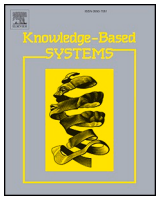
Knowledge-Based Systems 304 (2024) 112563

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



# Semi-supervised intrusion detection system for in-vehicle networks based on variational autoencoder and adversarial reinforcement learning

Trieu-Phong Nguyen [a], Jeongho Cho [b,1], Daehee Kim [a,*,1]

[a] Department of Mobility Convergence Security, Soonchunhyang University, Asan-si 31538, Chungcheongnam-do, South Korea
[b] Department of Electrical Engineering, Soonchunhyang University, Asan-si 31538, Chungcheongnam-do, South Korea

ARTICLE INFO

*Keywords:*
Intrusion detection system (IDS)
Controller area network (CAN)
Automotive security
Semi-supervised learning
Reinforcement learning

ABSTRACT

Despite the affordability, simplicity, and efficiency of controller area network (CAN) protocols, the security vulnerability remains a major challenge. Currently, a machine learning-based intrusion detection system (IDS) is considered an effective approach for improving security in CAN by identifying malicious attacks. However, earlier studies that relied on supervised learning methods required considerable amounts of labeled data. Data collection from vehicles is time-consuming and expensive. Furthermore, the obtained data exhibited a class imbalance, which presents further challenges in the analysis and model training. Thus, we propose a semi-supervised learning-based IDS that combines variational autoencoder (VAE) and adversarial reinforcement learning for the multi-class classification of both known and unknown attacks. The proposed system capitalizes on the diverse patterns inherent in unlabeled data, transforming this data space into one that is more conducive to classification. Concurrently, adversarial agents in the reinforcement learning algorithm interact competitively, progressively enhancing their ability to intelligently classify and select samples. To reduce the reliance on labeled data and effectively exploit them, we utilize a pseudo-labeling process for pre-training. Experimental results indicate that the proposed model achieves more effective classification while requiring less labeled data compared to other baseline models for known attacks. By inheriting the advantages of VAE, promising results demonstrate that the proposed system detects unknown attacks containing similar or completely different characteristics with high F1 scores exceeding 0.9 and 0.84, respectively. Finally, the proposed system was demonstrated to be a lightweight model for the expeditious detection of malevolent messages introduced into in-vehicle networks to ensure minimal latency.

## 1. Introduction

In the contemporary information landscape, vehicle-to-everything (V2X) communication and the internet of vehicles (IoV) have emerged as pivotal elements in the advancement of intelligent transportation systems [1]. The advent of the IoV has facilitated vehicles' access to real-time traffic data, consequently diminishing traffic mishaps and congestion, enhancing transportation efficiency, and conserving valuable time for individuals [2]. The integration of utilities requires the continuous expansion of in-vehicle networks (IVNs) to ensure smooth vehicle operation [3]. However, the increase in the number of network connections provides attackers with more opportunities for exploitation [4].

The security of IVN remains a paramount concern for the advancement of the IoV. Continually evolving threats in cyberspace increase the risks to vehicular security and passenger safety. An autonomous vehicle can execute driving tasks autonomously, encompassing actions such as self-driving, lane adherence, automated parking, and collision avoidance facilitated by an IVN interlinking various electronic control units (ECUs). The role of an ECU is to interpret data from sensors and transmit pertinent commands to actuators through a controller area network (CAN), which in turn implements the associated autonomous operations [5]. Although a CAN offers an efficient, reliable, and cost-effective communication conduit among ECUs, its deficient security features render it vulnerable to cyberattacks [6]. If an attacker successfully executes a malicious attack that results in behavior not controlled by an automated system, the directives imposed can lead to catastrophic consequences [7]. An intrusion detection system (IDS) is

* Corresponding author.
*E-mail address:* daeheekim@sch.ac.kr (D. Kim).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2024.112563
Received 18 February 2024; Received in revised form 9 September 2024; Accepted 27 September 2024
Available online 28 September 2024

designed to identify potential security threats and malevolent payloads by analyzing various facets of network data transmission, including network traffic, addresses, and data patterns, thereby facilitating proactive security countermeasures [8]. Among the various approaches, machine learning-based systems are effective in providing timely alerts for dangerous attacks. Approaches based on supervised learning concentrate primarily on the extraction of patterns utilizing convolutional neural network (CNN) algorithms [9] or on leveraging the periodicity of the identifier field in the CAN using long short-term memory (LSTM) networks [10]. Notable studies encompass the integration of CNN and LSTM networks [11], effectively merging spatial and temporal features to extract representations from CAN data. However, supervised learning-based systems encounter challenges in data acquisition and issues pertaining to data imbalance [12]. Furthermore, models that employ supervised learning requires abundant annotated data that expends considerable effort. Typically, malicious intrusions constitute only a minor proportion of the cumulative network traffic. Even when an IDS attains a high accuracy rate on imbalanced datasets, a significant portion of the intrusion data from minority classes is often undetected. This results in inadequate protection against potential attacks. To address the challenge of data imbalance, a resampling method [13] is frequently employed to alter the distribution of samples between the minority and majority classes in the original dataset. This adjustment aims to bolster the accuracy of minority class detection and consequently enhance the overall model performance. However, the resampling methods are susceptible to noise interference. For example, if minority classes contain a substantial number of noise points, oversampling can magnify the impact of noise, leading to a decline in the classification efficacy. The challenge of data labeling and imbalance is a difficult problem that requires further research.

To solve these data issues, our proposed model integrates unsupervised learning techniques to enhance feature representation and applies optimized strategies from an adversarial environment reinforcement learning (AERL) model to improve the sample distribution [14]. The main contributions of this study are summarized as follows:

1) We propose a novel semi-supervised learning-based IDS that combines variational autoencoder (VAE) and AERL for an IVN that performs multi-class classifications for a variety of attacks. We also utilize the pseudo-label method which helps the proposed model to learn new feature representations. The effectiveness of the proposed model shows that these learned features are strongly correlated with the true labels, even in the absence of these true labels during pretraining. The proposed system demonstrated an efficiency exceeding 98.5 % accuracy in detecting intrusive CAN messages. These results are significant and surpass the performance of the baseline models when only 50 % of the labeled data were used.
2) We solve the severe data imbalance problems of IVN by employing AERL, in which the environment agent attempts to select difficult-to-learn samples for the classifier agent in an adversarial manner. This approach is presented in detail in Section 5.3, where data from minority classes are preferentially sampled by the proposed system by optimizing the training strategy.
3) The proposed IDS can further detect unknown attacks by extracting critical features using a VAE without using unknown attack samples during training. These promising results indicate that the proposed model can be effectively adapted for detecting unknown malicious messages, achieving an accuracy of greater than 90 % in most scenarios.
4) We analyze the effectiveness of the proposed system using two public datasets: a car-hacking dataset (CHD) [35] and a real ORNL automotive dynamometer CAN intrusion dataset (ROAD) [36]. Although the second dataset is complex, the proposed model still achieves the best F1 score. Comprehensive assessments revealed that our system not only attained optimal performance but also markedly decreased error rates, particularly in terms of false alarms. When compared with the baseline models, the false-alarm rates in our proposed system showed a substantial reduction, lower than 0.1 % for CHD and lower than 5 % for ROAD.

## 2. Background

### 2.1. CAN bus system

CAN is a message-based vehicular bus standard established to facilitate communication between automotive ECUs [6]. In the CAN protocol, message frames are categorized as standard and extended frames. The primary distinction between these frame types lies in the arbitration field. In response to the increasing number of ECU nodes yielding a diversified array of communication messages, an extended frame has been engineered to encompass a more expansive arbitration field necessitating an increased number of CAN ID bits. Specifically, CAN 2.0A devices employ a standard frame that encompasses 11 bits of the identifier. By contrast, CAN 2.0B devices can use both frame formats, with the extended frame boasting a larger 29-bit frame [15]. The arbitration field not only serves as an identifier but also establishes a priority for message transmission. The lower order number of the identifier, the higher its priority and transmit permission. [16]. The structure of the CAN data frame is illustrated in Fig. 1. The other fields of a CAN frame include the start of frame (SOF, 1 bit); remote transmission request (RTR, 1 bit); identifier extension (IDE, 1 bit); reserved bit (R0, 1 bit); data length code (DLC, up to 4 bits); data payload (up to 8 bytes); cyclic redundancy code (CRC, 16 bits); acknowledge (ACK, 2 bits); end of frame (EOF, 1 bit), and inter-frame spacing (IFS, 7 bits) [17].

The CAN protocol incorporates multiple mechanisms for error verification, one of which is the CRC. The transmitting node computes the CRC and incorporates it into the transmitted CAN frame. Upon receiving the frame, the destination node recalculates the CRC. If a mismatch exists between the received and recalculated CRCs, an error flag is triggered. Subsequently, every node is notified instantaneously. In the most adverse scenario, the error recovery process transmits up to 31 bits. The duration of error signaling and correction typically spans 17–31-bit periods and is exclusive to the message's subsequent retransmission. Through this mechanism, the CAN protocol aims to ensure the accuracy of the transmitted data [18]. However, this safeguard predominantly addresses unintentional discrepancies during transmission. Malicious attackers can easily craft CAN frames accompanied by a valid CRC. Moreover, owing to the absence of authentication in the CAN bus system, an attacker can easily impersonate any node. The recipient node cannot authenticate the legitimacy of the transmitted frame [5].

### 2.2. Threat and attack model

As mentioned previously, the error-checking mechanisms used on a CAN bus can be easily bypassed. The absence of authentication in the system permits the participation of unauthorized nodes in the network communication [19]. Since CAN operates as a broadcast network, it does not utilize source and destination addresses, allowing every node to access all messages. As the transmitted data are unencrypted, it becomes possible for an adversary to listen to and decipher the data. A compromised ECU that exploits this vulnerability can flood a network with malicious CAN frames, thereby hindering the communication among legitimate nodes. Consequently, CAN buses are deficient in terms of availability [20]. Moreover, the increased number of connections in modern vehicles requires large amounts of data and poses security risks [21]. By exploiting these vulnerabilities, an attacker can initiate different types of attacks, as illustrated in Fig. 2, which can be generalized as follows.

**Fabrication attack.** The objective of this attack is to insert new frames into the CAN bus through a compromised ECU. All the legitimate ECUs persist in their operations and dispatch their original data. Intrusive frames attempt to override or overload any periodic legitimate


**Fig. 1.** CAN 2.0 data frame format.

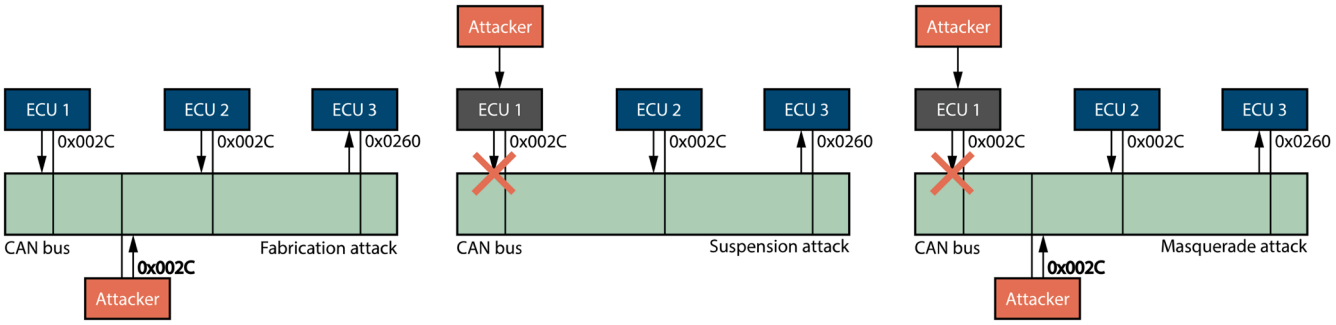



**Fig. 2.** Security risks in CAN bus.

messages transmitted by critical ECUs [22]. Fig. 2 illustrates an example of a fabrication attack. Typical examples of such attacks include DoS, spoofing, and fuzzy attacks. DoS attacks attempt to disrupt communication services by inundating systems with high-priority requests. A fuzzing attack can be launched in a more sophisticated scenario by randomly injecting anomalous messages into the CAN bus. However, spoofing attacks, which are often referred to as targeted ID attacks, concentrate on specific IDs without employing random injections.

