electronics

MDPI

Article

A Novel Multi-Attack IDS Framework for Intelligent Connected Terminals Based on Over-the-Air Signature Updates

Beibei Li, Wei Hu *, Xue Qu and Yiwei Li

School of Cybersecurity, Northwestern Polytechnical University, Xi'an 710060, China; beibeili@mail.nwpu.edu.cn (B.L.); quxue@mail.nwpu.edu.cn (X.Q.); lywei@mail.nwpu.edu.cn (Y.L.)
* Correspondence: weihu@nwpu.edu.cn

Abstract: Modern terminals are developing toward intelligence and ubiquitous connection. Such ICTs (intelligent connected terminals) interact more frequently with the outside world and expose new attack surfaces. IDSs (intrusion detection systems) play a vital role in protecting ICT security. Multi-attack IDSs that can cover both intra-terminal and inter-terminal networks are a promising research direction for improving detection accuracy and the strength of security protection. However, a major challenge is the frequent dynamic signature updates across the network boundary, which cause significant computational overheads and result in losses in detection performance. In light of this, we propose a novel IDS framework based on OTA (over-the-air) signature updates to implement multi-attack detection. It updates the attack signatures of the target ICTs and adds the new attack signatures to the signature database in order to minimize the local memory storage and computing resources. It employs a CNN (convolutional neural network) based on an auto-encoder to achieve multi-attack detection, which can ensure the detection accuracy of multi-attacks with the multiple classification function. We evaluated our framework on four types of real-world ICT attack data, drawing comparisons with four widely used IDS schemes, and demonstrated the non-negligible superiority of our scheme over all benchmarks in terms of accuracy, recall, precision, and F1-score. Our work represents an important step toward an IDS that can detect multi-attacks in both intra-terminal and inter-terminal networks.

Keywords: intelligent connected terminal; intrusion detection system; multi-attack; over-the-air; convolutional neural network; auto-encoder

Citation: Li, B.; Hu, W.; Qu, X.; Li Y. A Novel Multi-Attack IDS Framework for Intelligent Connected Terminals Based on Over-the-Air Signature Updates. *Electronics* **2023**, *12*, 2267. https://doi.org/10.3390/electronics12102267

Academic Editor: Cheng-Chi Lee

Received: 3 April 2023
Revised: 14 May 2023
Accepted: 15 May 2023
Published: 17 May 2023

1. Introduction

With the continuous development of computing technology and the IoT (Internet of things), ICTs (intelligent connected terminals), such as intelligent connected vehicles, intelligent connected watches, and smart homes, have become an important part of people's daily lives. Modern ICTs have strong capabilities regarding complex environment perception and intelligent control with the help of advanced computing architectures and sensors [1,2]. However, with the increasing degree of information exchange and open interface exposure between ICTs and the outside world, numerous risks of malicious attacks have emerged in both intra-terminal and inter-terminal networks. Such attacks may cause information leakage, privilege escalation, denial of service, etc. [3–5]. Taking intelligent connected vehicles as an example, Miller et al. successfully attacked target vehicles with both physical and wireless connections, taking control of the engine and steering functions of a Jeep Cherokee [6,7]. The Keen Security Lab of Tencent has successfully implemented attacks on different ICTs in recent years. They successfully launched remote attacks on a Tesla Model S/X in 2016 [8] and 2017 [9–11] and completed a remote non-contact attack on a Tesla Model S (version 2018.6.1) in 2019 [12]. In addition, they successfully attacked the Xiaomi balance car by exploiting multiple vulnerabilities and remotely reset the Bluetooth password [13]. These studies revealed that attacks on ICTs pose a significant threat to people's daily lives. Thus, research on ICT security is crucial.

In the past few years, numerous research studies have been conducted in the ICT security field, falling into three main categories: encryption and authentication [14–21], gateway isolation [22], and IDSs (intrusion detection systems) [23–38]. Specifically, the encryption and authentication approaches can accurately identify an attacker's intrusion and immediately distinguish the identity of the attacker. However, it should be noted that implementing encryption and authentication techniques comes at the cost of high computational complexity and memory consumption. Similarly, the gateway isolation methods help isolate high-risk application systems that require a higher level of security protection from non-critical systems. However, due to the complexity of the realistic terminal architecture, it is extremely difficult to deploy gateway isolation techniques. In addition, these solutions are often used to identify known attacks but cannot achieve the real-time detection of new threats. Therefore, reducing computation and memory overheads while accurately preventing potential attacks has become an important goal in order to mitigate the malicious threats to ICTs. IDSs provide a new perspective for ensuring ICT security, due to their excellent detection capabilities.

This paper first proposes a novel multi-attack IDS framework for ICTs based on over-the-air (OTA) signature updates. We injected attack datasets into the relevant components of the target ICTs and collected traffic data through the ICT interfaces. We refreshed the signature table by adopting the OTA signature update method, which minimized the local memory storage and computing resources of the ICTs. We further determined whether a single attack or multi-attack had occurred by checking the signature table and kept a connection with the ICTs by sending the new attack signatures to the signature database. We designed an auto-encoder based s CNN (convolutional neural network) to process the detection results and validate our method on public datasets. Specifically, this work makes the following main contributions:

- Proposing a novel IDS by adopting OTA signature upgrades to improve the local computational performance and attack detection accuracy;
- Developing a multi-attack detection method by adopting a CNN based on an auto-encoder through the multiple classification function in ICTs;
- Presenting experimental results to demonstrate the non-negligible superiority of the proposed method in multi-attack detection for ICTs.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 elaborates the system model and describes our proposed scheme in detail. Numerical results and performance evaluations are reported in Section 4. Finally, we conclude in Section 5.

