Traditional statistical methods face technical challenges regarding baseline parameter drift in dynamic environments. Physical feature-based detection is limited by vehicular physical-layer signal characteristic instability and electromagnetic interference susceptibility. The last approach predominantly employs supervised learning methods to classify abnormal IVN data [12]. Compared with unsupervised learning, this approach lacks relevant mechanisms for handling unlabelled attack scenarios inherent to the former. Autoencoders, as prominent representation learning architectures in unsupervised domains, play critical roles in constructing low-dimensional feature spaces. Notably, conventional autoencoders may exhibit insufficient feature compression when the information feature complexity exceeds their learning capacity, potentially impairing discriminative feature extraction efficacy. This limitation critically restricts classification performance when processing high-dimensional traffic data.

Although unsupervised learning methods have already been explored within intrusion detection, they also suffer from insufficient temporal and spatial constraints on shallow features, leading to fragmented feature representations that compromise model stability and accuracy. To provide more stable and efficient clustering performance, this paper proposes a novel intrusion detection framework incorporating efficient deep embedded subspace clustering (EDESC), which was introduced in [13]. The framework establishes a unified architecture that synergistically combines clustering with feature learning, where clustering processes and feature representations exhibit mutual reinforcement. Specifically, autoencoder-derived linear compact low-dimensional features enhance clustering precision, whereas optimized clustering reciprocally improves feature learning in autoencoders, thus increasing the model's representational capacity.

The main contributions of this paper are summarized as follows:

(1) We propose an intrusion detection system based on efficient deep embedded subspace clustering (EDESC-IDS), which extracts valuable spatiotemporal features from streaming data and enforces multiperspective constraints on low-dimensional feature learning, thereby achieving the precise classification of anomalous attack data.
(2) To effectively leverage low-dimensional feature representations inherent in the data, we design an extended convolutional autoencoder (ECAE) specifically engineered to extract spatiotemporal characteristics from CAN bus data. To establish the connection between this autoencoder and the clustering module, the latter provides an adaptive hybrid pooling module to optimally condense informational dimensions while preserving critical temporal-spatial correlation.
(3) We introduce a Bayesian-based automatic hyperparameter optimization (BAHO) algorithm and conduct a rigorous evaluation of the EDESC-IDS operational efficacy via a subset from the hacker and countermeasure research laboratory (HCRL [14]) dataset. Furthermore, its feasibility for real-world vehicular IoT device applications is discussed. Compared with five state-of-the-art detection models, the EDESC-IDS achieves superiority in identifying four prevalent attack categories, with F1-scores that consistently stabilize at approximately 97.79%.

Paper overview

The rest of this paper is structured as follows. Section 2 reviews the research progress of in-vehicle anomaly detection. Section 3 provides the background of CANs and examines characteristic attack scenarios. Section 4 elaborates on the algorithmic framework of the proposed EDESC-IDS. The experimental results are presented and analysed in Section 5. Finally, Section 6 concludes with a summary of our contributions.

2 Related Work

The transformative progress in artificial intelligence has propelled the widespread adoption of machine learning techniques for anomaly detection within vehicular networks [15]. This trend has been further amplified by exponentially expanding data volumes across connected transportation ecosystems, cementing data-driven machine learning approaches as a cornerstone methodology in anomaly detection research for intelligent vehicular systems. These approaches can be roughly divided into supervised and unsupervised methods. Their characteristics are itemized in Table 1.

Table 1: Comparison of IVN anomaly detection methods

Reference	Feature engineering	Known attack	Model optimization	Low complexity	Real-time	Unsupervised learning
Tariq et al. [16]	✓	✓			✓	
Ashraf et al. [17]	✓	✓	✓			
Qin et al. [18]	✓	✓	✓	✓		
Alqahtani and Kumar [19]	✓	✓				
He et al. [20]	✓	✓				
Wang et al. [21]	✓	✓		✓	✓	
Cheng et al. [22]	✓	✓	✓	✓	✓	
Taylor et al. [23]	✓	✓				✓
Duan et al. [24]		✓				✓
Amarbayasgalan et al. [25]	✓	✓				✓
Barletta et al. [26]	✓	✓				✓
Cheng et al. [27]	✓	✓	✓	✓	✓	✓
Jeong et al. [28]	✓	✓	✓		✓	✓
Gao et al. [29]	✓	✓				✓
This Paper	✓	✓	✓	✓	✓	✓

Supervised learning methods typically require extensively labeled training datasets, which limits their capability to identify most zero-day attacks. For instance, Tariq et al. [16] developed a long short-term memory (LSTM) artificial neural network-based framework for CAN bus attack detection, which substantially improved detection performance through comprehensive spatiotemporal feature extraction. Although this framework enhances system capability in identifying unknown attacks, its transfer learning strategy is highly dependent on prior knowledge, potentially compromising model efficacy and generalizability. Ashraf et al. [17] and Qin et al. [18] introduced an LSTM-based and a threshold-based anomaly detection algorithm for CAN bus systems. However, their algorithms focus solely on reconstructing normal message sequences and leveraging reconstruction errors for threshold-based detection, failing to fully exploit the latent value of extracted low-dimensional spatiotemporal features. Alqahtani and Kumar [19] developed a hybrid convolutional neural network (CNN) model that effectively captures spatiotemporal patterns in both normal and anomalous network traffic, achieving superior detection accuracy on the HCRL benchmark dataset. He et al. [20] proposed a wavelet kernel network with omni-scale convolution capable of adaptively selecting optimal scales for high-frequency signal processing, thereby demonstrating enhanced generalization performance in vehicular network intrusion detection. Wang et al. [21] devised a novel IDS by integrating depth-wise separable convolution, a bottleneck architecture and squeeze-and-excitation modules, achieving high model accuracy while significantly reducing parametric overhead. For example, Cheng et al. [22] developed a knowledge distillation-based semantic knowledge transfer architecture incorporating a weighted sampler and focal loss mechanisms to mitigate the class imbalance-induced long-tail effect, thereby enhancing parallel detection efficacy and operational efficiency in IVNs. To summarize, the excessive dependence of supervised methods on labeled data limits the exploitation of their learning capability in application scenarios with insufficient labeled data.

To address this limitation, researchers have proposed a diverse array of unsupervised methodologies. Building upon the unsupervised support vector machine (SVM) methodology, Taylor et al. [23] proposed a one-class SVM (OC-SVM) anomaly detection algorithm for identifying message frequency deviations. Although this algorithm partially alleviates label dependency, its applicability remains confined to frequency-based anomaly attacks, demonstrating limited capacity in extracting deeper semantic features while exhibiting elevated false positive rates. To quantify data anomaly severity, Duan et al. [24] introduced an isolation forest variant with a data mass for data tampering attack detection. A fundamental limitation shared by both approaches lies in their insufficient consideration of valuable shallow feature extraction from raw data.

To optimize low-dimensional feature learning in CAN systems, Amarbayasgalan et al. [25] devised a deep autoencoders (DAE) model by combining a multilayer perceptron with density-based clustering. The exclusive use of dense layers in their autoencoder design inherently restricts its ability to capture the spatiotemporal characteristics of CAN messages. Barletta et al. [26] proposed a Kohonen self-organizing map (SOM)-driven unsupervised intrusion detection framework that synergistically integrates SOM with k-means clustering through distance metric optimization. Cheng et al. [27] presented a unified framework that integrates streaming clustering with convolutional autoencoders for unsupervised anomaly detection. Jeong et al. [28] developed a multimodal autoencoder-based real-time vehicular network observer. However, these methods lack effective constraints on the learning direction of data during the clustering process of mapping high-dimensional data into low-dimensional spaces, resulting in the autoencoder's partial inability to extract valuable shallow features effectively. Furthermore, a representative case is the CNN-BiLSTM hybrid architecture with multiple-attention mechanisms proposed by Gao et al. [29], which synergistically integrates the spatiotemporal feature extraction capacity of CNNs with the strengths of bidirectional LSTM sequential data modelling.

While existing intrusion detection systems have achieved considerable advancements, they persist in facing critical deficiencies including high computational costs, reliance on labelled training data, and constrained real-time responsiveness. To address these deficiencies, this paper presents the EDESC-IDS, a deep-embedding subspace clustering approach, to enhance clustering module stability and reliability while enabling effective high-speed streaming data processing. This approach establishes subspace bases through deep feature representations and develops an optimized clustering loss mechanism. By projecting data into a lower-dimensional latent space, it achieves efficient and flexible feature extraction from large-scale streaming data, thus providing new insights for the cybersecurity research of intelligent vehicular networks.

3 Preliminaries

To ensure theoretical self-containment, this section recalls controller area networks and their attack scenarios.

3.1 Controller Area Networks

The automotive industry is undergoing a revolutionary transformation through the integration of advanced computing and communication technologies, significantly enhancing user convenience and driving safety [30]. In IVNs, information transmission is predominantly conducted via intercommunication among ECUs. The CAN bus has been extensively adopted for inter-ECU communication within automotive systems because of its cost-effectiveness and reliability, establishing it as the standard protocol for IVNs [31]. Unlike the well-known TCP/IP network communication on the internet, the CAN is a frame-based broadcast communication where each connected ECU node transmits data in structured frame formats, with message prioritization governed by identifier arbitration mechanisms [3].