**Suspension attack.** Suspension attacks involve the removal of legitimate frames from compromised ECUs [23]. As shown in Fig. 2, the goal of this attack is to halt or suspend message transmission from a weakly compromised ECU, thereby preventing communication with other nodes. This attack not only negatively affects the infiltrated ECU but also other nodes that require data from the attacked node to function properly.

**Masquerade attack.** To execute a masquerade attack, an attacker must compromise two ECUs, one of which is a strongly compromised ECU and the other which may be weaker [19] as depicted in Fig. 2. An adversary observes the pattern and frequency of the messages transmitted by a weakly compromised ECU. Subsequently, the completely controlled ECU dispatches messages that match the ID of the recorded ECU while maintaining the observed frequency [24]. Replay and impersonation attacks are examples of such threats.

## 3. Related works

Recently, machine learning-based techniques have witnessed a significant surge in development, and their potential application in the IVNs has been a topic of discussion [25]. In this section, we investigate key research studies on machine learning-based in-vehicle intrusion detection utilizing CAN datasets collected from real-world environments within the past 5 years, which are summarized in Table 1.

### 3.1. Machine learning-based in-vehicle IDS

Islam et al. introduced a graph-based architecture designed to identify attacks on the CAN bus [26]. They utilized Gaussian naive Bayes for related features to leverage the graph properties. With standard graph features, they achieved a 96.2 % detection accuracy for mixed attacks using the OTIDS dataset [27]. Based on the transmission period

**Table 1**
Literature reviews of machine learning-based IDS for the in-vehicle network.

| Related work | Main method | Feature | Multi-class classification | Attacks | | Amount of labeled data | Real dataset | | Real time evaluation |
|---|---|---|---|---|---|---|---|---|---|
| | | | | known | unknown | | single | multiple | |
| [8] | SMOTE, LOF-OBLR, Metric learning | CAN message | ✓ | ✓ | | Full | | ✓ | |
| [15] | Transformer | CAN ID | ✓ | ✓ | | Full | ✓ | | ✓ |
| [26] | GGNB | CAN message | | ✓ | | Full | ✓ | | ✓ |
| [28] | DCNN | CAN ID | | ✓ | | Full | ✓ | | ✓ |
| [29] | CNN-LSTM, Attention | CAN message | | ✓ | | Full | ✓ | | ✓ |
| [30] | SupCon ResNet | CAN ID | ✓ | ✓ | | Full | | ✓ | ✓ |
| [31] | CANintelliIDS | CAN message | ✓ | ✓ | | Full | ✓ | | |
| [32] | SSDDQN | CAN message | ✓ | ✓ | ✓ | Full | | ✓ | |
| [33] | CAAE | CAN ID | | ✓ | ✓ | Partial | ✓ | | ✓ |
| [34] | Semi-Supervised CNN | CAN message | | ✓ | | Partial | ✓ | | |
| [36] | Tree structure ML | CAN message | ✓ | ✓ | | Full | | ✓ | ✓ |
| Ours | VAE, AERL | CAN message | ✓ | ✓ | ✓ | Partial | | ✓ | ✓ |

property, Song et al. proposed an advanced Inception-ResNet IDS [28]. Their binary classification model was trained using the sequential CAN IDs extracted from the bus. Although they achieved high performance, their study did not take into account the payload of messages. This method may not be effective for detecting new or previously unseen attacks because of fitting in labeled data only. Another relevant approach, Sun et al. integrated a CNN, a LSTM, and an attention mechanism for an anomaly detection system in a CAN bus [29]. In this study, an IDS structure is built on a multi-branch neural network. Successive CAN IDs are arranged into a 2-dimensional time series matrix, which is subsequently fed into a convolutional layer to extract spatial features. Then, the bidirectional LSTM layers are responsible for temporally representing the outcome from the previous layers. The outputs from the LSTM are merged with an attention mechanism before passing through a dense layer. Since the RMSE is used as the anomaly score and a threshold is defined for its determination, this method may encounter challenges in recognizing complex anomalies and face limitations in categorizing various types of attacks. To enhance the performance with insufficient datasets, Hoang et al. [30] applied transfer learning techniques using a supervised contrastive (SupCon) residual network (ResNet) to achieve efficient multiple-attack classification results. Despite yielding promising outcomes, the SupCon ResNet system requires a proficient source model that necessitates training on a sufficiently labeled dataset to facilitate the transfer of knowledge to the target model. Other researchers have improved detection performance by correlating data from multiple ECUs. CANintelliIDS, developed by Javed et al. [31], is designed to process a continuous stream of CAN messages using a neural network. This network comprises two layers of CNN, followed by a gated recurrent unit layer. The proposed method achieved an F-score of 93 %, which was better than that of CNN-based models on the OTIDS dataset. In another approach, Nguyen et al. [15] exploited the advantages of the transformer model and self-attention mechanism to classify malicious messages in a CAN. The system effectively detects different attacks and improves the performance of the target model by applying transfer learning techniques. However, it is important to note that the methods mentioned above have limitations, because they cannot identify unknown or zero-day attacks and depends on sufficiently labeled data.

Although many models trained using supervised learning techniques have yielded promising results, amassing a large volume of labeled data for their training remains a significant hurdle. In most real-world scenarios, acquiring abnormal samples is challenging, and the availability of adequately labeled data is limited. Furthermore, certain models have difficulty in detecting zero-day attacks because supervised models fundamentally recognize only the patterns related to their training dataset [32]. However, a significant drawback of many unsupervised anomaly detection model training techniques is their limitation in implementing a multi-class attack classification framework, coupled with results that are often unimpressive [5]. These requirements motivated the development of semi-supervised learning algorithms that enhance the effectiveness of detecting intrusive messages while reducing the reliance on labeled samples during the training phase. To reduce reliance on labeled data, Hoang et al. [33] proposed a semi-supervised learning-based convolutional adversarial autoencoder (CAAE). Their system integrated two robust algorithms: a GAN and an autoencoder. By employing two discriminators, the CAAE was specifically designed to enhance the encoder phase to produce the outcomes for the classifier. Another approach, presented in [34], focused on maneuver classification. This method employed a simple CNN and learning strategy to categorize maneuvers into three groups: motion, velocity, and turning. Although this study did not require extensive volumes of labeled data, the results were not impressive. The reliance on a limited dataset often leads to issues of data imbalance and challenges in developing efficient multi-class classification models in existing research efforts.

### 3.2. Resampling of imbalanced data

As previously mentioned, the number of attack samples in most real datasets for CAN buses is often limited, leading to a pronounced class imbalance owing to the disparities in sample quantities. This problem is particularly severe for the supervised training of anomaly detection models. When deriving features from classes with sparse data, the prediction outcomes may be compromised by inherent biases or lead to model overfitting [35]. To address the issue of data imbalance, the resampling techniques are frequently employed to modify the distribution of both the minority and majority class samples in a primary dataset. This adjustment aims to boost the precision of detecting minority class instances and subsequently augment the overall efficacy of a model.

Yang et al. [36] used random oversampling and the synthetic minority oversampling technique (SMOTE) [37] to overcome the problem of class-imbalanced data in the CHD. Random oversampling simply duplicates samples to bolster minority classes but risks overfitting owing to overly specific learning. Conversely, the SMOTE algorithm uses the k-nearest neighbor concept to create new high-quality samples from minority classes for the proposed system. The results indicated that the proposed system outperformed existing methods by 2–3 % in terms of accuracy, detection rate, and F1 score, while also reducing the false-alarm rate. However, this method has limitations because it focuses only on oversampling data in minority classes. Meanwhile, Jin et al. [8] presented a similar concept but focused on analyzing imbalanced data ratios and removing outliers during data processing. In terms of resampling, the SMOTE method utilizes an oversampling strategy based on logarithmic ratio (OBLR) to reasonably equilibrate the numerical disparities among distinct categories. However, based on the experiments, this study indicated that SMOTE may amplify the effects of noisy data owing to the presence of outliers in minority classes. The challenge of optimizing sampling techniques requires an oversampling method that can recognize difficult-to-learn samples regardless of whether they belong to a minority or majority class. In another study, Caminero et al. [14] introduced a novel resampling strategy based on adversarial environment reinforcement learning (AERL) along with the SMOTE and its variants. Instead of concentrating only on the minority classes, this unique method uses an agent to perform an oversampling task. To fortify the model learning for a particular class, samples from classes that were regularly misclassified are presented at an increased frequency during the training process. By utilizing a reward function, predictability is enhanced through strategic adjustments and experience accumulated during training. This method mitigates the amplification of outliers by focusing primarily on resampling samples with inadequately extracted features. Thus, the performance of the classification agent is improved. In this study, we applied the AERL method to the IDS in IVN for the first time. In addition to the technique for handling data imbalance, the implementation of pseudo-labeling significantly contributes to the ability of the proposed model to classify various types of attacks.

## 4. Methodology

### 4.1. Overview of the proposed semi-supervised in-vehicle IDS

Our research is directed toward combining unsupervised and reinforcement learning to address two principal challenges in an in-vehicle IDS. The primary goal is to enhance training efficiency by minimizing the required volume of labeled data samples. Furthermore, we solve the data imbalance problem to improve the accuracy of the multi-class attack classification model. To leverage unlabeled data in semi-supervised learning, we employ two primary approaches: pre-training for self-supervised learning [32,33] and pseudo-labeling [6,38].