2. Related Work

Currently, security threats faced by ICTs mainly result from their increasing interaction with the external world. In this section, we review the research efforts that have aimed to mitigate such security risks. We discuss the application of the encryption and authentication and gateway isolation techniques for ICT security in Sections 2.1 and 2.2, respectively. In Section 2.3, we summarize the current popular intrusion detection techniques for protecting ICTs from security attacks.

2.1. Encryption and Authentication Techniques for ICT Security

As the fastest growing and most widely used ICT security measure, encryption techniques play a crucial role in ensuring the security of ICTs by protecting sensitive data and communications from unauthorized access and tampering. There are different types of encryption techniques that can be used for ICT security, depending on the specific requirements of the system. Generally, encryption is used to protect the integrity, confidentiality, and availability of data and communications within ICTs. The first encryption technique used in IDSs was symmetric key encryption. This method uses a single key to encrypt and decrypt data. The sender and receiver of the data share the same key, and the encrypted data can only be decrypted using this key. This technique is often used for real-time data

transmission in IDSs, as it provides fast and efficient encryption and decryption. The second encryption technique used in IDSs is asymmetric key encryption, also known as public key encryption. This method uses two keys, a public key and a private key, to encrypt and decrypt data. The sender of the data encrypts them using the recipient's public key, which can be freely distributed. The recipient then uses their private key to decrypt the data. This technique is often used for secure communication between two parties, as it provides a secure method for exchanging keys without the risk of interception. In addition to encryption techniques, IDSs may also use other security measures, such as digital signatures and certificates to ensure the authenticity and integrity of data.

To address such security issues, Roca et al. presented a semi-centralized key management framework for intelligent connected vehicle security [21,39]. They realized decentralized and dynamic key distribution during vehicle operation and considered different aspects such as the broadcast communication of the ECU (electronic control unit), the ECU manufacturing process, and ECU authentication without certificates of external third parties. Murvay et al. discovered a hidden voltage channel that could be used to transmit additional data [19]. They tested an attack scenario by encoding additional bits into different voltage levels with the help of CAN (controller area network) transceivers, which could manipulate the injected bits without affecting the decoding of normal transactions. Furthermore, they demonstrated the attack on both low-end and high-end automotive embedded platforms and implemented an authentication mechanism on the CAN while maintaining backward compatibility. Yu et al. addressed the excessive resource usage problem by transferring security operations to high-performance devices and adopted attribute-based access control to ensure message confidentiality from both the attackers and unauthorized users [20]. They reconstructed the existing access-control-based security protection technology to address new vulnerabilities arising from the use of edge computing and attribute-based access control. In addition, Dee et al. proposed centralized and distributed protocols for key distribution and freshness synchronization [14].

However, the above encryption and authentication methods were mostly designed based on event-triggered protocols, e.g., CAN bus, which are not suitable for scalable and complex time-triggered protocols. Shreejith et al. proposed a device-level technique to authenticate ECUs by adopting an enhanced network interface and hardware security module [16]. However, it required additional computing resources and made a large number of modifications to the existing automotive system, which could be inefficient. Lin et al. proposed a solution based on a VLAN (virtual local area network) to improve ICT security [17]. They addressed both safety and security issues in Ethernet-based automotive networks, especially their interactions in relation to three problems: secret key management, frame replication and elimination, and VLAN segmentation. However, this led to reduced bandwidth and low scalability. Anguluri et al. proposed a collaborative framework to improve message response time and meet the security requirements of ICTs [18]. Unfortunately, only a small portion of messages were encrypted to ensure control performance, which made the system still vulnerable to attack.

2.2. Gateway Isolation Techniques for ICT Security

The encryption and isolation gateway (gateway for short) is a protective device in the security entry area. Its security protection strategy involves message transmission security, key management security, equipment management security, and physical protection security. Maher et al. identified a new attack vector in intelligent connected vehicles by analyzing the interface data, which could prevent malicious attackers from accessing and invading vehicles [22]. Although the proposed scheme performed well in eliminating the security risks for intelligent connected vehicles, it was ineffective in preventing attacks for intra-terminal networks. Fan et al. summarized the security defense schemes in encryption and network isolation gateways for power systems [40]. They pointed out that the master stations of power-expenditure information collection systems are vulnerable to attacks from public networks. Thus, it is necessary to enhance the security defense in the access area of

the master station to achieve network isolation, terminal identity authentication, protocol filtering, and transmission protection. These security techniques can further be used to reduce intrusion risk in ICTs. More recently, Choi et al. proposed an architecture to protect the security control rules running in a hardware-isolated space, which could be achieved by providing an isolation environment to the protocol stack and security control rules through TrustZone, even if the protocol stack was compromised [41]. In addition, the proposed architecture could protect against attack scenarios such as command manipulation, information leakage, and fuzzing attacks.