The CAN message format, as shown in Fig. 1, comprises a series of fields such as start of frame (SOF), arbitration ID, data length code (DLC), DATA, cyclic redundancy check (CRC), acknowledgment (ACK), and end of frame (EOF) [32]. The blue-highlighted ID field and orange-marked DATA field are the core components of CAN messages. The Arbitration ID field determines the frame prioritization through a numerical hierarchy, where messages with lower ID values possess higher transmission priority. According to the CAN bus protocol's arbitration mechanism, higher-priority messages are transmitted first during broadcast collisions [33]. The DATA field carries the actual payload, which represents information transferred between nodes. Among the remaining fields, SOF marks the start of a CAN message, DLC specifies the payload length of DATA, CRC serves as an error detection mechanism, ACK confirms the presence of errors in the message, and EOF signals the end of the CAN frame. Collectively, all these fields implement ISO 11898-compliant error-checking mechanisms [34], ensuring reliable communication in safety-critical automotive systems.

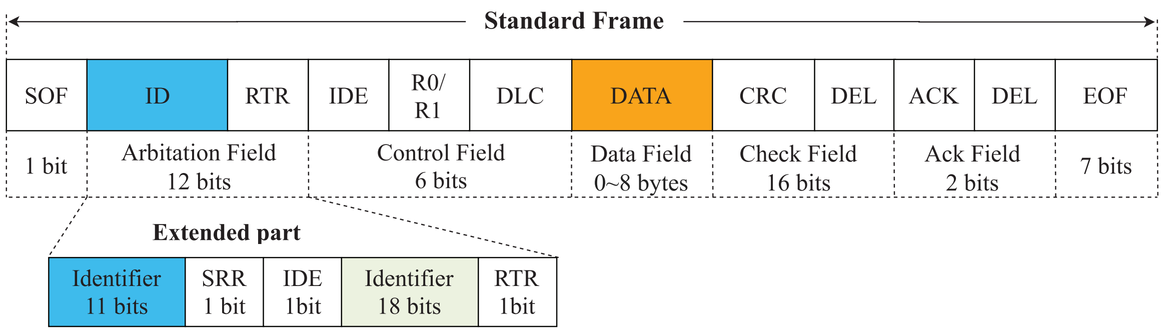

Figure 1: The dual-message architecture of a CAN

Fundamentally, the dual-message architecture of a CAN, comprising standard and extended frames with distinct arbitration field configurations, introduces inherent vulnerabilities in vehicular security models. While the expanded arbitration field of the extended frame ostensibly addresses scalability demands from proliferating ECU nodes, this evolution paradoxically creates critical attack surfaces. CAN 2.0.A devices exclusively employ standard frames with 11-bit identifiers, whereas CAN 2.0.B-compliant implementations support dual-format operations through the incorporation of both standard and extended frames with 29-bit identifiers [32]. This protocol coexistence systematically amplifies threat scenarios, specifically expanding the attack surface for ID spoofing via increased exploitable identifier permutations, creating timing channel vulnerabilities during frame-type arbitration through deterministic priority-handling delays, and fragmenting security enforcement across incompatible protocol versions, thereby impeding unified protective measures.

3.2 *Attack Scenarios*

In recent years, vehicles have introduced a series of additional interfaces to address the growing demand for vehicle services, providing basic smart functionalities. However, this expansion introduces security challenges and vulnerabilities. This subsection focuses on three common attack types that target modern in-vehicle CAN buses, DoS, fuzzy and spoofing attacks, all of which are categorized as tamper-based intrusions. In our attack scenarios, we assume that the attacker has compromised at least one ECU. Each compromised ECU achieves direct access to the internal CAN bus, enabling message sniffing and malicious data injection. Fig. 2 illustrates these three attack scenarios.

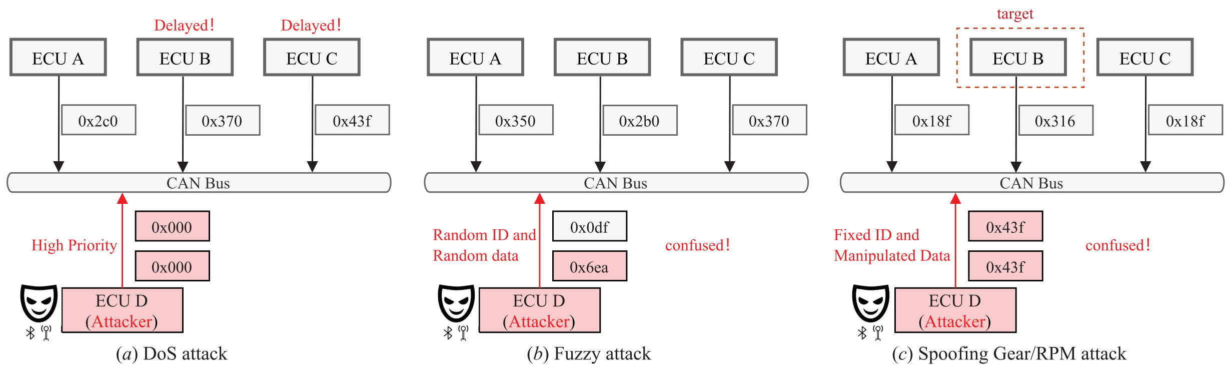

Figure 2: Three attack scenarios

DoS Attack: The goal of this attack is to disrupt the CAN bus by overwhelming it with excessive messages [35]. Leveraging the arbitration mechanism of the CAN protocol, the attacker floods the bus with a large volume of high-priority IDs carrying invalid or irrelevant messages (e.g., the 0x000 signals in Fig. 2a, causing congestion and potentially leading to denial of service (DoS) for legitimate communication. This attack degrades system performance, causes delays, and blocks normal communication between legitimate nodes even without detailed knowledge of the IVN. In real-world scenarios, such attacks pose a critical threat to availability, as driver commands might go unrecognized.

Fuzzy Attack: A fuzzy attack aims to introduce uncertainty and unreliability into the CAN bus by manipulating the ID and DATA fields (e.g., the 0x6ea signal in Fig. 2b. This highly randomized attack tricks the bus into accepting unexpected data, potentially causing malfunctions in various vehicle components. In real-world scenarios, all nodes may receive massive random messages, leading to unpredictable vehicle behaviours such as fluctuating tire speeds, unstable gear shifts, or random light flickering. Fuzzy attacks can pose severe risks such as signal misinterpretations, erroneous control decisions, and system failures.

Spoofing Attack: The feasibility of spoofing attacks in the CAN protocol depends on the characteristics of plaintext broadcast communication. In real-world scenarios, spoofing can trick nodes into performing unsafe actions, such as executing fake Gear position commands or accepting false revolutions per minute (RPM) readings. Attackers execute such attacks by forging messages containing legitimate IDs (e.g., the 0x43f signal in Fig. 2c, impersonating valid nodes to deceive others into accepting fraudulent commands [36]. Before starting this attack, the attacker is required to eavesdrop on critical messages and use reverse engineering to identify the significance of transmitted data. Compared with DoS and fuzzy attacks, spoofing attacks demonstrate significantly greater sophistication and demand higher greater technical complexity for execution.

4 Our Proposed Model: EDESC-IDS

On the basis of the temporal-spatial characteristics of CAN buses and their vulnerability to attacks, this section proposes an EDESC-IDS model integrated with a deep embedded subspace clustering approach, as illustrated in Fig. 3. This model includes data preprocessing, ECAE, and EDESC modules. Its strength lies in effectively combining the autoencoder's capability for shallow feature learning with the constraint mechanisms of deep embedded subspace clustering, enabling efficient end-to-end comprehensive training on CAN bus data.

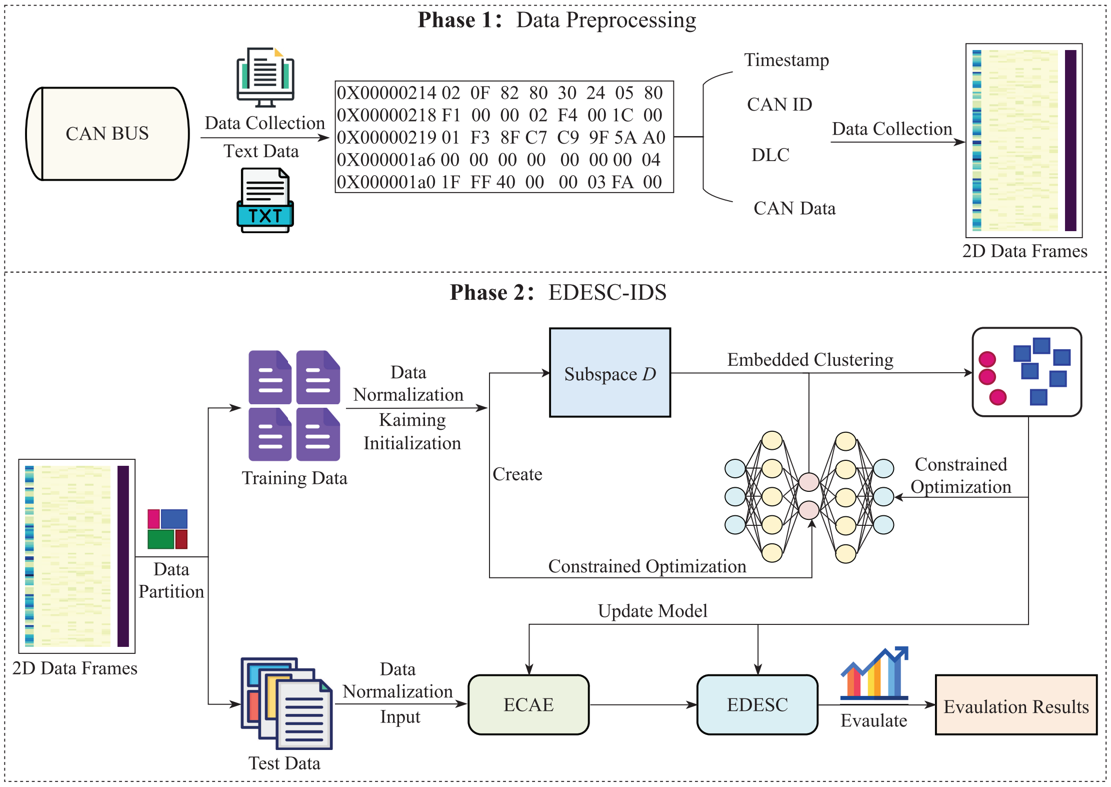

Figure 3: The framework of our EDESC-IDS

4.1 Overview

Unlike most existing IDSs that isolate feature extraction from clustering, a design approach that causes clustering efficacy to depend critically on feature quality, our EDESC-IDS integrates latent representation learning via autoencoders with efficient deep embedding-based subspace clustering. This integrated framework simultaneously optimizes shallow feature extraction and clustering outcomes through three sequential phases: data preprocessing, offline training, and online detection (see Algorithm 1 for more details).