Pre-training involves initially training the neural network using unlabeled data before fine-tuning it with labeled data. The overall structure of the proposed system is shown in Fig. 3, wherein the training

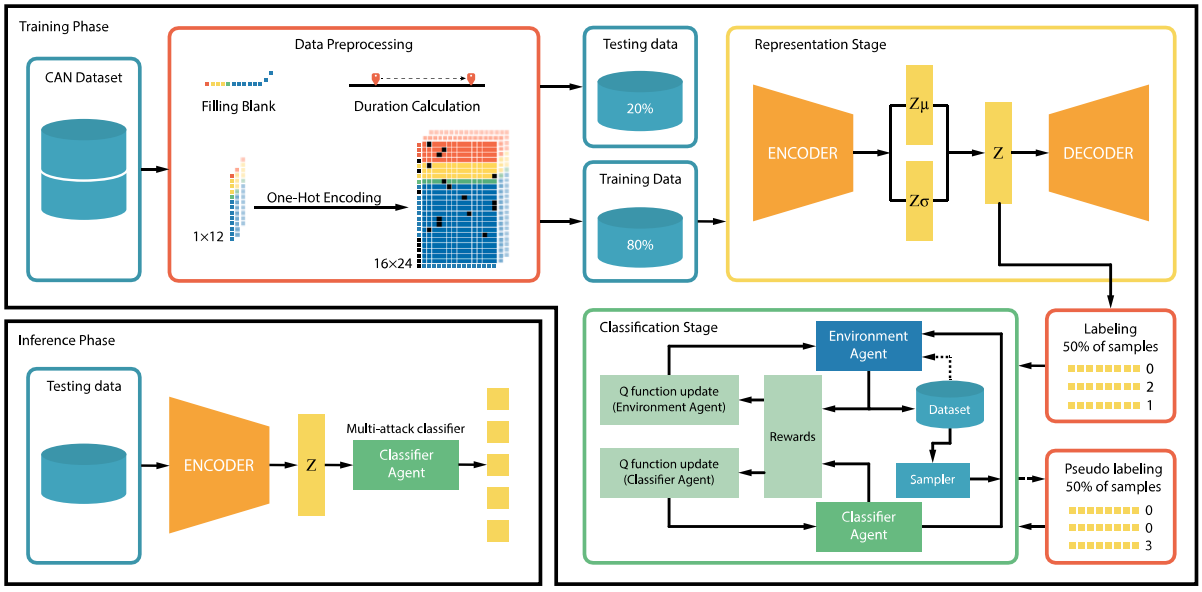



**Fig. 3.** Overall framework of the proposed system.

phase is divided into two stages: representation and classification. Representation involves the deployment of a fully unsupervised learning algorithm that maps the original sample to a low-dimensional representation without labels. Before applying an unsupervised learning algorithm, we preprocess the data from the CAN using one-hot encoding techniques, which enriches the information contained in the samples and provides statistical computation. The adoption of the VAE architecture, as an unsupervised learning algorithm, over other variants of autoencoders was motivated by the utilization of the advantages of the Gaussian probability density function to impose a distribution in the latent space. This transformation aims to provide a more compact and enhanced representation of the original data [39]. In the classification stage, we employ a multi-agent adversarial reinforcement learning approach. By applying two adversarial agents in the model, we categorize samples provided from the representation stage according to features indicative of different attack types while concurrently offering a solution to effectively ameliorate imbalances in the dataset.

On the other hand, pseudo-labeling requires a teacher model and concentrates on enhancing the student model. The teacher model, trained on existing labeled data, interprets all the unlabeled data. Subsequently, the student model is trained using both labeled and unlabeled data, by employing both ground-truth labels and pseudo-labels. Accordingly, we label half of the samples to train the teacher classifier. This classification model is then tasked with predicting the labels of the remaining unlabeled data. These samples are subsequently labeled based on their confident predictions. Finally, we build a comprehensive classification model using the combined dataset. This process enables the improvement of the classification model using only 50 % of the manual data labeling effort, provided that the teacher classifier achieves satisfactory performance.

We combine both methods using pre-training and pseudo-labeling for the proposed model. While previous studies have demonstrated that the pre-training method significantly enhances performance [23, 33], updating a new set of unlabeled data consumes substantial computational resources for the new representation phase. Pseudo-labeling allows for easier implementation on new unlabeled data since it only requires retraining from the classification model [38].

Despite the necessity for the encoder and the decoder in the representation stage, and the environment agent and the classifier agent in the classification stage during the training phase as shown in Fig. 3, the inference phase only requires the deployment of the encoder and the classifier agent. The function of the decoder block is to reconstruct the original input in VAE, thereby facilitating the optimization of the latent space. Accordingly, the decoder block is removed from the inference phase as illustrated in Fig. 3. Similarly, no environmental agent is required. Therefore, the proposed model can be built on a lightweight architecture, which enables the system to achieve low latency.

### 4.2. Data preprocessing

The quality of the data is crucial for determining the performance of models that utilize machine learning techniques. As previously mentioned, the original CAN data consist of features in hexadecimal format. Because most feature formats are not suitable for machine-learning algorithms, they must be transformed to facilitate computation. Redundant values are eliminated, and null values are appropriately filled in. To strongly utilize the correlation between the time and features of malicious attacks, we transformed the timestamp $T_n$ at the $n^{th}$ message into the duration $D_n = T_n - T_{n-1}$. Accordingly, the first CAN message is excluded from data processing. Commencing with the second message, the temporal interval between adjacent timestamps, which is identified as the inter-packet delay, is employed as a feature. Fig. 4 illustrates the process of the one-hot vector encoding of a CAN message. To simplify the encoding process, the durations are multiplied by $10^6$ and only the first four digits are considered. In the arbitration field, we are concerned only with the last three digits to eliminate unnecessary features. Because the null values are filled with 0, the length feature is necessary to distinguish between messages with the same payload post-filling.

The study in [40] indicated that processing CAN data using one-hot vector encoding yielded a better performance and reduced inference time. On the other hand, the data distribution analysis after applying one-hot vector encoding showed a certain threshold for facilitating the classification task. That is, one-hot vector encoding enhances the CAN features in a more detailed form and takes full advantage of the power of the probability distributions in the VAE model.

To perform one-hot vector encoding, we convert each element in the data features into a binary frame with 16 digits. The position of value 1 corresponds to the value of that component when converted from a hexadecimal space to a decimal space. For the 24 feature elements, each input message is expressed in $16 \times 24$ matrix form.

Subsequently, the CAN dataset is divided into training and evaluation samples. The proposed model uses all the unlabeled training data in

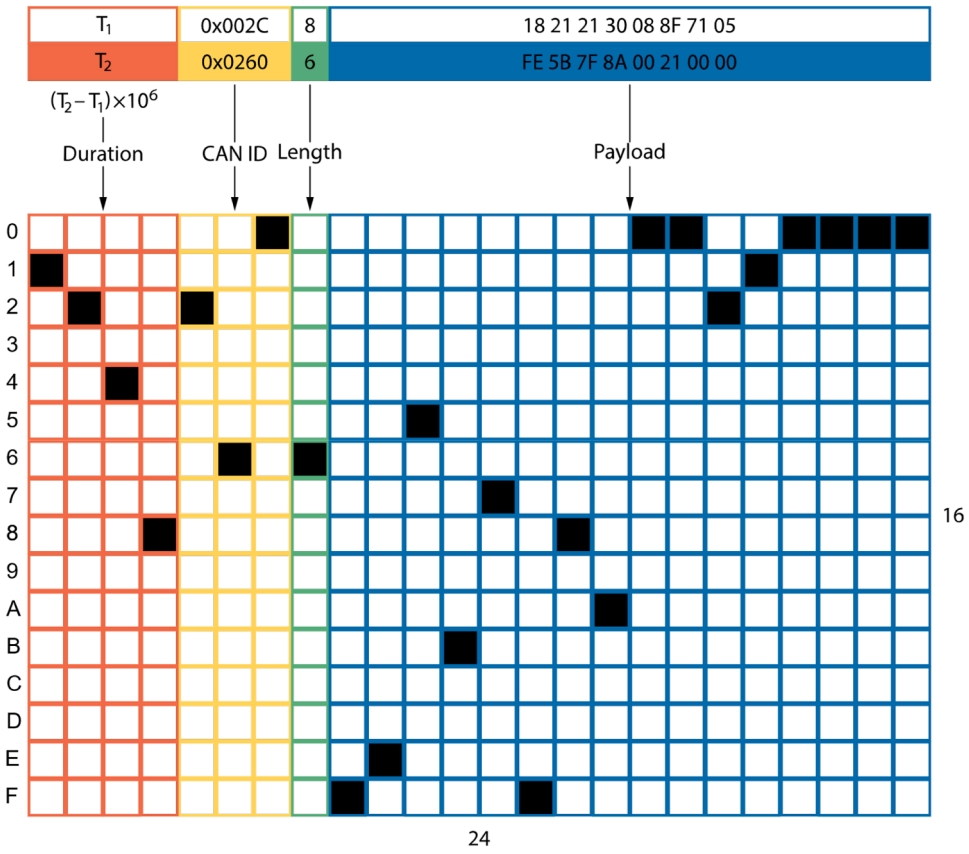



**Fig. 4.** Data preprocessing workflow.

the representation stage, whereas the classification stage requires only 50 % of the labeled data.

### 4.3. Representation stage using variational autoencoders

VAEs [41] are a type of generative model that learns the underlying probability distribution of data, allowing them to generate new data similar to the training data. They consist of two neural networks: an encoder that maps the input data to a latent space and a decoder that maps points from the latent space back to the original data space. The key idea behind VAEs is to use variational inference to approximate an intractable posterior distribution over the latent variables. This is achieved by introducing an inference model (encoder) that learns to map the input data to a distribution over the latent space. The encoder and decoder are trained jointly to maximize the lower bound on the log-likelihood of the data. The reparameterization trick enables efficient backpropagation through the stochastic layers of the VAE. Owing to flexible inference models and deeper generative models, which improves the accuracy and expressiveness, VAEs have diverse applications, including representation learning, semi-supervised learning, and data generation, demonstrating their significance in machine learning and artificial intelligence.

VAEs enforce a more organized and structured latent space, typically by imposing a prior distribution (such as a standard normal distribution) on the latent variables. This structure encourages similar data points to be mapped to nearby regions in the latent space, facilitating tasks such as anomaly detection and the generation of new meaningful data. By contrast, traditional autoencoders often have less structured latent spaces, making it difficult to generate new data or perform tasks that rely on understanding the relationships between data points in the latent space. Consequently, the encoding capabilities of VAEs are more comprehensive, effectively addressing the challenge of the nonregularized latent space existing in traditional autoencoders.

As shown in Fig. 5, the VAE approach comprises inference and generative networks. The inference network functions as an encoder, while the latter functions as a decoder. The VAE mechanism was inspired by the principles of the variational Bayes. Therefore, the VAE can return a latent space governed by probability distributions analogous to those characterizing the original CAN sample, thereby contributing to data enhancement.

Specifically, let the parameters for training the encoder and decoder be $\varnothing$ and $\theta$, respectively. The $i^{th}$ data point is denoted as $x^i$. The loss function of the VAE for $x^i$ comprises regularization and reconstruction and is expressed as

$$l_{VAE}(\theta, \varnothing, x^i) = D_{KL}(q_\varnothing(z|x^i) \parallel p(z)) - E_{q_\varnothing(z \mid x^i)}[\log p_\theta(x^i \mid z)] \quad (1)$$

The initial component of the loss function is the Kullback–Leibler (KL) divergence. The objective of the regularization term is to minimize the discrepancy between the distribution of the inference network $q_\varnothing(z|x)$ and the expected distribution $p(z)$. In the context of the VAE, $p(z)$ is determined as the standard deviation and standard normal distribution with a mean of zero, denoted as $\mathcal{N}(0,1)$. Accordingly, we penalize any deviation between each outcome $q_\varnothing(z|x)$ and the normal distribution. The second component corresponds to the expected negative log-likelihood. This term is commonly referred to as the reconstruction error because it motivates the generative network to learn how to reconstruct CAN samples efficiently. The reconstruction mechanism also allows the proposed model to evaluate the principal component analysis from the latent space without using data labels. For the effective training of the VAE, the loss function is modified through the reparameterization technique, enabling the sampling of a random variable $z$ from $p(z)$ as

$$l_{VAE}(\theta, \varnothing, x^i) = D_{KL}(q_\varnothing(z|x^i) \parallel p(z)) - \frac{1}{K}\sum_{k-1}^{K} \log p_\theta(x^i \mid z^{i,k}), \quad (2)$$