The above gateway isolation methods reduced the computing costs in ICTs by integrating intrusion prevention, encrypted channels, digital certificates, etc. However, most of them need further improvement in load balancing, redundant backup, and integrated management, due to the increasing number of external interfaces of ICTs.

2.3. Intrusion Detection Techniques for ICT Security

Both the encryption and authentication and gateway isolation methods can effectively improve the ICT security in intra-terminal networks. However, they are often limited by local computing power, bandwidth, and memory resources in realistic applications. Therefore, IDSs have become an important complement due to their high detection accuracy for ICT attacks. The recent research on IDSs can be divided into several directions: authentication, machine learning, DL (deep learning), and so on.

Liang et al. proposed an IDS based on principal component analysis [37]. Waszecki et al. proposed an on-board network monitoring system that detected attacks by monitoring the increase in the information transmission rate [23]. Meng et al. identified the difference between the sensor inputs and controller outputs by adopting a reactive runtime actuator named safety protection [24]. Dutta et al. recognized attackers with challenge-response authentication. However, this could not be used for passive safety sensors [25]. Choi et al. identified the attack source of the CAN bus by distinguishing the corresponding communication characteristics of CAN signals [26]. Kategada et al. proposed a sequence mining approach to detect low-rate injection attacks on the CAN bus in intelligent connected vehicles [27]. Taylor et al. proposed a frequency-based anomaly detection method that could send an anomaly alarm signal when the frequency exceeded the preset threshold [28]. Cho et al. successfully discovered DoS attacks in intra-vehicle networks by adopting a detection method based on the continuous error frames [29]. Suda et al. proposed an IDS based on time series by considering the periodicity of CAN bus messages, which could be combined with an RNN (recurrent neural network) [30]. Wu et al. proposed an information entropy sliding window mechanism based on fixed message numbers. It could be adopted to solve the problem of aperiodic CAN message entropy [31].

With the rapid development of DL, IDSs based on DL have been favored by numerous researchers. Kang et al. proposed an IDS based on a DNN (deep neural network) for classifying and identifying abnormal messages in ICTs [32]. Moreover, Khan et al. proposed a virtualized IDS based on short- and long-term memory neural networks [33]. CNNs are another widely used method in image recognition and network anomaly detection. For example, Lopez et al. proposed an IoT traffic classifier by combining a CNN and RNN, and it consistently achieved the best detection results compared to other DL algorithms [34]. Moreover, they further proposed a novel IDS based on a random CNN to detect the magnetoencephalography activity in the early stages of Alzheimer's disease [35]. In addition, Song et al. proposed an IDS based on the residual neural network CNN model in intelligent connected vehicles. The proposed method extracted the CAN ID features of each CAN message into a 29-bit vector, and then placed 29 of these in succession to form 29×29 grid data as the model input [36]. Although IDSs based on CNNs perform well in processing the grid data of images, they cannot achieve high-quality feature extraction in CNNs, nor can they ensure the time connection of CAN ID sequential data.

Over-the-air (OTA) is a technique that allows for remote updates to be sent and applied to software or firmware on a device. In the context of an intrusion detection system

for ICTs, OTA signature updates allow for the system to receive updated signatures to detect new and emerging threats. The specific implementation details of OTA signature updates in an intrusion detection system would involve setting up a secure and reliable communication channel between the system and a remote server or database that stores the updated signatures. The system would regularly check for new updates and apply them automatically or with user confirmation. To ensure the integrity and authenticity of the updated signatures, a secure digital signature or certificate could be used to verify the source and validity of the updates. Additionally, measures such as encryption and authentication could be taken to protect the communication channel from unauthorized access or interception. Furthermore, OTA signature updates in an intrusion detection system for ICTs allow for quick and efficient updates to be applied to the system, improving its ability to detect and prevent new and emerging threats.

In conclusion, existing works have mainly focused on anomaly detection in intra-terminal networks, while few researchers have paid attention to multi-attack detection in both intra-terminal and inter-terminal networks and across the network boundary. We made an attempt to design a multi-attack IDS that could cover both intra-terminal and inter-terminal networks for ICTs.

3. System Model

Figure 1 shows the overall process of the proposed framework. We divided the framework into three main stages, the injection of the multi-attack, signature upgrade based on OTA, and the CNN IDS based on an auto-encoder, which we will introduce in more detail below.