Algorithm 1: EDESC-IDS Detection Process Algorithm

```
Input: Attack dataset $X_a$, Normal dataset $X_n$, Test dataset $X_t$, and actual label $y$.
Output: performance metrics.
1: #Data Preprocessing Phase:
2: Convert $X_a$, $X_n$ to $X_{af}$, $X_{nf}$ via Frame builder    //Frame construction
3: Normalization: $X_{af}$, $X_{nf}$ ←z-score normalization($X_{af}$, $X_{nf}$)    //z-score normalization
4: #Offline Training Phase:
5: Initialize ECAE ← Kaiming_init(ECAE)    //Kaiming initialization
6: while ECAE not converged do
7:     for each $X \in X_{nf}$ do
8:         $X_{de} \leftarrow$ ECAE($X$)    //Obtain shallow features of the normal dataset in ECAE
9:         $\zeta \leftarrow$ MSE($X_{de}, X$)
10:        $\zeta$.backward()
11:    end for
12: end while
13: while EDESC not converged do
14:     for each $X \in X_{af}$ do
15:         $X_{ae} \leftarrow$ ECAE($X$)    //Extract shallow features of the attack dataset from ECAE
16:         $\zeta \leftarrow \zeta_{\text{EDESC}}(X_{ae})$    //Optimize the model with EDESC-based clustering loss
17:         $\zeta$.backward()
18:     end for
19: end while
20: #Online Detection Phase:
21: for each $X \in X_t$ do
22:     $\dot{y} \leftarrow$ EDESC-IDS($X$)    //Model prediction
23:     res.append(metrics($\dot{y}$, $y$))
24: end for
```

During data preprocessing, the system first collects CAN bus traffic data and converts these raw data into two-dimensional frames via a frame builder, followed by z-score normalization for standardization before partitioning into training, validation, and test sets. In offline training, the system initializes the ECAE via Kaiming initialization and trains it on normal datasets, enabling the model to sufficiently capture the data characteristics of the CAN bus until convergence. Finally, during online detection, the system evaluates the trained EDESC-IDS to obtain performance metrics.

4.2 Data Preprocessing

In deep learning, training data quality significantly impacts model training time and classification performance. However, real-world CAN bus data collected through capture tools often have uneven distributions and incompleteness, while containing substantial irrelevant information and noise. These issues make it difficult for intrusion detection systems to learn authentic data patterns. To address this issue, data preprocessing serves as a mandatory preparatory step for the EDESC-IDS.

This paper utilizes the publicly accessible HCRL intrusion dataset, acquired from a Hyundai YF Sonata test vehicle [14]. During data acquisition, researchers interfaced with the OBD-II port and employed a Raspberry Pi 3 for CAN traffic monitoring. To generate anomalous data records, four attack types were injected via the CAN bus: DoS, Fuzzy, spoofing Gear, and spoofing RPM. Each dataset entry comprises a timestamp, CAN ID, DLC, and an 8-byte data field. The HCRL dataset details are listed in Table 2.

Table 2: Statistical analysis of CAN bus message distribution

Attack type	Total messages	Normal messages	Injected messages
DoS	3,665,771	3,078,250	587,521
Fuzzy	3,838,860	3,374,013	491,847
Spoofing gear	4,443,142	3,845,890	597,252
Spoofing RPM	4,621,702	3,966,805	654,897

During data processing, all the hexadecimal values in the IDs and data fields are converted to the decimal format. Missing bytes are filled with "00". Notably, the timestamp characteristics are strongly correlated with the number of cyberattack simulation cycles. However, their dependence on data collection frequency causes model learning and prediction biases. To address this, we replace the timestamp features with the average time difference. The time difference $Diff_i$ is defined as follows.

$$Diff_i \triangleq \begin{cases} \dfrac{|T_i - T_{i-1}| + |T_{i+1} - T_i|}{2} & \text{if } i > 1, \\ \dfrac{|T_{i+1} - T_i|}{2} & \text{if } i = 1. \end{cases} \tag{1}$$

Compared with timestamps, the average temporal difference $Diff_i$ aggregates localized sequential contexts to mitigate timing disruptions from clock drift and network latency. By doing so, it enhances the robustness of modeling abnormal message transmission intervals.

4.3 Frame Builder

Focusing on the spatiotemporal characteristics of CAN buses, this section constructs features with temporal-spatial correlations. Specifically, the frame builder aggregates consecutive data into time windows on the basis of time series length. These windows are then converted into two-dimensional data frames with short-term temporal dependencies. This construction method effectively captures spatiotemporal relationships within data, enabling efficient detection of system anomalies as illustrated in Fig. 4.

Let tsl be the time series length. The frame builder constructs a two-dimensional data frame (width = 11, length = tsl) through the stacking of consecutive CAN messages. Given the total number of CAN messages as a, the quantity m of generated data frames is calculated as:

$$m = \left\lceil \frac{a}{tsl} \right\rceil. \tag{2}$$

In the above equation, tsl measures the size of the two-dimensional data frame, and its construction method aligns well with the high-frequency data flow characteristics of the CAN bus. For simplicity, the features of the j-th time step f_i are denoted as:

$$T(f_j) = \left\{ f_j(t_1), f_j(t_2), \cdots, f_j(t_i), \cdots, f_j(t_{11}) \right\}, \tag{3}$$

where $f_j(t_i)$ represents the i-th feature in the j-th time step of the two-dimensional data frame $T(f_j)$.

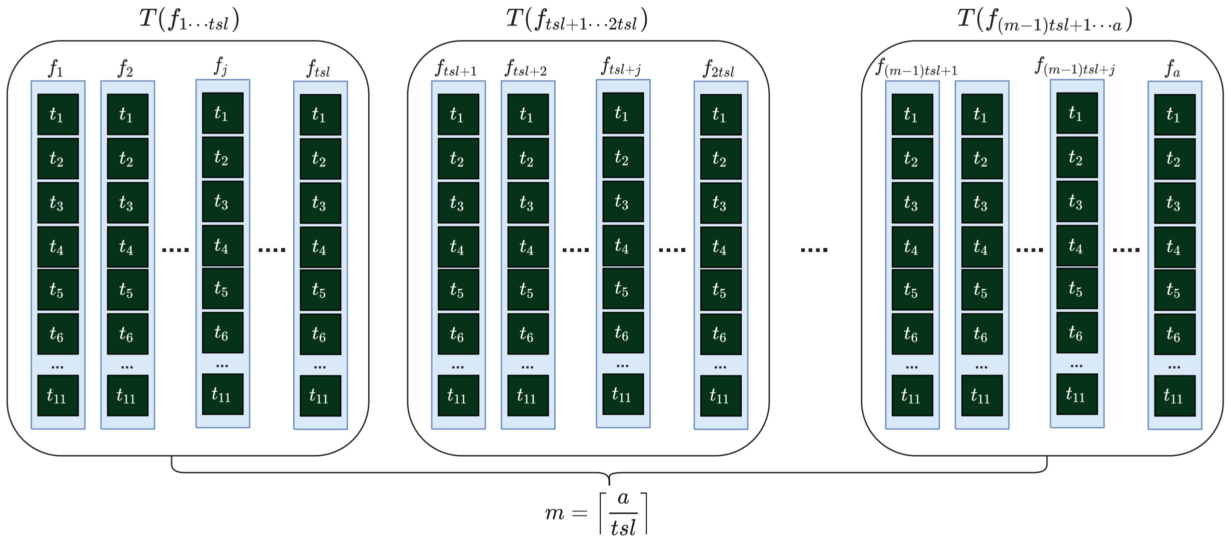
$T(f_{1\cdots tsl})$
$T(f_{tsl+1\cdots 2tsl})$
$T(f_{(m-1)tsl+1\cdots a})$
f_1 f_2 f_j f_{tsl}
f_{tsl+1} f_{tsl+2} f_{tsl+j} f_{2tsl}
$f_{(m-1)tsl+1}$ $f_{(m-1)tsl+j}$ f_a
t_1 t_2 t_3 t_4 t_5 t_6 t_{11}
$m = \left\lceil \frac{a}{tsl} \right\rceil$

Figure 4: Construction of a two-dimensional data frame

The equation below provides symbol notations for all feature domains.

$$T(f_{1\cdots n}) = \{T(f_1), T(f_2), \cdots, T(f_i), \cdots, T(f_n)\}, \tag{4}$$

where $T(f_{1\cdots n})$ represents a two-dimensional data frame with a custom time series length n. Since all selected feature domains inherently contain temporal properties, $T(f_{1\cdots n})$ consequently exhibits temporal characteristics.