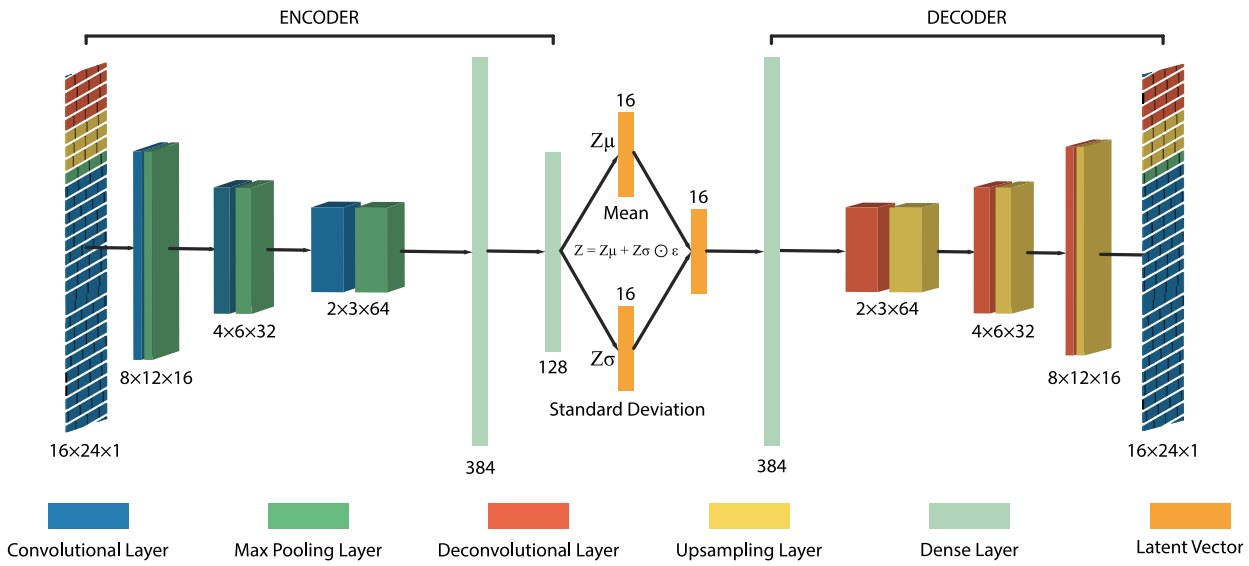



**Fig. 5.** VAE architecture in representation stage.

where $z^{i,k} = g_{\varnothing}(\epsilon^{i,k}, x^i)$ is the deterministic function, and $\epsilon^{i,k}$ stands for $\mathcal{N}(0,1)$ [41]. In other words, instead of directly sampling $z^{i,k}$ from the distribution $q_{\varnothing}$, this technique involves sampling from $\mathcal{N}(0,1)$ and then applying a differentiable function $g_{\varnothing}$.

Fig. 5 illustrates the representation stage in detail, which mainly consists of convolutional and deconvolutional layers. By employing one-hot encoding, an input configuration with dimensions of $24 \times 16$ is used. In the inference network, the convolutional layers are sequentially alternated with max-pooling layers with a filter size that increases incrementally from 16 to 64. Conversely, the opposite segment is structured with alternating deconvolutional and upsampling layers, each with a corresponding size. The symmetrical architecture of the model enhances the efficient reconstruction of the input samples and eliminates the need for additional input padding or output cropping procedures. The outcome of our VAE model is manifested as a latent space configured with a dimensional size of 16.

After the training phase, we leverage only the encoder of the VAE in the inference phase. Through the latent space, the CAN samples are dimensionally reduced while retaining the most important features. Furthermore, probability distributions take advantage of correlated features and facilitate the division of data into distinct regions in a new space.

### 4.4. Classification stage using adversarial environment reinforcement learning

Caminero et al. [14] first introduced the Adversarial Environment Reinforcement Learning (AERL) algorithm for network intrusion detection. The authors highlighted the challenges of traditional IDS in handling large, noisy, and imbalanced datasets, and proposed AERL, which integrates a simulated environment with an agent (classifier) to predict intrusion labels from network traffic samples. The environment generates rewards based on the accuracy of the agent's predictions, and it also adapts its behavior to challenge the classifier, thereby improving its performance. Since training a multi-class classification model in IVNs also raises similar concerns regarding data imbalance, we establish a mechanism designated as an environmental agent to address this issue.

The AERL design in Fig. 6 is structured on a multi-agent reinforcement learning architecture where one agent is deployed in the environment to handle data imbalance, whereas the other agent functions as a classifier to optimize the classification of multiple attacks. The primary objective of AERL is to enable an agent serving as a classifier to adapt and enhance its learning for minority classes. Suppose that wrong predictions come from samples that are difficult to predict or rarely appear during training. By generating samples where the classifier frequently predicts incorrectly, AERL enables the environment agent to implement

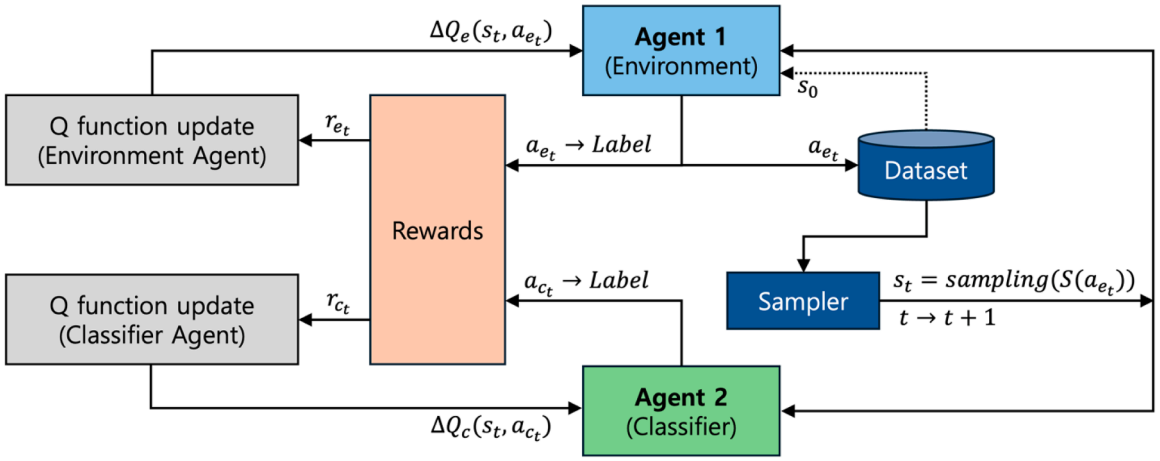



**Fig. 6.** AERL architecture in classification stage.

a strategy for creating more balanced data sampling. On the other hand, the environmental agent is penalized if the selected samples do not challenge the classification agent. The fundamental purpose of the environmental agent is to assess the overall performance and select the most challenging categories of data for the classifier agent to train. Hence, the classifier agent is continually compelled to train on the most difficult samples currently available, effectively balancing the distribution of the training samples. Utilizing an adversarial with the environment agent, a more evenly distributed dataset is created, potentially enhancing the accuracy. This approach not only concentrates on minority classes but also carefully learns samples that challenge the classification agent, thereby improving its malicious CAN message detection capabilities.

After completing the representation stage, the labeled dataset will be passed through the latent space, receive labels, and then become accessible within the AERL system environment. AERL training consists of two phases: training a teacher model for the pseudo-labeling task, and then training a student model by mixing ground-truth labeled data with a pseudo-labeled dataset. The AERL algorithm comprises key elements such as the agent, state, action, and reward, as presented in Table 2. The classification agent outputs intrusion classes, after which the environmental agents generate actions to select new samples to challenge the classification agent. During training, the environment agent and the classifier agent are both trained in parallel, utilizing the Deep Q-Network (DQN). DQN is built on the Q-learning algorithm, which is used to find the optimal Q-function for the agent. In Q-learning, the Q-function determines a value for each state-action pair and is updated iteratively using the following expression [42]:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_A Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (3)$$

where $S_t$ represents the current state, $A_t$ represents the current action, $A_{t+1}$ represents the next action, $R_{t+1}$ represents the next reward, $\alpha$ and $\gamma$ indicates the learning rate and discount factor, respectively.

To estimate the Q-function, DQN employs a deep neural network (DNN) as shown in Fig. 7. The size of the hidden layers is selected to be divisible by eight to optimize the computational performance on Tensor Cores [43]. The size of the output is determined by the number of attack types in each dataset, which amounts to five for CHD and seven for ROAD, respectively. We use the same DNN framework for both environment agent and classifier agent. We feed the current state into this neural network, and the output is the Q-function for all possible actions. To update the Q-function, we use the quadratic loss function as follows:

$$(R_{t+1} + \gamma max_A Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))^2 \quad (4)$$

By applying adversarial optimization, any positive rewards for the classification agent turn into negative rewards for the environment. In the classification agent, correct predictions are assigned a value of +1 and incorrect predictions a value of –1. Conversely, the environmental agent will receive a reward value of –1 for each correct prediction, and +1 for each incorrect prediction. In this case, each agent must be optimized using separate Q-functions, in which $Q_c(s, a_c)$ performs optimization for the classifier agent and $Q_e(s, a_e)$ performs optimization for the environmental agent. The training process is summarized in Algorithm 1, where the classifier and environment agents are jointly optimized in an adversarial manner.

First, we initialize the Q-function values $Q_e(s, a_{c_i})$ and $Q_c(s, a_{e_i})$ for both the environment and classifier agents, respectively. For each episode, the initial state value $s_0$ is a randomly selected sample from the dataset. From the current state, we obtain the initial action $a_{e_0}$ from its corresponding policy. In our approach, the action of environment agent $a_{e_0} \in [0; C]$ is a data class. After obtaining $a_{e_0}$, we sample $s_0$ again; this state depends on the class outcome selected from the action. In the given state from the environment, the classifier agent attempts to categorize the data points following the policy and then assigns their predicted class to an action $a_{c_i}$ as a standard DQN algorithm. We then compare $a_{c_i}$ with the ground truth data $a_{e_0}$. The reward functions $r_{c_i}$, $r_{e_i}$ are designed to optimize the training strategies of the classification and environment agents, respectively. A positive reward is awarded to the classification agent if the prediction is accurate. Conversely, if the outcome $a_{c_i}$ is incorrect, the environment agent receives a positive reward. These two agents function based on adversarial principles, and enhance the accuracy of the classification agent. In the next state derivation, the new state $s_{i+1}$ from the environment agent is refreshed after each step following the outcome action $a_{e_{i+1}}$ and is provided with a new feature-label pair $(s_{i+1}, a_{e_{i+1}})$. During the training episodes, the policy functions of both the environment and classifier agents are continuously updated based on the reward value and the next inference state.

**Table 2**
AERL components and corresponding implementations.

| Component | Environment agent | Classifier agent |
|---|---|---|
| State | Sample | Sample |
| Action | Choose next class | Predict class |
| Reward | +1 for incorrect prediction<br>-1 for correct prediction | +1 for correct prediction<br>-1 for incorrect prediction |

## 5. Experimental results

This section presents a comprehensive performance analysis of the proposed IDS, utilizing two public CAN datasets. We first introduce the datasets employed in our experiments, followed by a description of the performance metrics used to evaluate the IDS. Subsequently, we present and discuss the results obtained for both known and unknown attacks, comparing them with state-of-the-art methods. Finally, we assess the computational time required to validate the feasibility of real-time deployment. The train/test split ratio was set at 8:2. The VAE was trained on the full dataset without utilizing labels, whereas the AERL was trained on 50 % of the labeled dataset for the proposed teacher and student models. Each experiment, encompassing the entire process from dataset splitting to evaluation, was conducted five times, and the average results were reported. The experiments were performed on a server equipped with an Intel Xeon Silver 4216 CPU, an NVIDIA GeForce RTX 4090 GPU, 256 GB of RAM, and a 512 GB hard drive, running Ubuntu 20.04. The proposed model was implemented using Python 3.9.17 and PyTorch 2.1.0.

### 5.1. Dataset

Two CAN datasets were used to evaluate the performance of the proposed model. The first dataset is the CHD [28], which is widely used to measure the performance of models. The second dataset is the ROAD [44], which is considered to be a modern and enhanced CAN bus dataset. As shown in Figs. 8 and 9, both CHD and ROAD are extremely imbalanced.