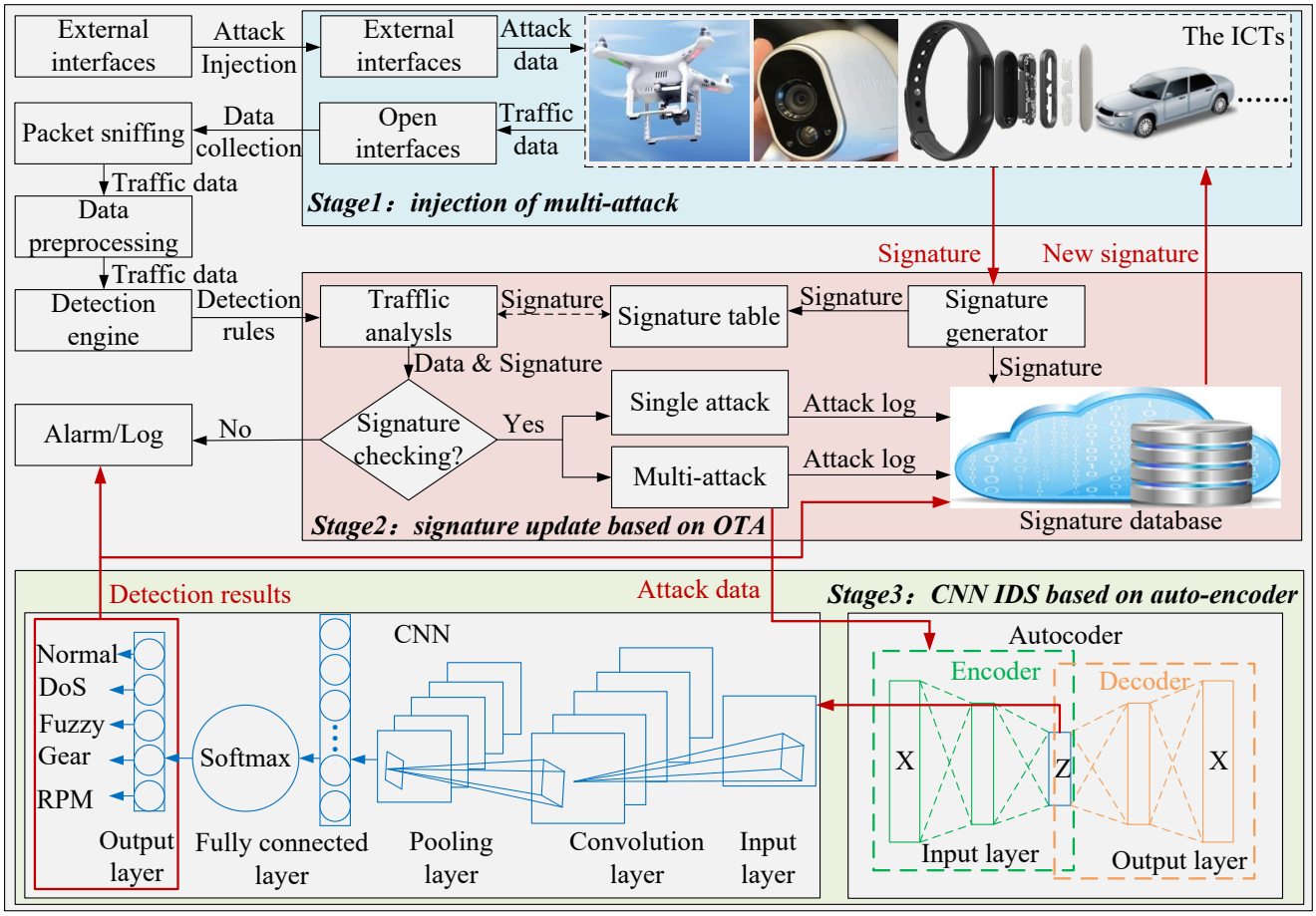

Figure 1. Overall process of the proposed framework.

3.1. Stage 1: Injection of Multi-Attacks

In this stage, we injected attack dataset into the selected components of the target ICTs and collected traffic data through the relevant ICT interfaces. We then sent the collected traffic data to the detection engine and performed data traffic analysis according to the detection rules. Below are listed three common attacks that can be executed on ICTs:

DoS (denial of service) attacks: Attackers continuously send messages with the highest priority to occupy the communication resources of ICTs, which prevents the users from sending any normal transmission messages.

Fuzzy attacks: Attackers break the ICTs by simulating legal messages and randomly injecting messages, which may cause problems such as steering wheel shaking, signal lights switching on or off irregularly, and automatic gear shifting.

Spoofing attacks (gear/RPM): Attackers inject messages with specific UINs (unique identification numbers) into the ICTs, which causes the ICTs to be abnormal. Spoofing attacks normally select the UINs on the main transmission lines to attack the target ICTs, while fuzzy attacks can create any simulated UIN to launch an attack.

3.2. Stage 2: Signature Update Based on OTA

After the injection of a multi-attack, attack data were collected through packet sniffing and data preprocessing. First, we checked the collected traffic data from stage 1 and the signature stored in the signature table and sent alerts to the users and security management system administrators when matching a specific pattern. There are three cases of detection:

Case 1: If no attack is detected, the detection engine returns the information directly to the management log.

Case 2: If a single type of attack is detected, the new attack log is directly sent to the could-based management platform.

Case 3: If a combined attack is detected, the detection engine uses the CNN classifier based on an auto-encoder to classify the traffic (see Section 3.3).

Then, the classified attack logs were sent to the cloud-based security management system, which basically contained the attack signature generator, a database, complete information about the attack signature, the signature list of each connected terminal, etc. The system sent the new attack signatures to all networked terminals. Note that the general OTA concept refers to the method of distributing new software updates, configuration settings, or encryption keys from the central unit to all connected devices or users. These devices or users can accept or reject these updates.

At present, most ICT manufacturers use OTA in vulnerability repairs and software updates [42], but as far as we know, there has been no research on the use of OTA in ICT IDSs. Thus, we regarded each terminal as a packet checker and used the packet checker to find attack signatures and update the cloud signature database. Then, we extended the packet checker to the newly connected terminal framework in order to further improve the success rate and real-time performance of attack detection.