Let L_j be the label of the feature at the j-th time step in a two-dimensional data frame. The data frame $Label$ is then defined as, for each time step j,

$$Label = \begin{cases} 0 & \text{if } \forall j\,(L_j = 0), \\ 1 & \text{if } \exists j\,(L_j = 1). \end{cases} \tag{5}$$

Intuitively, if all time series' label values are normal, the label of the entire data frame is normal; otherwise, it is abnormal. Next, we describe the information flow constructed from two-dimensional data frames and its standardization. As shown below, the information flow essentially consists of a series of data frames.

$$Stream(M) = \{T_1(f_{1\cdots n}), T_2(f_{1\cdots n}), \cdots, T_i(f_{1\cdots n}), \cdots, T_m(f_{1\cdots n})\}, \tag{6}$$

where m is the number of two-dimensional data frames generated by the information data stream M and $T_i(f_{1\cdots n})$ represents all feature domains of the i-th data frame.

To balance the comparability between features with larger and smaller initial data ranges, the two-dimensional data frame applies z-score normalization for standardization, as shown below:

$$x' = \frac{x - \mu}{\sigma}, \tag{7}$$

$$\mu = \frac{1}{n \times 11} \sum_{i=1}^{n} \sum_{k=1}^{11} T_i\,(f_k), \tag{8}$$

$$\sigma = \sqrt{\frac{1}{n \times 11} \sum_{i=1}^{n} \sum_{k=1}^{n} \left(T_i\left(f_k\right) - \mu\right)^2}, \tag{9}$$

where x represents the original data, x' represents the normalized data, and μ and σ denote the mean value and standard deviation of the original data, respectively.

4.4 Extended Convolutional Autoencoder

Traditional convolutional autoencoders generally fail to model temporal dynamics [26]. To bridge this gap, we introduce an adaptive hybrid pooling strategy coupled with stacked dilated convolutions, constructing an extended convolutional autoencoder (ECAE) module. This design specifically enables receptive field expansion while retaining key spatiotemporal features in CAN messages. During stream clustering, given the real-time nature and large-scale characteristics of input data, batches are typically set to small or medium sizes to better capture streaming data patterns. Notably, these data may contain various types of noise.

As shown in Fig. 5, the ECAE encoder module contains two convolutional blocks. Each block includes a dilated convolution layer, a dropout layer, a normalization layer, and LeakyReLU activations. The dropout layer randomly deactivates neurons, disrupting coadaptation to prevent overfitting and enhance generalization. The normalization layer accelerates training and reduces the dependence on initialization, whereas LeakyReLU improves the model's nonlinear modelling capability.

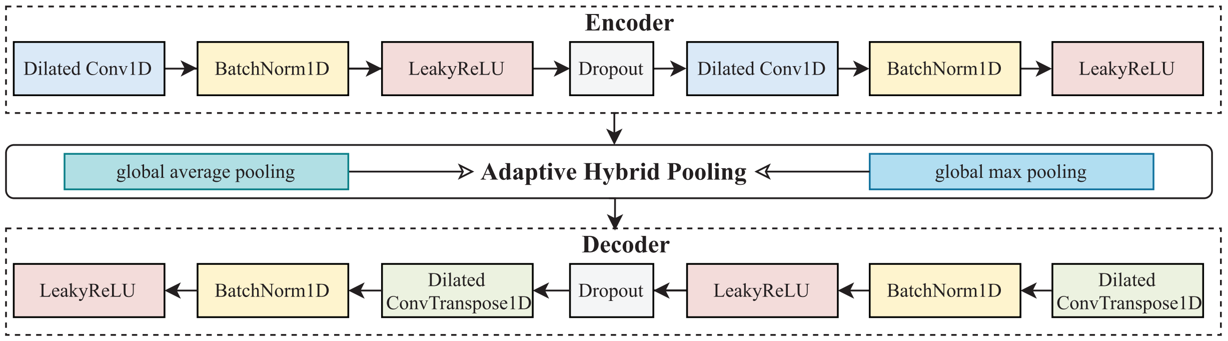

Figure 5: The Encoder-Decoder architecture of ECAE

The dilated convolution layers are the core components of the ECAE decoder module, where the decoder uses them to achieve sampling and input reconstruction. These layers mainly extract local spatial features, and their inclusion helps the model capture long-term historical dependencies in time series. The detailed structure of the dilated convolution is shown in Fig. 6.

For any output element at position s, the computation of its convolution output signal $F_d(s)$ is given by the following equation:

$$F_d(s) = \sum_{i=0}^{k} f(i) \times x_{s+d\times i}, \tag{10}$$

where d is the dilation factor, k is the filter size, $s + d \times i$ denotes the direction, and $f(i)$ defines the elements of the expanded convolution kernel.

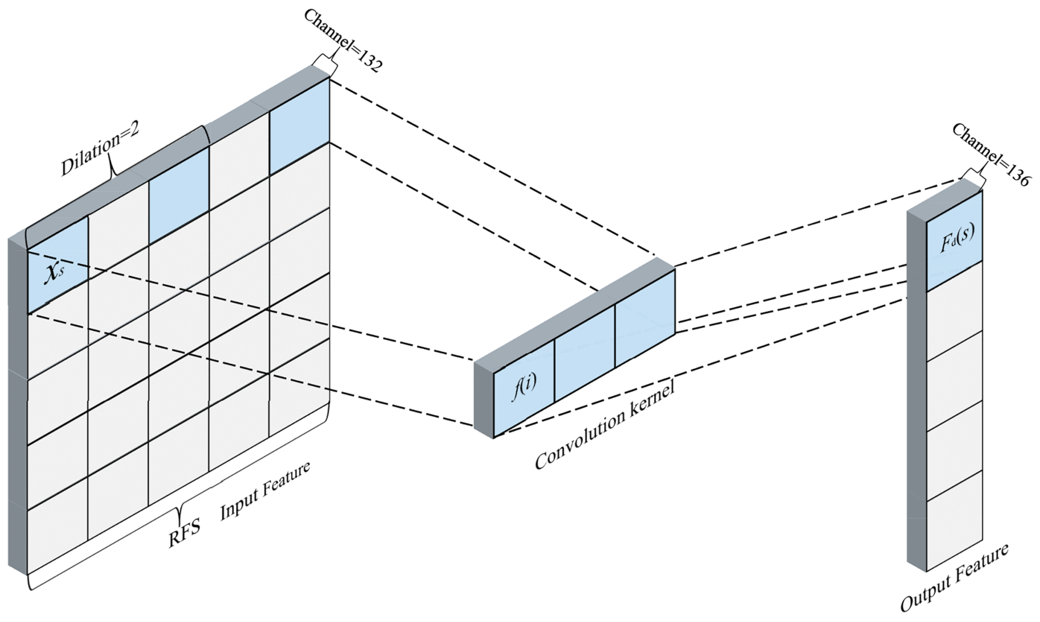

Figure 6: An extended convolutional module with spatial feature extraction

To quantify the range of captured information, the following formula defines the receptive field size (RFS) of the convolutional kernel's input features.

$$RFS = (k-1)\sum_{i=1}^{L} d_i + 1, \tag{11}$$

where L is the number of hidden layers and represents the dilation factor of the i-th hidden layer. To capture more temporal information, the factor d typically increases exponentially with the network's depth.

To facilitate the generation of lower-dimensional nonlinear features by the ECAE, we introduce an adaptive hybrid pooling strategy combining global average pooling and max pooling. This strategy employs a mechanism that incorporates learnable weight parameters, nonlinear mapping via sigmoid functions constraining weights to $[0, 1]$, and complementary constraints to dynamically optimize the relative contribution weights of these two pooling operations. Compared with single pooling methods, this hybrid approach effectively captures complex patterns in time series data while improving the model's discriminative power and generalization capability in temporal tasks.

As shown in Fig. 7, to increase the generalizability of the ECAE model and prevent overfitting, weight decay is applied to all network parameters to improve performance. The weight decay term defined below prevents excessive weight values. Essentially, it represents the sum of all weight parameters in the model.

$$J_{wd} = \frac{\lambda}{2}\sum_{k=1}^{m} \left\| W_k^2 \right\|, \tag{12}$$

where λ denotes the weight decay coefficient, m represents the number of all weight layers in the model, and W_k denotes the weight matrix of the k-th layer.