### 5.2. Car-hacking dataset (CHD)

The CHD was recorded in a real IVN, obtained via the OBD-II port during message injection attacks in the CAN, and was injected into four attack scenarios. This is a dataset with visual features, each of which comprises a timestamp, CAN ID, DLC, and 8-byte payload. Providing labels for each message is advantageous and makes the CHD popular in CAN studies and performance evaluations for intrusion detection. Fig. 8 illustrates the types of attacks, their sizes, and the injection message rates of the CHD. For DoS attacks, the CHD contains a total of 3,665,771 data samples, with 84 % classified as normal data and 16 % as intrusion data. For Fuzzy attacks, the dataset includes 3,838,860 samples, of which 87 % are normal data and 13 % represent intrusion data. Regarding Gear Spoofing and RPM Spoofing attacks, the CHD dataset comprises 4,443,142 and 4,621,702 samples, respectively. For Gear

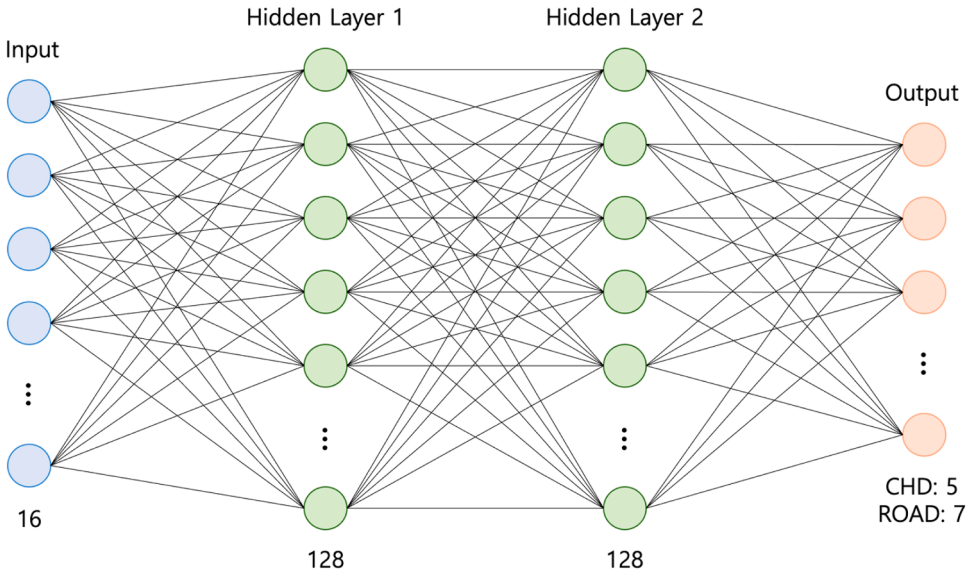



**Fig. 7.** DNN framework for environment agent and classifier agent.

**Algorithm 1**
Algorithm for the AERL classification model.

**Input**:
-Dataset: $\mathscr{D}$ includes $\{\boldsymbol{X}_i, y_i\}$
where $0 \leq i < N$, $N$ is the total number of samples, $\boldsymbol{X}_i \in R^{16}$, and $y_i \in [0; C]$.
**Output**:
-the weight for classifier: $\boldsymbol{\theta}_C$
Initialize $Q_c(s, a_c)$ arbitrarily.
Initialize $Q_e(s, a_e)$ arbitrarily.
For each episode:
Initialize the state value: $s_0 = random\ sampling(\mathscr{D})$
Environment agent: choose initial action $a_{e_0}$ using policy from $Q_e(s_0, a_{e_0})$
$s_0 = random$ sampling $(S(a_{e_0}))$, where $S(a_{e_0})$ are all samples labeled as $a_{e_0}$
For each sample in episode:
Classifier agent: choose action $a_{c_i}$ using policy from $Q_c(s_i, a_{c_i})$
Obtain rewards: $r_{c_i}$, $r_{e_i}$
Derive next state:
Environment agent: choose action $a_{e_{i+1}}$ using policy from $Q_e(s_i, a_{e_i})$
$s_{i+1} = random$ sampling $(S(a_{e_{i+1}}))$, where $S(a_{e_{i+1}})$ are all samples labeled as $a_{e_{i+1}}$
Update Q-function:
Environment agent: $\left(r_{e_i} + \gamma max_{a_{e_{i+1}}} Q_e(s_{i+1}, a_{e_{i+1}}) - Q_e(s_i, a_{e_i})\right)^2$
Classifier agent: $\left(r_{c_i} + \gamma max_{a_{c_{i+1}}} Q_c(s_{i+1}, a_{c_{i+1}}) - Q_c(s_i, a_{c_i})\right)^2$

Spoofing attacks, 82 % of the data are normal data and 18 % are intrusion data, whereas for RPM Spoofing attacks, 78 % of the data are normal and 22 % are intrusion data.

### 5.3. Real ORNL automotive dynamometer dataset (ROAD)

The ROAD is currently the most representative and available dataset because of its diversity and provides more comprehensive attack types such as accelerators, targeted ID fabrication, and masquerade attacks. We evaluated the proposed model using the fabrication datasets listed in Table 3. In particular, the flam delivery technique was used to perform dynamic injections, making it difficult to detect malicious CAN messages. The ROAD features are fully provided in CHD, in that data labeling can be extracted through metadata. Fig. 9 presents the attack types, sizes, and injection message rates of the ROAD. In the case of Fuzzy (FZ) attacks, the ROAD comprises 94,931 samples, of which 99 % are classified as normal data, and 1 % are categorized as intrusion data. For Correlated signal (CS) attacks, there are 192,748 samples, with normal data accounting for 97 % and intrusion data accounting for 3 %. Regarding Max engine coolant temp (MECT) attacks, the ROAD contains a total of 61,923 samples, with an overwhelming 99.94 % being normal data and only 0.06 % representing intrusion data. For Max speedometer (MS) attacks, the dataset features 572,842 samples, with 98 % of the

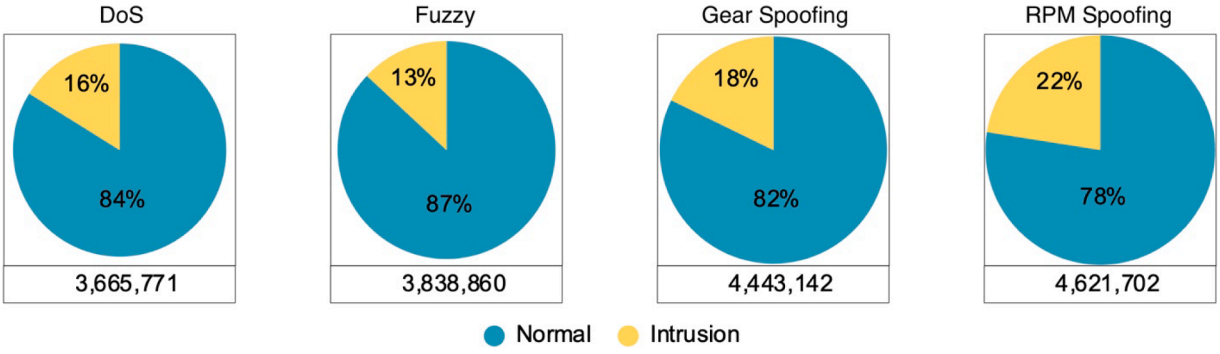



**Fig. 8.** CHD injection rate analysis.

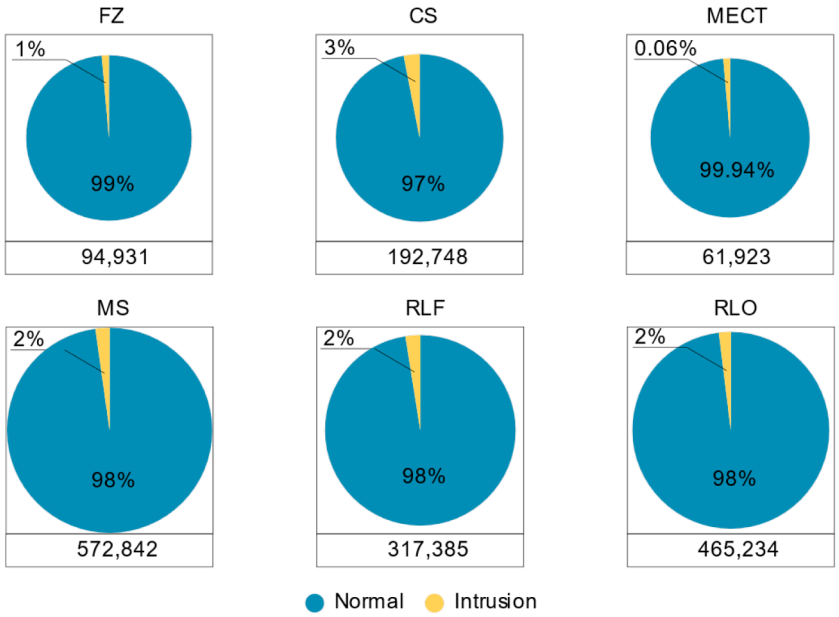



**Fig. 9.** ROAD injection rate analysis.

data labeled as normal and 2 % as intrusion data. Finally, Reverse light off (RLF) and on (RLO) attacks consist of 317,385 and 465,234 samples respectively, both characterized by 98 % normal data and 2 % intrusion data.

### 5.4. Evaluation metrics

To thoroughly evaluate the performance of the proposed model, we utilized two distinct datasets and assessed a variety of scenarios categorized as either known or unknown. The evaluation of known attacks was understood in terms of the classes in the test data that appear during the training phase. Conversely, the scenarios for evaluating unknown attacks were assumed for each dataset. Unknown attacks were evaluated for the three scenarios listed in Table 4. In the CHD, we segregated each class for independent evaluation and subsequently trained the classification model after excluding the class from the dataset. For the ROAD, we selected datasets that were similar to those of the injected malicious data. These characteristics exist in the arbitration field or payload. Furthermore, we evaluated the performance of our IDS in detecting unknown attacks on dissimilar datasets.