3.3. Stage 3: CNN IDS Based on Auto-Encoder

3.3.1. Design of Auto-Encoder

Due to the limitations of the calculation and storage capacity of ICTs, reducing the number of model parameters and the computational complexity is an effective means to reduce the data dimension. An auto-encoder is an unsupervised neural network composed of an encoder and decoder, and it is a common data dimensionality reduction method. The encoder compresses the input data into a low-dimensional space, and then the decoder restores them to the original data. The encoder and decoder are connected to form a neural network, and the network parameters are optimized by the backpropagation algorithm. Therefore, we used an auto-encoder to reduce the dimensions of the data, which also effectively reduced the number of model parameters and computational complexity. Table 1 shows the detailed parameter settings of the auto-encoder. In order to preserve generality, we referred to the relevant experimental settings in [35]. We adopted the Adam optimizer

with a learning rate of 0.001 for parameter optimization. A small-batch gradient descent method with a batch size of 200 was used to calculate the error, and the model was trained with five epochs. These experimental settings are commonly used in deep learning applications and can help to achieve better results by allowing the model to converge to a more optimal solution. The Adam optimizer is known for its ability to adapt to varying learning rates and converge quickly to a good solution. The small batch size of 200 allowed for more frequent updates to the parameters of the model, which could lead to better convergence and more accurate results. Finally, training the model with only five epochs could help prevent overfitting and reduce the training time while still achieving reasonable accuracy.

Table 1. Detailed parameter settings of the auto-encoder.

Item	Value
Input	X
Number of hidden neurons	17
Optimizer	Adam
Learning rate	0.001
Error calculation algorithm	Small-batch gradient descent method
Input of error algorithm	X_{Pred}, X_{True}
Batch size	200
Number of epochs	5
Output	Z

Suppose that the encoder is expressed as $\mathbb{E}$ ($\mathbb{E} : X \to Z$) and the decoder is expressed as $\mathbb{D}$ ($\mathbb{D} : Z \to X$); the structure of the network can be expressed as:

$$\mathbb{E}, \mathbb{D} = \arg\min_{\mathbb{E},\mathbb{D}} \left\| X - \left(\mathbb{E}^0 \mathbb{D} \right) X \right\|^2 \tag{1}$$

The auto-encoder input the UIN X and output characteristic data after compression. Z is the intermediate feature data that were extracted and saved to the local file as the input of the CNN after the model training was completed. Taking the 11-bit or 29-bit CAN ID of the vehicles as an example, 9-bit data were converted into a 3×3 grid, and a mosaic pattern of 24×24 elements was formed using a data grid of 8×8 as the input of the CNN model.

We used X_{Pred} and X_{True} for error calculation, and the mean square error X_{MSE} was used as the error function. The training result could be calculated as:

$$X_{MSE} = \frac{1}{n} \sum_{i=1}^{n} (X_{\text{Pred}} - X_{\text{True}})^2. \tag{2}$$

The model adopted the Adam optimizer, and the learning rate was set to 0.001 for parameter optimization. In addition, the small-batch gradient descent method with a batch size of 200 was used to calculate the error, and the model was trained with five epochs. After the model training was completed, the intermediate feature data Z were extracted and saved to a local file as the CNN input.

3.3.2. Design of CNN

The CNN model consisted of an input layer, convolution layer, pool layer, fully connected layer, and output layer. Note that the input was the output of the auto-encoder. In the convolution layer, 20 convolution kernels were used for operation. These convolution kernels could capture different levels of abstraction and enhance the robustness of the model. A larger number of convolution kernels can increase the expressiveness of a model, but it also leads to a higher computational cost and overfitting risk. On the other hand, a smaller number of convolution kernels may not be sufficient to extract the necessary

features for accurate intrusion detection. Therefore, 20 convolution kernels are commonly used to balance model expressiveness and computational efficiency in vehicle intrusion detection research. The output size of the convolution layer was calculated as:

$$O = \frac{W - K + 2P}{S} + 1, \tag{3}$$

where O is the output size, W is the data size, K is the kernel size, P is the filling method, and S is the step size. The filling method P was used to ensure that the convoluted data size was the same as the input data size. The dimension of the pooling window f was 2×2, and average pooling was used to obtain the average value of the data in each non-repeating 2×2 area. Therefore, the dimensions of the output data after pooling were reduced to half of the original dimensions.

The input of the convolution layer depended on the output of the last layer, and its output determined the input of the next layer. When there are multiple convolution layers, the output of each convolution layer can be computed as:

$$a^i = \sigma(z) = \sigma\left(a^{i-1} * W^i + b^i\right), \tag{4}$$

where a^i is the output of a convolution layer, σ is the activation function, a^{i-1} is the output of the last layer, W^i is the connection weight, and b^i is the deviation of the layer. The initial value of offset b^i was set to 0.1.

We used 128 neurons in the fully connected layer, which was connected to the neurons of the flat final pool layer. The deviation value B in the 128 neurons was also set to the initial value of 0.1, and *tanh* was used as the activation function. In order to prevent overfitting in the process of model training, a dropout layer was adopted, which was realized by making the activation function of some neurons fail according to a certain probability. In the training process, the failure probability in the dropout layer was set to 0.5, but it was equal to 0 in the test process. Therefore, there was no dropout at all in the test phase.