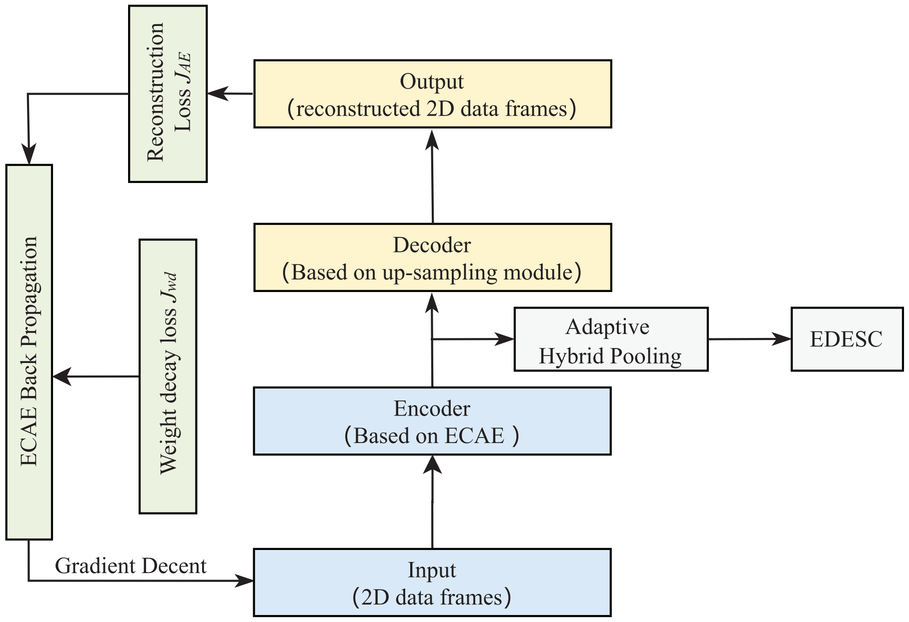
Reconstruction Loss J_{AE}
Output (reconstructed 2D data frames)
Decoder (Based on up-sampling module)
Adaptive Hybrid Pooling
EDESC
ECAE Back Propagation
Weight decay loss J_{wd}
Encoder (Based on ECAE)
Gradient Decent
Input (2D data frames)

Figure 7: Operation mechanism of ECAE

4.5 Efficient Deep Embedded Subspace Clustering

In traditional autoencoder structures, the loss function typically focuses only on the reconstruction loss of raw data, ignoring the importance of clustering for feature optimization. Therefore, this paper introduces an end-to-end clustering method, namely, efficient deep embedded subspace clustering (EDESC) [13]. Compared with conventional subspace clustering methods that rely on iterative self-expression mechanisms (resulting in high computational complexity), EDESC outperforms traditional self-expression frameworks with linear time and space complexity, enabling the efficient processing of streaming datasets. This characteristic strongly aligns with the requirements for handling large-scale real-time data generated by CAN bus systems.

To establish dual learning objectives for the optimization problem, the EDESC model creates dual constraints between the low-dimensional mapping and subspace modules. By learning the data's embedded representations through the autoencoder and constructing subspace affinity vectors with subspace structures, the model obtains standardized affinity subspaces. Building on this, an effective update strategy is established under the combined supervision of subspace loss and reconstruction loss, with specific steps illustrated in Fig. 8.

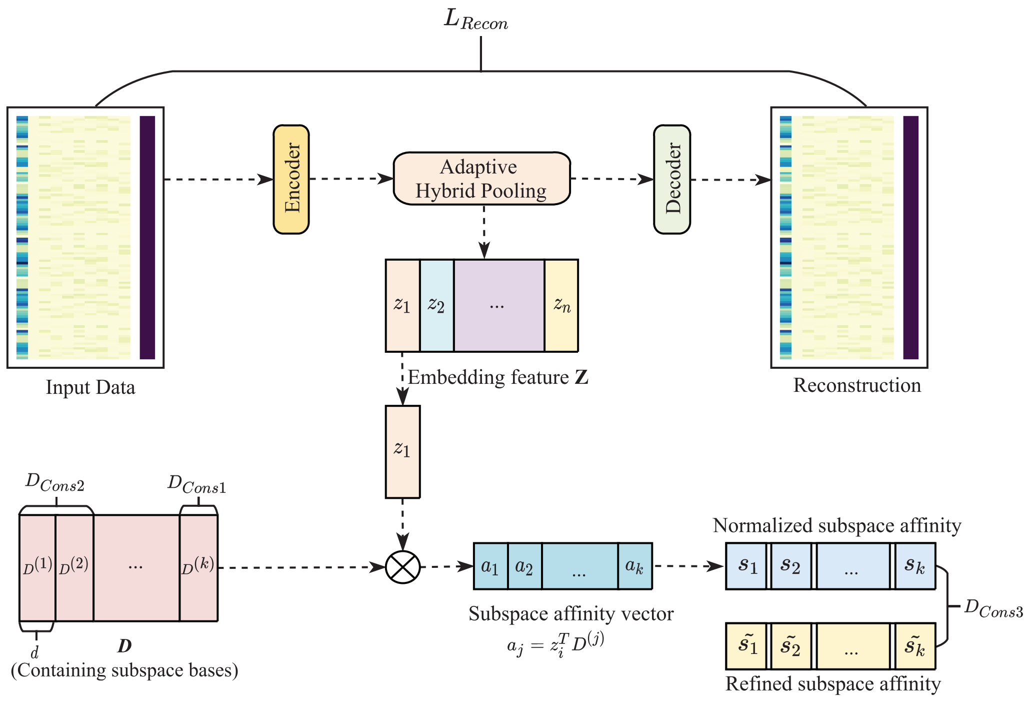

Figure 8: Workflow of EDESC

Let D be the subspace, n the sample count, k the cluster number, z the shallow features from the autoencoder's output, X the input data, and $\hat{X}$ the reconstructed output. The EDESC model aims to minimize the root mean square error between the reconstructed and original data, which is modelled as follows:

$$\begin{aligned} \min \quad & \frac{1}{2n}\|X-\hat{X}\|_F^2 & \\ \text{s.t.} \quad & \|D_u^{(j)}\| = 1, & u=1,\dots,d,\ j=1,\dots,k, \\ & \|D^{(j)\top}D^{(l)}\|_F \le \tau, & j \ne l, \\ & \|z_i^\top D^{(\alpha_i)}\| \gg \max_{j\ne a_i}\|z_i^\top D^{(j)}\|, & i=1,\dots,n. \end{aligned}$$

where τ is a small constant and where $\alpha_i = \arg\max_j \left\|z_i^\top D^{(j)}\right\|$ is the block label. The first constraint limits the norm of the u-th column in the j-th subspace block to 1, preventing gradual decay towards zero during iterations. The irrelevance constraint $\|D^{(j)\top}D^{(l)}\|_F \le \tau$ helps ensure independence among subspaces to extract the most valuable subspaces. The final constraint guarantees that shallow features strongly correlate with only one specific block in subspace D. For computational simplicity, the authors reformulate this optimization problem into the reconstruction loss function L_{Recon} and constraints D_{Cons1}, D_{Cons2}, and D_{Cons3}, as shown below [13].

$$L_{\text{Recon}} = \frac{1}{2n}\sum_{i=1}^{n}\|x_i-\hat{x}_i\|_F^2, \tag{13}$$

$$D_{\text{Cons 1}} = \frac{1}{2}\left\|D^TD\odot I - I\right\|_F^2, \tag{14}$$

$$D_{\text{Cons 2}} = \frac{1}{2}\left\|D^TD\odot O\right\|_F^2, \tag{15}$$

$$D_{\text{Cons }3} = \sum_i \sum_j \tilde{s}_{ij} \log \frac{\tilde{s}_{ij}}{s_{ij}}, \tag{16}$$

where $\odot$ denotes the Hadamard product, O is a hollow matrix, and I is an identity matrix, whereas the subspace affinity s_{ij} and its refined variant $\tilde{s}_{ij}$ are defined in the following two equations.

$$s_{ij} = \frac{\left\| z_i^T D^{(j)} \right\|_F^2 + \eta d}{\sum_j \left(\left\| z_i^T D^{(j)} \right\|_F^2 + \eta d \right)}, \tag{17}$$

$$\tilde{s}_{ij} = \frac{s_{ij}^2 / \sum_i s_{ij}}{\sum_j \left(s_{ij}^2 / \sum_i s_{ij} \right)}, \tag{18}$$

where d is the subspace block size and where η is the parameter controlling smoothness.

Notably, the error in EDESC mainly comes from the autoencoder loss L_{Recon} and the subspace loss D_{Cons}. For any tuning parameter ξ, the latter is defined as:

$$D_{Cons} = \xi \left(D_{Cons1} + D_{Cons2} \right) + \beta D_{Cons3}. \tag{19}$$

4.6 Runtime Complexity

In this section, we analyse the runtime complexity of the EDESC-IDS. Since the ECAE features full symmetry, its encoder and decoder share parameters, each of which contains n neurons. Let I be the number of input samples and let m be the number of neurons in the latent space. As all ECAE training samples require both feedforward J_{AE} and backpropagation ∂J_{AE}, the time complexity for reconstruction error computation is

$$T(J_{AE} + \partial J_{AE}) = \mathcal{O}(2mnI + 2mnI) = \mathcal{O}(mnI). \tag{20}$$

During both the feedforward phase and backpropagation phase for the reconstruction error term, the time complexity is $\mathcal{O}(2mnI)$. The combined time complexity of these two phases yields the total time complexity $\mathcal{O}(mnI)$ for the reconstruction error term. Analogously, the time complexity of the weight decay term is calculated as follows:

$$T(J_{wd} + \partial J_{wd}) = \mathcal{O}(0 + mnI) = \mathcal{O}(mnI). \tag{21}$$

The weight decay constraint adjusts the model's weights and biases only during the feedback phase ∂J_{wd} and incurs no computational cost in the feedforward phase J_{wd}. Thus, its time complexity is identical to that of the sparse regularization term, both of which are $\mathcal{O}(mnI)$. Therefore, the overall complexity of the ECAE is

$$T(J_{ECAE} + \partial J_{ECAE}) = T((J_{AE} + J_{wd}) + (\partial J_{AE} + \partial J_{wd})) = \mathcal{O}(mnI). \tag{22}$$

Notably, ECAE contains merely 4800 parameters, enabling storage as 32-bit floats in approximately $4800 \times 4 \approx 19$ KB, and this storage requirement comfortably fits within low-end ECU random access memory constraints. This efficiency extends to its hybrid pooling module's sparse m-dimensional weight vector containing only 6 trainable parameters, while weight-sharing in convolutional kernels maintains neuron count significantly below the sample size I. These combined optimizations reduce the dominant time complexity to $\mathcal{O}(I)$.