The accuracy, precision, recall, F-measure, and rates of false positives and negatives were selected as performance metrics to assess the efficacy of the proposed system. The false-negative rate, also known as the misdetection rate, correlates with a substantial number of undetected intrusions and presents a risk to drivers in serious attack scenarios. Conversely, a high false-positive rate, referred to as the false-alarm rate, annoys users with inaccurate alarms. Thus, in the range 0–1, a system is deemed accurate if it has a high F1 score, and low misdetection and false-alarm rate. Based on the outcomes derived from the evaluation matrix, the evaluation metrics are expressed as follows:

$$Precision\ (Pre) = \frac{True\ Positive}{True\ Positive + False\ Positive}, \tag{5}$$

$$Recall\ (Rec) = \frac{True\ Positive}{True\ Positive + False\ Negative}, \tag{6}$$

$$F1\ score\ = 2 \times \frac{Pre \times Rec}{Pre + Rec}, \tag{7}$$

$$False\ Positive\ Rate\ (FPR) = \frac{False\ Positive}{False\ Positive + True\ Negative} \times\ 100\%, \tag{8}$$

$$False\ Negative\ Rate\ (FNR) = \frac{False\ Negative}{False\ Negative + True\ Positive} \times\ 100\%. \tag{9}$$

### 5.5. Improving imbalanced CAN data with AERL

To evaluate the effectiveness of the AERL algorithm on the CAN data, we performed supervised classification training on both datasets. Figs. 10 and 11 illustrate the temporal progression of the sample distribution of the environmental agent during training. We trained the multi-agent reinforcement learning model for 200 episodes. The sampling process was visualized every 40 episodes. In the CHD, the

**Table 3**
High-frequency injection (fabrication) attacks on ROAD.

| Attack type | Injection technique | Impact |
|---|---|---|
| Fuzzy (FZ) | Inject random IDs and arbitrary payloads | Causes a variety of unexpected dangerous behaviors. |
| Correlated signal (CS) | Inject false wheel-speed values (ID-6E0) | Loss of control by changing the speed of four wheels. |
| Max engine coolant temp (MECT) | Change one byte of payload to maximum (FF) value (ID-4E7) | "Engine coolant too high" warning light illuminates on dash. |
| Max speedometer (MS) | Change one byte of payload to maximum (FF) value (ID-0D0) | Speedometer falsely displays a maximum value. |
| Reverse light off (RLF) and on (RLO) | Change one bit of payload (ID-0D0) | Reverse lights do not reflect current car gear. |

**Table 4**
Scenarios for unknown attack evaluation on ROAD.

| Scenario | Training dataset | Testing dataset |
|---|---|---|
| Same injection ID (0xd0) | RLF, RLO | MS |
| Same injection Payload (XXXXXXXXXXFFXXXX) | FZ, MCT | MS |
| Uncorrelated injection | FZ, MS, MCT, RLF, RLO | CS |

histograms in Fig. 10 show that starting from episode 40, the sampler decreased the sampling frequency in DoS attacks, and the normal samples started to decrease from episodes 80. Furthermore, the agents focused on spoofing and fuzzy attacks, particularly fuzzy attacks, indicating that the model had difficulty in classifying samples belonging to these two attack types. This behavior can be explained by the fact that the messages in a fuzzy attack are injected randomly, whereas the malicious messages in a DoS attack mostly have a certain format; for example, CAN ID of $0 \times 000$ which has a highest priority. The results for the ROAD indicate similar signs when the data belonging to the normal class were sparsely sampled. By contrast, the environment agent strongly focused on sampling in the MCT class, which had the most obvious data imbalance when the intrusion rate reached only 0.06 %.

To evaluate the effectiveness of multiple attack classification, Fig. 12 presents a comparison between AERL and other machine-learning algorithms: support vector machine (SVM), decision tree (DT), and DNN. The results demonstrate that AERL is suitable for designing an IDS in an in-vehicle environment. In CHD, all the classification models encountered insignificant challenges with fuzzy attacks. The AERL algorithm proved to be stable as it outperformed the other baseline models on the ROAD. For classes with severe data imbalance, the AERL model significantly improved the F1 score compared with the baseline models.

### 5.6. Results on known-attack detection

For the known attacks, we compared the proposed model with other baseline models as presented in Tables 5 and 6. Each baseline represents a model type, SVM for a kernel-based, DT for a tree-based and DNN for modeling neural networks. These models were trained using all labeled data. In CHD, we compared the proposed model with the CAAE model [33], which is mentioned in Related Works section. Both models use a pre-training method, while the proposed model is more advanced in using pseudo-labeling and solving the problem of data imbalance. Additionally, we compared our proposed model with the state-of-the-art model, ECF-IDS [45], which leverages Bidirectional Encoder Representation from Transformers (BERT) and an enhanced cuckoo filter for intrusion detection in IVNs. In terms of ROAD, to the best of our knowledge, previous studies on the ROAD have not mentioned multi-class classification. Therefore, we compared the performance of our proposed model with that of deep evolving stream clustering (DESC) [17] that was trained using an unsupervised learning algorithm. Owing to the limitations of unsupervised learning, DESC was implemented as a binary classification model. Furthermore, we evaluated our model against the state-of-the-art LSF-IDM [46]. Although LSF-IDM utilizes semantic fusion for effective intrusion detection, it is restricted to binary classification. The performance of the proposed model was evaluated through two experiments. The proposed teacher system, whose classification model was trained with only half the amount of labeled data, is referred to as the proposed teacher model. The second proposed student model employs a mix of labeled data and pseudo-labeled data generated by the teacher model. Because of the class imbalance, we used the macro average scores as the evaluation metric.

The results showed that the proposed teacher model outperformed baseline models when only half of the labeled data were used. Based on the empirical evaluations conducted on both datasets, the baseline models achieved positive results as most of the F1 scores exceeded 0.95. Our teacher model also achieved positive results with an overall F1 score greater than 0.98 for both datasets, even though our teacher model used only half of the labeled data. When observing the FPR, we found that our proposed teacher model significantly diminished the false-alarm rate to below 5 % in the ROAD and to less than 1 % in the CHD. The best outcomes for the baseline models demonstrated false-alarm rates exceeding 5 % for CHD and 14 % for ROAD, which were significantly higher than our rates. Furthermore, our teacher model not only reduced the false-alarm rate but also showed that the missed detection rate was less than 1 % and less than 2 % for the CHD and ROAD, respectively. We utilized this model in conjunction with a pseudo labeling technique, where securing high F1 scores and low error rates are essential for enhancing the overall effectiveness of the second model.

In terms of utilizing the combined data, the student model was improved and surpassed the performance of the teacher model, which uses only half of the labeled data without the pseudo labeling technique. The results also demonstrated the effectiveness of combining the analysis of data features through a probability distribution and reducing the data imbalance rate using a multi-agent adversarial reinforcement learning algorithm. In CHD, the student model enhanced the performance, as demonstrated by the significant reduction in FPR and FNR. Similarly, the error rates in this model decreased by more than three times for FPR and decreased from 1.892 % to 1.106 % for FNR on ROAD. Moreover, our student model exhibited comparable performance to the state-of-the-art methods ECF-IDS on CHD and LSF-IDM on ROAD in terms of Recall, Precision, and F1 score. It is worth noting that our model achieves this while utilizing only half of the labeled data and performing multi-class intrusion detection, whereas ECF-IDS and LSF-IDM leverage all the labeled data, and LSF-IDM is limited to binary-class intrusion detection. The results of the teacher model demonstrated the critical contributions of the VAE model in the pre-training phase. Furthermore, we believe that addressing data imbalance and implementing pseudo-

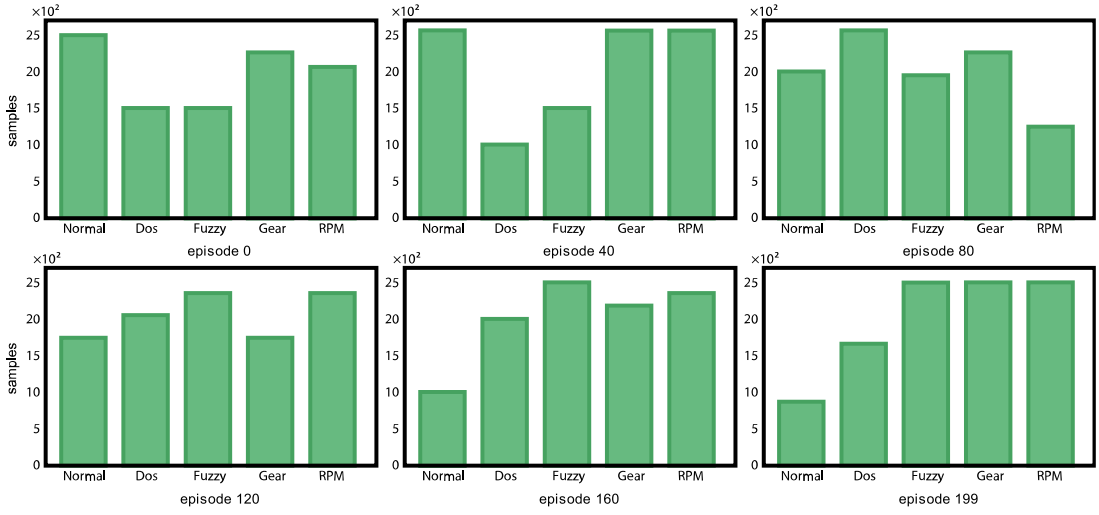



**Fig. 10.** Evolution of sampled class distribution on CHD.

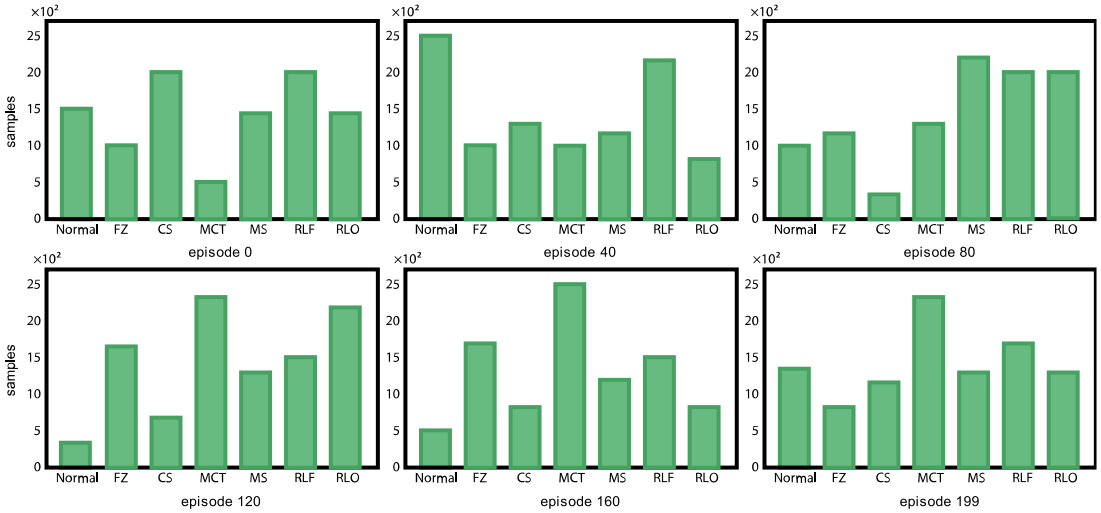



**Fig. 11.** Evolution of sampled distribution on ROAD.

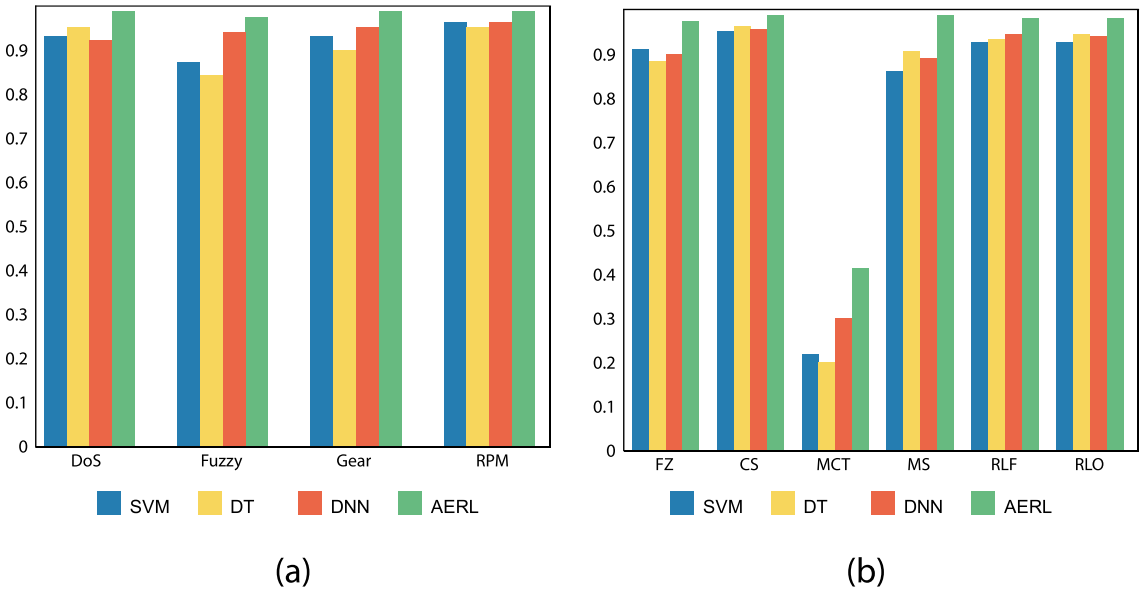



**Fig. 12.** Performance comparison between AERL and base models: (a) CHD and (b) ROAD.

**Table 5**
Model comparisons for known attacks on CHD.

| Model (percentage of labeled data) | Rec | Pre | F1 score | FPR (%) | FNR (%) |
|---|---|---|---|---|---|
| SVM (100 %) | 0.9774 | 0.9713 | 0.9742 | 14.807 | 2.274 |
| DT (100 %) | 0.9771 | 0.9748 | 0.9756 | 13.616 | 2.271 |
| DNN (100 %) | 0.9832 | 0.9862 | 0.9854 | 6.808 | 1.624 |
| CAAE (40 %) | 0.9972 | 0.9997 | 0.9984 | - | - |
| ECF-IDS (100 %) | 0.9998 | **0.9999** | 0.9998 | - | - |
| Proposed teacher (50 %) | 0.9999 | 0.9999 | 0.9999 | 0.08 | 0.091 |
| Proposed student (50 %*) | **0.9999** | **0.9999** | **0.9999** | **0.0** | **0.001** |