Because we considered five types of data, including normal, DoS, fuzzy, gear, and RPM attacks, and our goal was to accurately classify them, the output layer also contained five neurons. A softmax classifier was used to classify these five categories, which was more suitable for multiple classification tasks than other activation functions and could be calculated as:

$$P_i = \frac{e^{o_i}}{\sum_{i=1}^{n} e^{o_i}}, \tag{5}$$

where P_i is the probability of neuron i, o_i is the output of neuron i, and n is the classification number ($n = 5$). The softmax activation function was used to calculate the probability of the input traffic belonging to different attack classes. By using softmax, the model could predict the likelihood of the input data belonging to different attack types, which could help identify and classify the intrusion in the intra-terminal network. Therefore, softmax played a critical role in enabling the multi-classification and multi-attack detection capabilities of the proposed IDS. A kind of thermal coding was adopted, and the maximum output neuron was used as the output type of the model. In the optimization of the CNN, the cross-entropy loss function was selected, and the adaptive moment estimation optimization method was used to minimize the function. The optimizer dynamically adjusted its learning rate and achieved optimal performance through continuous self-adaptation.

The data processing operation of the CNN model based on an auto-encoder was as follows:

Step 1: Converting the UIN into a two-dimensional data grid, which included an x-axis and y-axis;

Step 2: Combining multiple two-dimensional data grids to form a 2D pattern;

Step 3: Filling the grid data;

Step 4: Setting a two-dimensional data grid to form a larger two-dimensional pattern structure, in which the numbers in the grid represented binary bits;

Step 5: For this two-dimensional pattern, maintaining the temporal relationships among the features of the sequential data by arranging the data grids in sequence according to the continuous data sequence, so as to realize the transformation from the original data grid to a larger two-dimensional pattern structure;

Step 6: Using the converted data pattern structure as the training set and test set of the CNN to retain the time characteristics of the original data.

Similarly, we took the CAN ID of the vehicles as an example. The 11-bit CAN ID formed a 4 × 4 data grid, in which the 11-bit binary number was first filled on the four edges of the grid, and then the remaining five elements were filled with 0. The data filling method was based on the fact that the CNN could easily identify patterns around the grid. Then, we further set the two-dimensional data grid to form a larger 8 × 8 two-dimensional structure of data patterns, and the numbers in the grid represented binary bits. This completed the pattern conversion of the original 4 × 4 data grid into 32 × 32 elements. For the 29-bit CAN ID, we first converted each datum into a 6 × 6 data grid. At this stage, only the first 11-bit of 29-bit portions of data were non-zero, so they were placed near the center of the 6 × 6 grid to make full use of the CNN for recognizing two-dimensional patterns. Since there were 36 elements in the 6 × 6 grid, 29 of them were filled with the CAN ID, and the remaining 7 elements were filled with zeros. Similarly, after the data were gridded, they were pieced together to form a pattern structure to maintain the time characteristics of the data. This paper used an 8 × 8 data grid to form a 48 × 48 two-dimensional pattern structure as the input of the network training and testing sets.

4. Results and Performance

In this section, we first introduce the dataset used in the experiment, then provide the experimental results, and finally conduct a detailed performance analysis of the experimental results from the perspectives of computational performance and detection performance.

4.1. Test Dataset

We took the internal structure of the vehicles as an example to show the attack injection and data collection processes of ICTs, as shown in Figure 2. The model was trained and tested using the attack datasets collected by the Korean hacker and countermeasure research laboratory [43]. The datasets included the typical attack types as described in Section 3.1. These were DoS attacks, fuzzy attacks, gear attacks, and RPM attacks. In modern vehicles, the CAN bus is the most common communication mode, and hackers can easily attack vehicles due to their lack of corresponding security mechanisms. In Figure 2, taking the DoS attack as an example, we injected the attack data into the CAN bus with the CAN ID 0X000 through the external interface of the vehicle and then collected the attack data through packet sniffing, followed by the process shown in Figure 1. Other attack injection methods are similar.

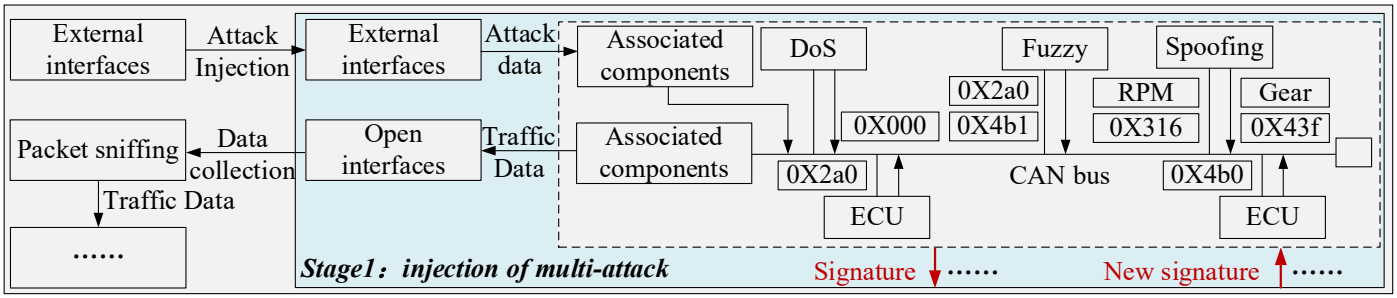

Figure 2. An example to show the attack injection and data collection processes of ICTs.