In the clustering layer, the time complexity comprises two components: offline training $T_{offline}$ and online detection T_{online}. Let $\tilde{m}$ denote the size of the widest hidden layer in each neural network, t denotes the time window size, k denotes the cluster count, and d denotes the subspace dimension. Then,

$$T(J_{pool} + \partial J_{pool}) + T_{offline} = \mathcal{O}(2mtI + 2m) + \mathcal{O}(kdmI + \tilde{m}mI), \tag{23}$$

$$T(J_{pool}) + T_{online} = \mathcal{O}(2mtI) + \mathcal{O}(kdmI) = \mathcal{O}(mtI) + \mathcal{O}(kdmI), \tag{24}$$

where J_{pool} and ∂J_{pool} correspond to the pooling operation and its activation gradient in EDESC, respectively. Since $I \gg \max(m, \tilde{m}, t, k, d)$, the I-linear terms dominate, reducing both complexities to $\mathcal{O}(I)$. Consequently, the EDESC-IDS ultimately achieves a final time complexity of just $\mathcal{O}(I)$.

5 Experiments

To evaluate the performance of our in-vehicle IDS, this section presents a comprehensive experimental evaluation.

5.1 Experimental Setup and Evaluation Metrics

For efficient implementation, the core ECAE and clustering modules of the EDESC-IDS are developed via the PyTorch framework for deep learning architecture and the Scikit-learn library for clustering operations.

The evaluation metrics for intrusion detection systems rely on four core statistical measures: true positive (TP), false positive (FP), false negative (FN), and true negative (TN). These measures form the foundation for five key performance metrics: precision, recall, F1-score, false negative rate, and error rate. Among these, precision (P) quantifies the proportion of actual attack frames within the detected attack frames, directly reflecting the false alarm probability in the IDS.

$$P = \frac{TP}{TP + FP}. \tag{25}$$

Recall (R) measures the proportion of all attack frames that are detected. As shown in the formula below, this metric calculates how many attacks are properly detected, which indirectly reflects the system's rate of missing threats.

$$R = \frac{TP}{TP + FN}. \tag{26}$$

On the basis of the above metrics, the F1-score (F1) is introduced as a comprehensive evaluation measure for the IDS, with its defining formula shown below.

$$F1 = 2 \times \frac{P \times R}{P + R}. \tag{27}$$

Additionally, the false negative rate (FNR) and error rate (ER) are used to quantify the proportions of undetected attacks belonging to the attack two-dimensional data frames and false detections, respectively, as shown below.

$$FNR = \frac{FN}{TP + FN + FP}. \tag{28}$$

$$ER = \frac{FP + FN}{TP + TN + FP + FN}. \tag{29}$$

When handling extremely large datasets, computational resources and time costs often become bottlenecks in evaluating model performance. Sequential sampling mitigates this by generating smaller yet statistically representative datasets, substantially reducing computational overhead while retaining predictive validity. As shown in Table 3, we extract sequential samples for model analysis and partition the dataset into 80% training and 20% test sets. Our sequential sampling and targeted attack ratio adjustment strategy not only circumvents the temporal data leakage risks of traditional cross-validation and the computational overhead of time-series-specific validation methods, but also preserves the complete temporal features of attacks with strong sequential dependencies such as DoS, Gear and RPM. Given the sequentially sampled small-scale dataset, time-series cross-validation is implemented on the training set to maintain temporal ordering and evaluate model performance across varied time segments, thereby enhancing generalizability and stability under data-limited conditions.

Table 3: Distribution analysis of CAN bus attack patterns via sequential sampling

Attack type	Total messages	Normal messages	Injected messages
DoS attack	201,595	152,053	49,542
Fuzzy attack	203,881	174,330	29,551
Spoofing gear attack	202,805	165,320	37,485
Spoofing RPM attack	201,837	164,611	37,226

5.2 Hyperparameters

It is well known that hyperparameters are critical factors in determining model performance. To improve computational efficiency and reduce tracking time, the EDESC-IDS model uses Bayesian-based automatic hyperparameter optimization (BAHO) to find optimal configurations. The relevant hyperparameter settings are reported in Table 4.

Table 4: Hyperparameter configuration

Parameter	Value
Loss function	MSE
Optimizer	AdamW
Batch size	256
Training epochs	50
Convolution kernel size	3 × 3
Dilation rate	1, 2
Learning rate	1e-3

Based on the above hyperparameters, Fig. 9 shows the loss function convergence curves during EDESC-IDS model training under four attacks. Specifically, fuzzy attacks result in the highest final loss value and slower convergence, indicating their training complexity. Similarly, spoofing RPM attacks exhibit slow convergence with high final loss values, whereas spoofing Gear attacks converge faster. In contrast, high-priority DoS injection attacks result in lower initial loss, faster convergence, and maintain the lowest loss values throughout training.

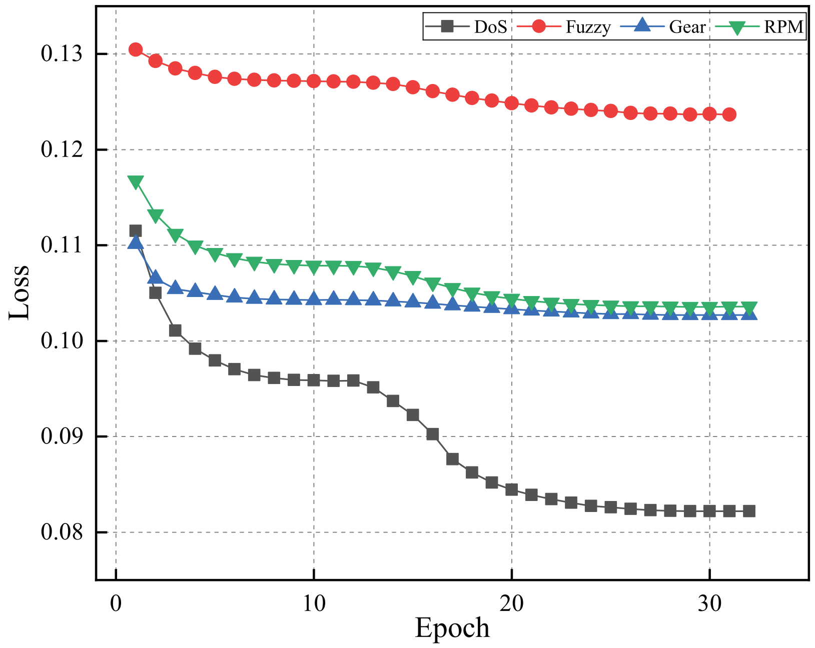

Figure 9: Training loss on each dataset from HCRL

5.3 Detection Performance

To measure the performance of our IDS, we compare it in detail with five baseline methods: the LSTM-AE-IDS [17], OC-SVM [23], DAE [25], SOMK-C [26], and DESC-IDS [27]. The ER, FNR, P, R and F1 values of different baseline methods against the EDESC-IDS are listed in Table 5, where the best performance values are highlighted in bold.

The proposed EDESC-IDS model learns low-dimensional spatiotemporal features through ECAE while integrating subspace clustering to optimize its learning path. This ensures effective utilization of spatiotemporal features within the clustering module, achieving stable clustering performance. Table 5 shows F1-scores of 99.64%, 98.74%, 99.68%, and 97.79% on four attack datasets. The elevated error rate of SOMK-C against fuzzy attacks reveals its low generalizability, thereby constraining its applicability in various attack scenarios. Similarly, the LSTM-AE-IDS results in high volatility in both the ER and FNR under fuzzy attacks. Furthermore, the semisupervised OC-SVM yields notably lower precision scores than the other baselines across all the attack datasets, coupled with substantially higher error rates on Dos, Gear and RPM attacks. These limitations confirm the unsuitability of OC-SVM for CAN bus anomaly detection. In contrast, unsupervised clustering-based models DESC and DAE achieve superior performance on multiple metrics. However, their traditional loss function designs focus solely on data reconstruction and distribution modelling while neglecting the intrinsic relationship between clustering and feature learning, resulting in ambiguous convergence patterns during dimensionality reduction and inflexible adaptation to diverse attack scenarios.

To further validate the generalization capability of EDESC-IDS, similar experiments are conducted on the ORNL dataset [37], as shown in Table 6. The experimental results demonstrate consistent strong performance in this regard.

Table 5: EDESC-IDS performance comparison against baseline methods, with best values highlighted in bold

Attack type	Method	ER (%)	FNR (%)	P (%)	R (%)	F1 (%)
DoS	EDESC-IDS	**0.32**	0.53	**99.81**	99.47	**99.64**
	DESC-IDS	3.74	4.38	96.69	95.62	96.08
	OC-SVM	30.95	0.69	55.72	99.31	71.39
	DAE	3.76	0.12	91.27	99.88	95.36
	LSTM-AE-IDS	5.75	6.73	99.58	93.27	96.32
	SOMK-C	2.22	**0.00**	90.88	**100.00**	95.22
Fuzzy	EDESC-IDS	**1.26**	**1.26**	**98.77**	**98.74**	**98.74**
	DESC-IDS	4.63	4.67	95.25	95.53	95.27
	OC-SVM	32.22	94.50	6.94	5.50	6.14
	DAE	6.13	0.12	91.27	99.88	95.36
	LSTM-AE-IDS	17.06	0.00	82.94	100.00	90.68
	SOMK-C	26.80	99.80	0.46	0.20	0.28
Gear	EDESC-IDS	**0.32**	0.32	**99.69**	99.69	**99.68**
	DESC-IDS	2.11	2.16	97.88	97.84	97.86
	OC-SVM	73.80	28.85	35.43	71.15	47.30
	DAE	11.98	18.20	94.63	81.80	87.75
	LSTM-AE-IDS	24.77	28.45	87.25	75.23	80.80
	SOMK-C	4.40	**0.00**	79.70	**100.00**	88.70
RPM	EDESC-IDS	**2.22**	2.22	**97.89**	97.78	**97.79**
	DESC-IDS	3.76	4.00	96.41	96.00	96.17
	OC-SVM	60.54	34.85	33.43	65.15	44.18
	DAE	8.40	4.27	95.73	95.73	92.10
	LSTM-AE-IDS	27.20	27.55	90.75	72.80	80.79
	SOMK-C	4.51	**0.00**	79.04	**100.00**	88.29