**Table 6**
Model comparisons for known attacks on ROAD.

| Model (percentage of labeled data) | Rec | Pre | F1 score | FPR (%) | FNR (%) |
|---|---|---|---|---|---|
| SVM (100 %) | 0.9435 | 0.9411 | 0.9425 | 31.917 | 5.608 |
| DT (100 %) | 0.9744 | 0.9411 | 0.9575 | 31.917 | 2.542 |
| DNN (100 %) | 0.9741 | 0.9716 | 0.9739 | 14.896 | 2.542 |
| DESC (100 %) | 0.9573 | 0.9133 | 0.9348 | - | 4.270 |
| LSF-IDM (100 %) | **0.9970** | 0.9911 | **0.9938** | - | - |
| Proposed teacher (50 %) | 0.9817 | 0.9905 | 0.9852 | 4.684 | 1.892 |
| Proposed student (50 %*) | 0.9842 | **0.9966** | 0.9921 | **1.425** | **1.106** |

* 50 % labeled ground-truth data combined with 50 % pseudo-labeled data.

labeling techniques play a key role in significantly lowering error rates across both the teacher and student models. Given these advantages, we assume that through the feature extraction process, the proposed model is adept at retaining and emphasizing the fundamental features of malicious samples. This premise motivated our subsequent experiments to evaluate the performance of the proposed model against unknown attacks, the details of which are presented in the following section.

### 5.7. Results on unknown-attack detection

Table 7 presents the performance comparisons between the proposed system and those of previous studies on unknown attacks on the CHD. We used a new dataset, which accounted for 30 % of the total labeled data. We reused the latent space from the above experiments. The

baseline models employed were selected based on their capacities to detect unknown attacks. In particular, the multi-tiered hybrid (MTH) system in [47] required entire labeled data for signature-based supervised learning and then utilized unsupervised learning to identify unknown attacks. Meanwhile, the self-supervised learning (SSL) in [6] is a binary classification, which uses only a normal dataset and abnormal samples generated by itself for the training phase. We compared two semi-supervised learning algorithms: the CAAE model in the [33] and our proposed model. To ensure a fair comparison, we employed the proposed student model with only 30 % of the labeled data, as CAAE also utilizes only 30 % of the labeled data. We also compared our model with the state-of-the-art model named CLUSTER [48], which is a clustering-based open set recognition approach for identifying unknown attacks in CAN bus intrusion detection systems.

The results in Table 7 indicate that the SSL model was challenged by the complexity of fuzzing attacks. According to the F1 score, this model achieved high accuracy when exceeding the threshold of 0.9 for DoS and spoofing attacks. By contrast, the F1 score for fuzzy attacks was only 0.8861, which was unimpressive owing to the low recall value of 0.8345. The MTH system proved effective against spoofing and DoS attacks, as all the results reached the best score. However, this algorithm still had difficulty in facing fuzzy attacks and only achieved an F1 score of 0.8439. Furthermore, the MTH model was trained with full CHD data, whereas our proposed model only utilized half the data. The CAAE model demonstrated good performance when the F1 score exceeded 0.91 for most unknown attack types, particularly for fuzzy attacks reaching 0.9145. For the same data size, the comparisons demonstrate that the proposed model achieves impressive performance, surpassing the baseline models for fuzzy and gear spoofing attacks. Finally, CLUSTER demonstrated excellent performance with an F1 score of 0.98 or higher against all attacks. Notably, it achieved a very high F1 score of 0.995 against fuzzy attacks, outperforming the other models. However, it exhibited a slightly lower detection performance against spoofing attacks than the proposed model. Overall, CLUSTER outperformed the proposed model in DoS and fuzzy attacks, while the proposed model excelled in spoofing attacks. It is important to note that CLUSTER utilized all the labeled data, whereas the proposed model only used 30 % of the total labeled data.

We also compared evaluation analyses of the proposed and MTH models to detect unknown attacks on the 30 % ROAD dataset. To the best of our knowledge, this study is the first to evaluate the detection of unknown attacks on the ROAD. The results of the scenarios are presented in Table 8. Comparisons between the models indicated that the MTH model achieved a slightly better F1 score in Scenarios 1 and 2. However, the proposed model had a false-alarm rate nearly 10 times lower than that of the MTH in Scenarios 1 and 3. In the scenarios in which the proposed model effectively reduced the false-alarm rate, the normal sample space was larger. In Scenario 2, the performances of the proposed and MTH models were almost identical. Using the same test dataset, the results from Scenarios 1 and 2 indicated that the presence of common characteristics within the payload led to a reduction in the missed detection rate. In other words, these characteristics proved more effective than similarities in the arbitration field. In Scenario 3, our model reduced the false alarm rate to 0.013 % more effectively than the MTH's false alarm rate of approximately 0.136 %. It should be noted that our proposed model used only 50 % of the labeled data.

**Table 7**
Model performance comparison for unknown attacks on CHD.

| Attack type | Model (Percentage of labeled data) | Rec | Pre | F1 score |
|---|---|---|---|---|
| DoS | SSL** | 0.9916 | 0.9751 | 0.9833 |
| | MTH (100 %) | **1.00** | **1.00** | **1.00** |
| | CAAE (30 %) | 0.9823 | 0.9992 | 0.9907 |
| | CLUSTER (100 %) | 0.9937 | 0.9937 | 0.9937 |
| | Proposed (30 %) | 0.9854 | 0.9995 | 0.9924 |
| Fuzzy | SSL** | 0.8345 | 0.9445 | 0.8861 |
| | MTH (100 %) | 0.7305 | 0.9990 | 0.8439 |
| | CAAE (30 %) | 0.8426 | **0.9999** | 0.9145 |
| | CLUSTER (100 %) | **0.9934** | 0.9936 | **0.9935** |
| | Proposed (30 %) | 0.8539 | 0.9975 | 0.9201 |
| Gear spoofing | SSL** | 0.8803 | 0.9768 | 0.9261 |
| | MTH (100 %) | **1.00** | 0.9948 | 0.9974 |
| | CAAE (30 %) | 0.9978 | 0.9977 | 0.9977 |
| | CLUSTER (100 %) | 0.9892 | 0.9893 | 0.9892 |
| | Proposed (30 %) | 0.9962 | **0.9996** | **0.9979** |
| RPM spoofing attack | SSL** | 0.9997 | 0.9720 | 0.9850 |
| | MTH (100 %) | **1.00** | **1.00** | **1.00** |
| | CAAE (30 %) | 0.9955 | 0.9984 | 0.9970 |
| | CLUSTER (100 %) | 0.9807 | 0.9809 | 0.9808 |
| | Proposed (30 %) | 0.9955 | 0.9993 | 0.9974 |

** Unsupervised learning model.

### 5.8. Model complexity analysis

To be effectively integrated and deployed within an ECU, a CAN-bus IDS must adhere to stringent hardware constraints, including limited computational power, small memory capacity, and low response time. In this section, we evaluate the feasibility of deploying the proposed student model in real-world environments. For this purpose, we selected the NVIDIA Jetson AGX Xavier platform, which features a 512-core Volta GPU with Tensor Cores, an 8-core ARM CPU, and 32 GB of RAM. This platform was chosen due to its suitability for integration into a vehicle's XPU, allowing it to detect intrusions on the CAN bus [29], [30]. We trained the proposed student model on a high-performance server equipped with an Intel Xeon Silver 4216 CPU and an NVIDIA GeForce RTX 4090 GPU, utilizing half of the labeled CHD dataset. The resulting model weights were then transferred to the Jetson AGX Xavier for deployment. The inference time was recorded for each message in the test set (20 % of the dataset) of CHD across five iterations, and the final result represents the average of these measurements.

Table 9 provides a comparison of the memory usage and inference time on the Jetson AGX Xavier platform with the other models. Although the CLAM model occupies the least memory among the compared methods, our proposed model requires approximately four times more memory, totaling 2.542MB. Nevertheless, we believe that our proposed model is still feasible for real-time vehicles, considering that 2.542MB is a relatively small portion of the 32GB total memory available on the Xavier platform. Moreover, when evaluating the inference time, which is a critical metric for real-time IDS performance, our proposed model demonstrates the fastest inference time, 3.21ms, outperforming all other methods. This efficiency can be attributed to the streamlined lightweight architecture of our model, which consists of only a straightforward encoder and a simple classifier for attack detection. These comprehensive evaluations indicate that the proposed model is not only lightweight but also highly suitable for practical deployment in IVNs, making it an excellent candidate for real-world implementation.

## 6. Conclusions

Our research aimed to develop an optimized IDS for IVNs that minimized the number of required labeled samples and to address data imbalance issues during training. During training, we designed two phases, representation, and classification, which learned distributed features from unlabeled data and leveraged them for multi-class classification. In this manner, we incorporated the principal component analysis capabilities of the VAE, and a sampling method based on adversarial multi-agent reinforcement learning.

The performance of the proposed system was evaluated using two well-known datasets covering a variety of attack types. The results demonstrated that the AERL method effectively mitigated the negative impacts of data imbalances. With only half of the labeled data, our proposed IDS achieved a high performance compared to that of the baseline models in classifying known attacks. Furthermore, our system

**Table 8**
Model performance comparison for unknown attack on ROAD.

| Experiment | Attack type | Model (Percentage of labeled data) | Rec | Pre | F1 score | FPR (%) | FNR (%) |
|---|---|---|---|---|---|---|---|
| Scenario 1 | MS | MTH (100 %) | **0.9573** | 0.9098 | 0.9330 | 0.196 | **4.260** |
| | | Proposed (50 %) | 0.8551 | **0.9854** | **0.9156** | **0.026** | 14.486 |
| Scenario 2 | MS | MTH (100 %) | 0.9056 | **0.9051** | **0.9054** | **0.196** | **9.435** |
| | | Proposed (50 %) | **0.9064** | 0.9022 | 0.9043 | 0.204 | 9.355 |
| Scenario 3 | CS | MTH (100 %) | 0.7281 | 0.9400 | 0.8206 | 0.136 | 27.180 |
| | | Proposed (50 %) | 0.7281 | **0.9937** | **0.8405** | **0.013** | 27.180 |

**Table 9**
Training and inference cost comparisons.

| Model | Memory footprint (KB) | Inference time (ms) |
|---|---|---|
| CLAM [29] | 682 | 5.7 |
| SupCon [30] | 5,615 | 5.96 |
| Proposed | 2,542 | 3.21 |

delivered impressive results in detecting unknown attack types in the assumed scenarios. It is expected that our model will not only perform multi-class classification but also significantly reduce the effort required for data labeling. The proposed system achieved stable performance for all attack types on the CHD. For the ROAD, our system achieved results comparable to those of the MTH system by using fewer labeled data.