In order to evaluate the effectiveness of our method, the combination of the above four datasets was adopted to meet the requirements of multi-attack detection. The information regarding the four datasets is listed in Table 1. In the encoding process of the auto-encoder, in order to ensure the time connectivity between the CAN ID sequences, we arranged the 3×3, 4×4, or 6×6 CAN IDs into a pattern in time order. After all CAN ID data were encoded in the form of patterns, the model randomly selected a pattern in the process of training and testing, which was unaffected by the data arrangement. The training set contained 75% of the data, which were randomly selected, and the remaining 25% were used as the testing set. Although the model was trained using a combination of all datasets, all possible combinations of different types of datasets were used in the test phase. The test sets comprised four multi-attack scenarios, encompassing a total of 15 possible cases. Specifically, there were four cases comprising a single type of attack ($C_4^1 = 4$), six cases comprising two types of attacks ($C_4^2 = 6$), four cases comprising three types of attacks ($C_4^3 = 4$), and four cases comprising four types of attacks ($C_4^4 = 4$). Overall, these combinations contributed to the 15 scenarios available for analysis. Each possible case and its index number are shown in Table 2. Our model was able to detect all types of multi-attack in Table 3 by training and testing.

Table 2. Datasets of attacks. Note that the original normal message in the normal dataset was 988,872. To ensure the integrity of the normal dataset, we randomly added 115 normal messages.

Type of Attack	Total Messages	Normal Messages	Injected Messages	Percentage of Normal Messages
DoS	3,665,771	3,078,250	587,521	84.0%
Fuzzy	3,838,860	3,347,013	491,847	87.2%
Gear	4,443,142	3,845,890	597,252	86.6%
RPM	4,621,702	3,966,805	654,897	85.8%
Normal	988,987	988,987	-	100%

Table 3. All possible multi-attack scenarios.

Number	Number of Attacks	Dataset
C1	1	RPM
C2	1	Gear
C3	1	Fuzzy
C4	1	DoS
C5	2	Gear and RPM
C6	2	Fuzzy and RPM
C7	2	Fuzzy and Gear
C8	2	DoS and RPM
C9	2	DoS and Gear
C10	2	DoS and Fuzzy
C11	3	Fuzzy and Gear and RPM
C12	3	DoS and Gear and RPM
C13	3	DoS and Fuzzy and RPM
C14	3	DoS and Fuzzy and Gear
C15	4	DoS and Fuzzy and Gear and RPM

4.2. Experimental Results

We tested the proposed framework on 11-bit and 29-bit CAN IDs, and the results are shown in the confusion matrix in Figures 3 and 4. In both figures, the first row of each square is the percentage of error classification, and the second row of each square is the number of samples classified from the original category represented by the horizontal axis to the category represented by the vertical axis. For example, the intersection between fuzzy (vertical) and normal (horizontal) represents the number of normal samples incorrectly classified as fuzzy by the model.