Table 6: EDESC-IDS performance evaluation on the ORNL dataset

Attack type	ER (%)	FNR (%)	P (%)	R (%)	F1 (%)
reverse light on attack	4.00	4.00	96.34	96.00	96.02
reverse light on attack masquerad	1.10	1.30	98.70	98.7	98.70
reverse light off attack	2.20	2.00	98.10	98.00	98.10

5.4 Time Cost

Given the high latency requirements of CAN systems, it has become a crucial metric for real-time detection. Specifically, the detection latency t_l is defined below.

$$t_l = t_d - t_a, \tag{30}$$

where t_a is the attack occurrence time and where t_d is the attack detection time. The former falls between the start t_{cs} and end t_{ce} of the batch data aggregation duration, i.e., $t_{cs} \le t_a \le t_{ce}$, whereas the latter equals the sum of t_{ce}, the inference time t_i of the EDESC-IDS model, and the data preprocessing delay t_{dp}. On this basis, the calculation for t_l can be refined as follows:

$$t_i + t_{dp} \le t_l = t_{ce} + t_i + t_{dp} - t_a \le t_{bf} + t_i + t_{dp}, \tag{31}$$

where the time delay of the frame builder is $t_{bf} = t_{ce} - t_{ce}$. According to the above formula, the maximum detection delay occurs at $t_{cs} = t_a$, whereas the minimum detection delay appears at $t_{ce} = t_a$.

Our EDESC-IDS adopts a server-side offline training architecture, where the training time is excluded from the in-vehicle latency metrics. However, deep non-clustering models such as AttGRU [31] and ConvGRU [34] are widely applied in CAN anomaly detection for vehicles. To further demonstrate the efficiency of EDESC-IDS, comparative analyses of training and inference time among these three models are conducted, as shown in Fig. 10. The results indicate that EDESC-IDS not only achieves superior training efficiency on the server side compared to deep non-clustering models, but also reduces in-vehicle detection latency, demonstrating the advantages of its lightweight architectural design.

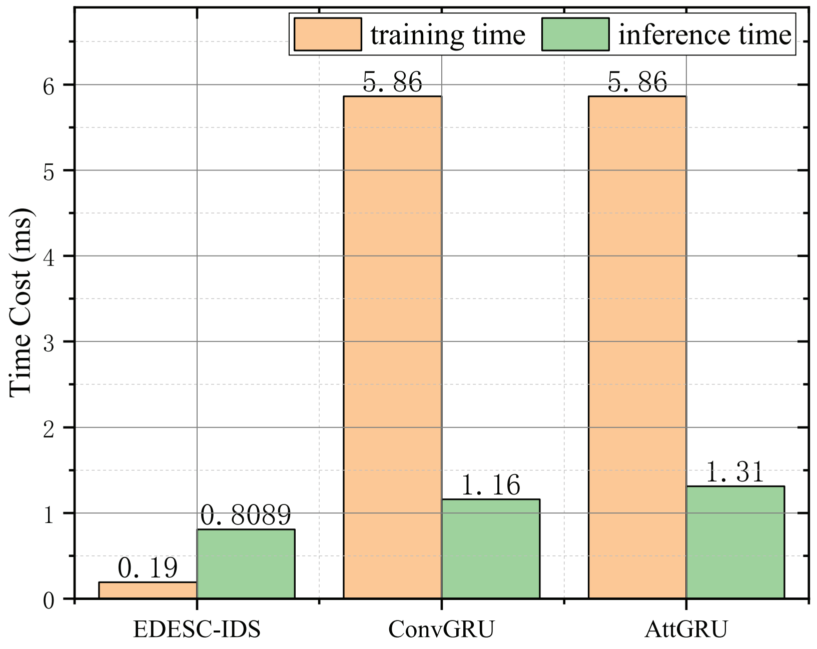

Figure 10: Comparative results of per-batch training and inference time costs across models

5.5 Ablation Study

This paper introduces an in-vehicle intrusion detection system that is based on deep embedded subspace clustering. Its core components include the ECAE and the EDESC model. The aforementioned experiments demonstrated that the system achieves strong detection performance in CAN bus anomaly detection. To evaluate the impact of its core components on anomaly detection, we conduct ablation experiments using the F1-score as the evaluation metric to analyse the completeness of the EDESC-IDS system. The experimental setups are as follows:

- **Scheme A:** Replace the extended convolutional layers and dynamic hybrid pooling layers in ECAE with standard convolutional layers and linear layers, respectively.

- **Scheme B:** Replace EDESC with the DESC flow clustering model proposed by Cheng [27].
- **Scheme C:** Replace EDESC with principal component analysis and k-means clustering model.

As shown in Fig. 11, when the ECAE autoencoder or the EDESC model is removed, the performance of the EDESC-IDS significantly decreases, indicating that both modules are critical components of the system. The F1-score in the ablation experiments shows that without the ECAE autoencoder's effective processing of shallow features, the system's performance decreases sharply, which demonstrates the module's foundational role in detecting abnormal data. Similarly, the EDESC model is also key to the superior performance of the EDESC-IDS, as its embedding clustering module's constraints optimize the autoencoder's convergence direction, enabling stronger generalization capabilities against various attack data. In summary, both contribute essentially to the overall performance of the EDESC-IDS.

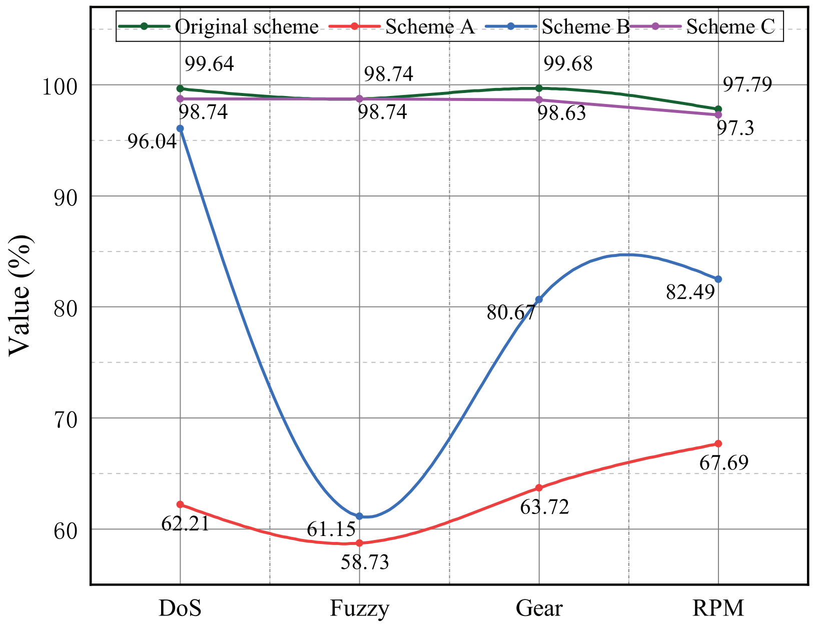

Figure 11: Comparative F1-score evaluation of different ablation study schemes

6 Conclusion

This paper develops an unsupervised intrusion detection system via deep embedding subspace clustering. The system effectively detects different injection patterns of both known and unknown attacks, and contains four modules: data preprocessing, feature engineering, feature extraction, and deep embedding subspace clustering. For the data preprocessing and feature engineering modules, the system fully considers the spatiotemporal characteristics of CAN data and uses a stream builder to create feature domains. In the feature extraction phase, the EDESC-IDS employs a lightweight autoencoder model, ECAE, with dilated convolutions to capture spatiotemporal information and extract valuable shallow features.

To address computational resource limitations from massive datasets, the EDESC-IDS creates a subset through sequential sampling from the benchmark automotive hacking dataset. Experiments show that this system has strong robustness and effectiveness in identifying various attack scenarios (DoS, Fuzzy, Gear, and RPM). Compared with traditional unsupervised methods, it achieves significant improvements across multiple detection metrics. Additionally, the real-time detection ability of the EDESC-IDS is validated. These collective results thoroughly confirm the system's effectiveness in anomaly detection.

Our approach has two critical limitations in addressing vehicular CAN network challenges. First, it lacks a framework enabling privacy-preserving collaboration across vehicles with data silos, impeding model training without exchanging raw data. Second, the subspace clustering module's decision-making process lacks transparency, as its grouping logic fails to align with domain-specific automotive knowledge such as ECU operational principles or attack patterns. To address these limitations, future work will employ a federated learning framework [38], enabling vehicles to conduct local model training and share encrypted parameter updates. This enhances cross-vehicle model generalization while preserving data privacy. Moreover, we will integrate argumentation technology to formalize clustering rules using extension [39] or labeling [40] semantics.

Acknowledgement: Not applicable.

Funding Statement: The work was supported by the National Natural Science Foundation of China (Grant No. 62172292).

Author Contributions: Lixing Tan: Writing—original draft, Methodology, Investigation; Liusiyu Chen: Data curation, Software, Investigation; Yang Wang: Writing—review & editing; Zhenyu Song: Resources, Formal analysis, Supervision; Zenan Lu: Resources, Review & editing, Supervision. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Zhenyu Song, upon reasonable request.