The proposed system was designed as a lightweight model comprising a straightforward encoder and simple classifier. Therefore, we optimized both the training and operational costs while simultaneously enhancing the performance by addressing data-related challenges to fulfill the demands of a real IDS in an IVN. In the future, we expect to propose a solution to combine the encoder model with a multi-agent reinforcement learning framework and to optimize the data labeling costs through few-shot learning algorithms. Furthermore, we plan to extend the experiments of the proposed system on a variety of attacks, such as suspension and masquerade attacks, for comprehensive evaluations. To investigate its applicability for general Internet anomaly detection would be our important future work by exploring its scope and strengths thoroughly.

## CRediT authorship contribution statement

**Trieu-Phong Nguyen:** Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Jeongho Cho:** Writing – review & editing, Validation, Formal analysis, Data curation. **Daehee Kim:** Writing – review & editing, Supervision, Methodology, Investigation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgments

This research was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE) (2021RIS-004), supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-01197, Convergence security core talent training business (SoonChunHyang University)), and supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00277255). This work was also supported by the Soonchunhyang Research Fund.

## References

[1] K. Wang, A. Zhang, H. Sun, B. Wang, Analysis of recent deep-learning-based intrusion detection methods for in-vehicle network, IEEE Transact. Intell. Transport. Syst. 24 (2) (2022) 1843–1854.
[2] P. Agbaje, A. Anjum, A. Mitra, E. Oseghale, G. Bloom, H. Olufowobi, Survey of interoperability challenges in the internet of vehicles, IEEE Transact. Intell. Transport. Syst. 23 (12) (2022) 22838–22861.
[3] P.F. de Araujo-Filho, M. Naili, G. Kaddoum, E.T. Fapi, Z. Zhu, Unsupervised GAN-based intrusion detection system using temporal convolutional networks and self-attention, IEEE Transact. Netw. Serv. Manag. (2023).
[4] S. Anbalagan, G. Raja, S. Gurumoorthy, R.D. Suresh, K. Dev, IIDS: Intelligent intrusion detection system for sustainable development in autonomous vehicles, IEEE Transact. Intell. Transport. Syst. (2023).
[5] B. Lampe, W. Meng, A survey of deep learning-based intrusion detection in automotive applications, Expert Syst. Applic. (2023) 119771.
[6] H.M. Song, H.K. Kim, Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data, IEEE Transact. Vehicul. Technol. 70 (2) (2021) 1098–1108.
[7] S. Sharma, B. Kaushik, A survey on internet of vehicles: applications, security issues & solutions, Vehicul. Commun. 20 (2019) 100182.
[8] F. Jin, M. Chen, W. Zhang, Y. Yuan, S. Wang, Intrusion detection on internet of vehicles via combining log-ratio oversampling, outlier detection and metric learning, Inform. Sci. 579 (2021) 814–831.
[9] H.M. Song, J. Woo, H.K. Kim, In-vehicle network intrusion detection using deep convolutional neural network, Vehicul. Commun. 21 (2020) 100198.
[10] M.D. Hossain, H. Inoue, H. Ochiai, D. Fall, Y. Kadobayashi, LSTM-based intrusion detection system for in-vehicle can bus communications, IEEE Access 8 (2020) 185489–185502.
[11] W. Lo, H. Alqahtani, K. Thakur, A. Almadhor, S. Chander, G. Kumar, A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic, Vehicul. Commun. 35 (2022) 100471.
[12] C. Liu, R. Antypenko, I. Sushko, O. Zakharchenko, Intrusion detection system after data augmentation schemes based on the VAE and CVAE, IEEE Transact. Reliab. 71 (2) (2022) 1000–1010.
[13] O. Loyola-González, J.F. Martínez-Trinidad, J.A. Carrasco-Ochoa, M. García-Borroto, Study of the impact of resampling methods for contrast pattern based classifiers in imbalanced databases, Neurocomputing 175 (2016) 935–947.
[14] G. Caminero, M. Lopez-Martin, B. Carro, Adversarial environment reinforcement learning algorithm for intrusion detection, Comput. Netw. 159 (2019) 96–109.
[15] T.P. Nguyen, H. Nam, D. Kim, Transformer-Based Attention Network for in-Vehicle Intrusion Detection, IEEE Access, 2023.
[16] H.J. Jo, W. Choi, A survey of attacks on controller area networks and corresponding countermeasures, IEEE Transact. Intell. Transport. Syst. 23 (7) (2021) 6123–6141.
[17] P. Cheng, M. Han, G. Liu, DESC-IDS: Towards an efficient real-time automotive intrusion detection system based on deep evolving stream clustering, Future Generat. Comput. Syst. 140 (2023) 266–281.
[18] H. Olufowobi, C. Young, J. Zambreno, G. Bloom, Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing, IEEE Transact. Vehicul. Technol. 69 (2) (2019) 1484–1494.
[19] M.L. Han, B.I. Kwak, H.K. Kim, TOW-IDS: Intrusion detection system based on three overlapped wavelets for automotive ethernet, IEEE Transact. Inform. Forens. Secur. 18 (2022) 411–422.
[20] L. Zhang, X. Yan, D. Ma, A binarized neural network approach to accelerate in-vehicle network intrusion detection, IEEE Access 10 (2022) 123505–123520.
[21] C. Dong, H. Wu, Q. Li, Multiple Observation HMM-Based CAN Bus Intrusion Detection System for in-Vehicle Network, IEEE Access, 2023.
[22] S. Rajapaksha, H. Kalutarage, M.O. Al-Kadri, G. Madzudzo, A.V. Petrovski, Keep the moving vehicle secure: context-aware intrusion detection system for in-vehicle CAN bus security, in: 2022 14th International Conference on Cyber Conflict: Keep Moving!(CyCon) 700, IEEE, 2022, pp. 309–330.
[23] S. Rajapaksha, H. Kalutarage, M.O. Al-Kadri, A. Petrovski, G. Madzudzo, Beyond vanilla: Improved autoencoder-based ensemble in-vehicle intrusion detection system, J. Inform. Secur. Applic. 77 (2023) 103570.

[24] H.J. Jo, J.H. Kim, H.Y. Choi, W. Choi, D.H. Lee, I. Lee, Mauth-can: masquerade-attack-proof authentication for in-vehicle networks, IEEE Transact. Vehicul. Technol. 69 (2) (2019) 2204–2218.
[25] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, K. Li, A survey of intrusion detection for in-vehicle networks, IEEE Transact. Intell. Transport. Syst. 21 (3) (2019) 919–933.
[26] R. Islam, M.K. Devnath, M.D. Samad, S.M.J. Al Kadry, GGNB: graph-based Gaussian naive Bayes intrusion detection system for CAN bus, Vehicul. Commun. 33 (2022) 100442.
[27] H. Lee, S.H. Jeong, H.K. Kim, OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame, in: 2017 15th Annual Conference on Privacy, Security and Trust (PST), IEEE, 2017, pp. 57–5709.
[28] H.M. Song, J. Woo, H.K. Kim, In-vehicle network intrusion detection using deep convolutional neural network, Vehicul. Commun. 21 (2020) 100198.
[29] H. Sun, M. Chen, J. Weng, Z. Liu, G. Geng, Anomaly detection for in-vehicle network using CNN-LSTM with attention mechanism, IEEE Transact. Vehicul. Technol. 70 (10) (2021) 10880–10893.
[30] T.N. Hoang, D. Kim, Supervised contrastive ResNet and transfer learning for the in-vehicle intrusion detection system, Expert Syst. Applic. 238 (2024) 122181.
[31] A.R. Javed, S. Ur Rehman, M.U. Khan, M. Alazab, T. Reddy, CANintelliIDS: detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU, IEEE Transact. Netw. Sci. Eng. 8 (2) (2021) 1456–1466.
[32] S. Dong, Y. Xia, T. Peng, Network abnormal traffic detection model based on semi-supervised deep reinforcement learning, IEEE Transact. Netw. Serv. Manag. 18 (4) (2021) 4197–4212.
[33] T.N. Hoang, D. Kim, Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders, Vehicul. Commun. 38 (2022) 100520.
[34] A. Mammeri, Y. Zhao, A. Boukerche, A.J. Siddiqui, B. Pekilis, Design of a semi-supervised learning strategy based on convolutional neural network for vehicle maneuver classification, in: 2019 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), IEEE, 2019, pp. 65–70.
[35] Y. Bao, S. Yang, Two novel SMOTE methods for solving imbalanced classification problems, IEEE Access 11 (2023) 5816–5823.
[36] L. Yang, A. Moubayed, I. Hamieh, A. Shami, Tree-based intelligent intrusion detection system in internet of vehicles, in: 2019 IEEE global communications conference (GLOBECOM), IEEE, 2019, pp. 1–6.
[37] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artifi. Intell. Res. 16 (2002) 321–357.
[38] H. Zhang, J. Li, A new network intrusion detection based on semi-supervised dimensionality reduction and tri-LightGBM, in: 2020 International Conference on Pervasive Artificial Intelligence (ICPAI), December, IEEE, 2020, pp. 35–40.
[39] S. Harini, K. Nivedha, S.K. BG, R. Gokul, B.S. Jayasree, Data anomaly detection in wireless sensor networks using β-variational autoencoder, in: 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS), February, IEEE, 2023, pp. 631–636.
[40] E. Seo, H.M. Song, H.K. Kim, GIDS: GAN based intrusion detection system for in-vehicle network, in: 2018 16th Annual Conference on Privacy, Security and Trust (PST), August, IEEE, 2018, pp. 1–6.
[41] D.P. Kingma, M. Welling, An introduction to variational autoencoders, Found. Trend. Mach. Learn. 12 (4) (2019) 307–392.
[42] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 1st ed., A Bradford Book, March 1, 1998.
[43] Optimizing for Tensor Cores. https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html#opt-tensor-cores.
[44] Verma, M.E., Iannacone, M.D., Bridges, R.A., Hollifield, S.C., Moriano, P., Kay, B., & Combs, F.L. (2020). Addressing the lack of comparability & testing in CAN intrusion detection research: a comprehensive guide to CAN IDS data & introduction of the ROAD dataset. arXiv preprint arXiv:2012.14600.
[45] S. Li, Y. Cao, H.J. Hadi, F. Hao, F.B. Hussain, L. Chen, ECF-IDS: an enhanced cuckoo filter-based intrusion detection system for in-vehicle network, IEEE Transact. Netw. Serv. Manag. (2024).
[46] P. Cheng, L. Hua, H. Jiang, G. Liu, LSF-IDM: deep learning-based lightweight semantic fusion intrusion detection model for automotive, Peer-to-Peer Network. Applic. (2024) 1–22.
[47] L. Yang, A. Moubayed, A. Shami, MTH-IDS: a multitiered hybrid intrusion detection system for internet of vehicles, IEEE IoT J. 9 (1) (2021) 616–632.
[48] L. Du, Z. Gu, Y. Wang, C. Gao, Open world intrusion detection: an open set recognition method for can bus in intelligent connected vehicles, IEEE Netw. 38 (3) (2024) 76–82.