Ethics Approval: This study did not involve any human or animal subjects, and therefore, ethical approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Agrawal K, Alladi T, Agrawal A, Chamola V, Benslimane A. NovelADS: a novel anomaly detection system for intra-vehicular networks. IEEE Trans Intell Transp Syst. 2022;23(11):22596–606. doi:10.1109/tits.2022.3146024.
2. Xie Y, Wu Q, Fan P, Cheng N, Chen W, Wang J, et al. Resource allocation for twin maintenance and task processing in vehicular edge computing network. IEEE Internet Things J. 2025;12(15):32008–21. doi:10.1109/jiot.2025.3576582.
3. Han M, Cheng P, Ma S. PPM-InVIDS: privacy protection model for in-vehicle intrusion detection system based complex-valued neural network. Veh Commun. 2021;31(1):100374. doi:10.1016/j.vehcom.2021.100374.
4. Olufowobi H, Young C, Zambreno J, Bloom G. SAIDuCANT: specification-based automotive intrusion detection using controller area network (CAN) timing. IEEE Trans Veh Technol. 2019;69(2):1484–94. doi:10.1109/tvt.2019.2961344.
5. Aliwa E, Rana O, Perera C, Burnap P. Cyberattacks and countermeasures for in-vehicle networks. ACM Comput Surv. 2021;54(1):1–37. doi:10.1145/3431233.
6. Liang J, Lin Q, Chen J, Zhu Y. A filter model based on hidden generalized mixture transition distribution model for intrusion detection system in vehicle ad hoc networks. IEEE Trans Intell Transp Syst. 2019;21(7):2707–22. doi:10.1109/tits.2019.2905415.
7. Khan MH, Javed AR, Iqbal Z, Asim M, Awad AI. DivaCAN: detecting in-vehicle intrusion attacks on a controller area network using ensemble learning. Comput Secur. 2024;139(8):103712. doi:10.1016/j.cose.2024.103712.
8. Studnia I, Alata E, Nicomette V, Kaâniche M, Laarouchi Y. A language-based intrusion detection approach for automotive embedded networks. Int J Embed Syst. 2018;10(1):1–12. doi:10.1504/ijes.2018.10010488.
9. Suda H, Natsui M, Hanyu T. Systematic intrusion detection technique for an in-vehicle network based on time-series feature extraction. In: 2018 IEEE 48th International Symposium On Multiple-valued Logic (ISMVL). Piscataway, NJ, USA: IEEE; 2018. p. 56–61.

10. Gmiden M, Gmiden MH, Trabelsi H. An intrusion detection method for securing in-vehicle CAN bus. In: 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA). Piscataway, NJ, USA: IEEE; 2016. p. 176–80.
11. Yang L, Moubayed A, Shami A. MTH-IDS: a multitiered hybrid intrusion detection system for internet of vehicles. IEEE Internet Things J. 2021;9(1):616–32. doi:10.1109/jiot.2021.3084796.
12. Sun H, Chen M, Weng J, Liu Z, Geng G. Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism. IEEE Trans Veh Technol. 2021;70(10):10880–93. doi:10.1109/tvt.2021.3106940.
13. Cai J, Fan J, Guo W, Wang S, Zhang Y, Zhang Z. Efficient deep embedded subspace clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway, NJ, USA: IEEE; 2022. p. 1–10.
14. Seo E, Song HM, Kim HK. GIDS: GAN based intrusion detection system for in-vehicle network. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST). Piscataway, NJ, USA: IEEE; 2018. p. 1–6.
15. Gu X, Wu Q, Fan P, Fan Q, Cheng N, Chen W, et al. DRL-based resource allocation for motion blur resistant federated self-supervised learning in IoV. IEEE Internet Things J. 2024;12(6):7067–85. doi:10.1109/jiot.2024.3492326.
16. Tariq S, Lee S, Woo SS. 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. New York, NY, USA: ACM; 2020. p. 1048–55.
17. Ashraf J, Bakhshi AD, Moustafa N, Khurshid H, Javed A, Beheshti A. Novel deep learning-enabled LSTM autoencoder architecture for discovering anomalous events from intelligent transportation systems. IEEE Trans Intell Transp Syst. 2020;22(7):4507–18. doi:10.1109/tits.2020.3017882.
18. Qin H, Yan M, Ji H. Application of controller area network (CAN) bus anomaly detection based on time series prediction. Veh Commun. 2021;27:100291. doi:10.1016/j.vehcom.2020.100291.
19. Alqahtani H, Kumar G. A deep learning-based intrusion detection system for in-vehicle networks. Comput Electr Eng. 2022;104(1):108447. doi:10.1016/j.compeleceng.2022.108447.
20. He Z, Chen Y, Zhang H, Zhang D. WKN-OC: a new deep learning method for anomaly detection in intelligent vehicles. IEEE Trans Intell Veh. 2023;8(3):2162–72. doi:10.1109/tiv.2023.3243356.
21. Wang S, Wang Y, Zheng B, Cheng J, Su Y, Dai Y. Intrusion detection system for vehicular networks based on MobileNetV3. IEEE Access. 2024;12:106285–302. doi:10.1109/access.2024.3437416.
22. Cheng P, Hua L, Jiang H, Liu G. LSF-IDM: deep learning-based lightweight semantic fusion intrusion detection model for automotive. Peer Peer Netw Appl. 2024;17(5):2884–905. doi:10.1007/s12083-024-01679-x.
23. Taylor A, Japkowicz N, Leblanc S. Frequency-based anomaly detection for the automotive CAN bus. In: 2015 World Congress on Industrial Control Systems Security (WCICSS). Piscataway, NJ, USA: IEEE; 2015. p. 45–9 doi:10.1109/wcicss.2015.7420322.
24. 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. 2021;24(2):2122–34. doi:10.1109/tits.2021.3128634.
25. Amarbayasgalan T, Jargalsaikhan B, Ryu KH. Unsupervised novelty detection using deep autoencoders with density based clustering. Appl Sci. 2018;8(9):1468. doi:10.3390/app8091468.
26. Barletta VS, Caivano D, Nannavecchia A, Scalera M. Intrusion detection for in-vehicle communication networks: an unsupervised Kohonen SOM approach. Future Internet. 2020;12(7):119. doi:10.3390/fi12070119.
27. Cheng P, Han M, Liu G. DESC-IDS: towards an efficient real-time automotive intrusion detection system based on deep evolving stream clustering. Future Gener Comput Syst. 2023;140(1):266–81. doi:10.1016/j.future.2022.10.020.
28. Jeong S, Kim HK, Han ML, Kwak BI. Aero: automotive ethernet real-time observer for anomaly detection in in-vehicle networks. IEEE Trans Ind Informatics. 2023;20(3):4651–62. doi:10.1109/tii.2023.3324949.
29. Gao K, Huang H, Liu L, Du R, Zhang J. A multi-attention based CNN-BiLSTM intrusion detection model for in-vehicle networks. In: 2023 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). Piscataway, NJ, USA: IEEE; 2023. p. 809–16. doi:10.1109/ispa-bdcloud-socialcom-sustaincom59178.2023.00138.

30. Leen G, Heffernan D. Expanding automotive electronic systems. Computer. 2002;35(1):88–93. doi:10.1109/2.976923.
31. Javed AR, Ur Rehman S, Khan MU, Alazab M, Reddy T. CANintelliIDS: edetecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU. IEEE Trans Netw Sci Eng. 2021;8(2):1456–66. doi:10.1109/tnse.2021.3059881.
32. Farsi M, Ratcliff K, Barbosa M. An overview of controller area network. Comput Control Eng. 1999;10(3):113–20. doi:10.1049/cce:19990304.
33. Song HM, Kim HK. Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data. IEEE Trans Veh Technol. 2021;70(2):1098–108. doi:10.1109/tvt.2021.3051026.
34. Wang S, Cheng J, Wang Y, Li S, Kang L, Dai YF. ConvGRU: a lightweight intrusion detection system for vehicle networks based on shallow CNN and GRU. IEEE Access. 2025;13(4):73297–318. doi:10.1109/access.2025.3563908.
35. Huan S, Zhang X, Shang W, Cao H, Li H, Yang Y, et al. T-shaped CAN feature integration with a lightweight deep learning model for in-vehicle network intrusion detection. IEEE Trans Intell Transp Syst. 2024;25(12):21183–96. doi:10.1109/tits.2024.3478371.
36. Lo W, Alqahtani H, Thakur K, Almadhor A, Chander S, Kumar G. A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. Veh Commun. 2022;35(2–3):100471. doi:10.1016/j.vehcom.2022.100471.
37. Verma ME, Bridges RA, Iannacone MD, Hollifield SC, Moriano P, Hespeler SC, et al. A comprehensive guide to CAN IDS data and introduction of the ROAD dataset. PLoS One. 2024;19(1):e0296879. doi:10.1371/journal.pone.0296879.
38. Gu X, Wu Q, Fan P, Cheng N, Chen W, Letaief KB. DRL-based federated self-supervised learning for task offloading and resource allocation in ISAC-enabled vehicle edge computing. Digit Communicat Netw. 2024;11(5):1614–27. doi:10.1016/j.dcan.2025.11.005.
39. Tan L, Zhu Z, Zhang J. A general approach to extension-based semantics in abstract argumentation. Artif Intell. 2023;315(2):103836. doi:10.1016/j.artint.2022.103836.
40. Tan L, Zhu Z, Wang F, Zhang J. Graded labellings for abstract argumentation. Int J Approx Reason. 2023;152(2):59–93. doi:10.1016/j.ijar.2022.10.009.