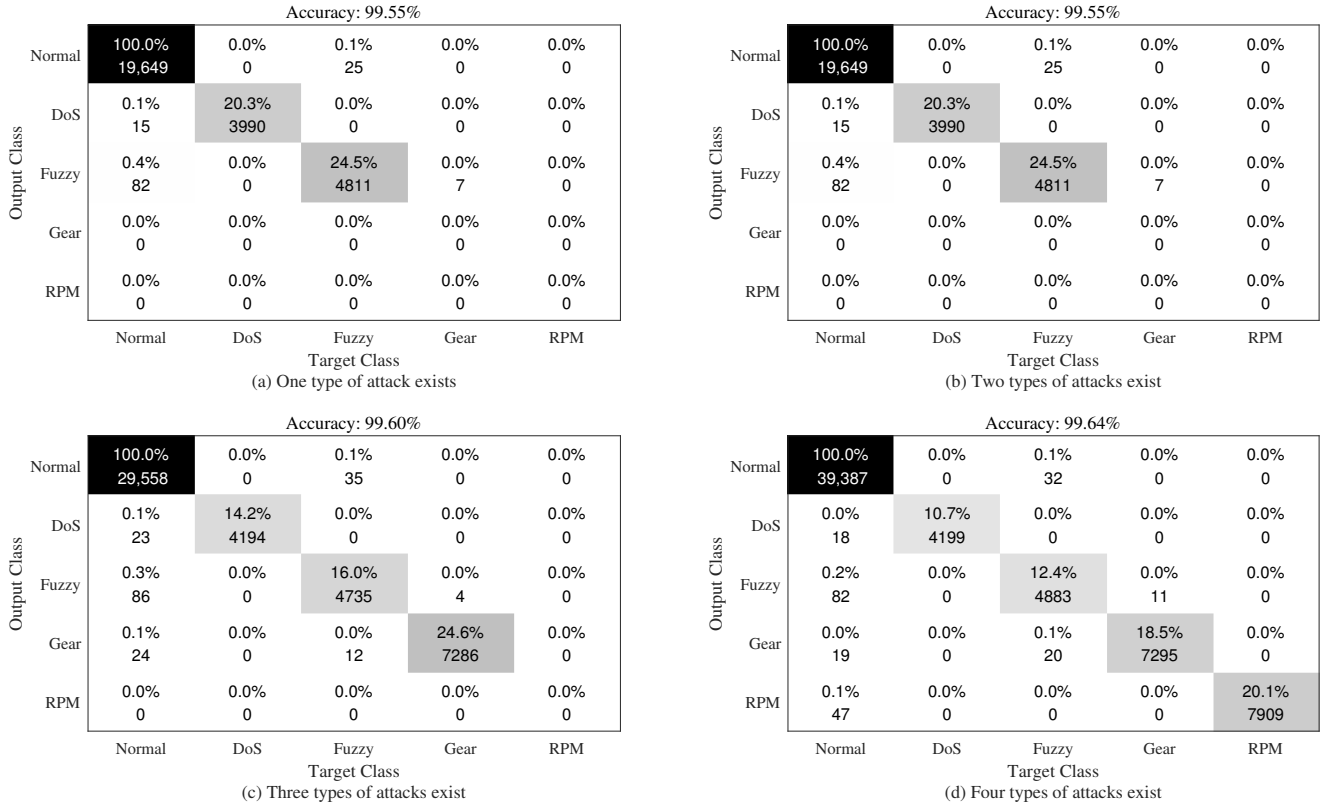

Figure 3. The typical confusion matrix of 11-bit CAN ID.

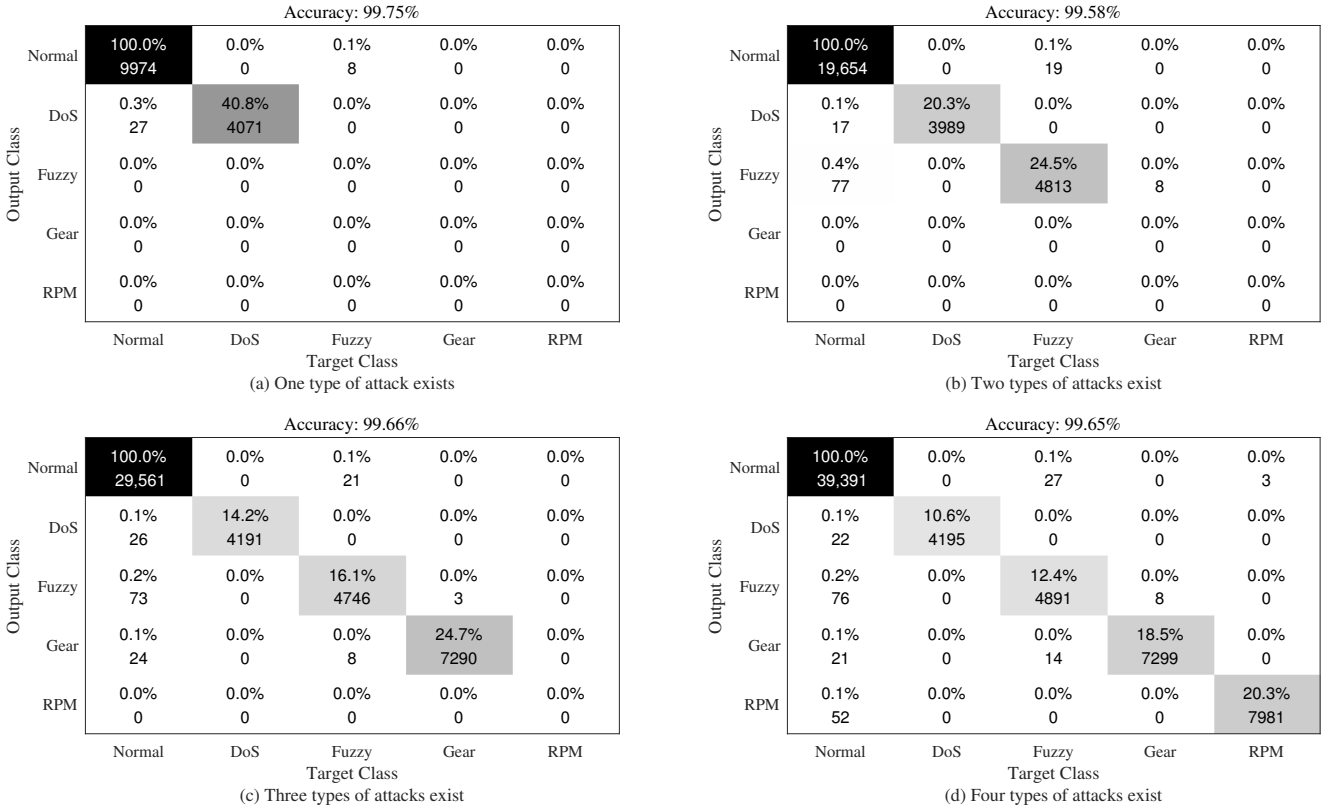

Figure 4. The typical confusion matrix of 29-bit CAN ID.

In Figures 3 and 4, the number in each cell represents the prediction of the horizontal type (prediction type) to the vertical type (real type). For example, the number in the first column of the confusion matrix represents the number of different types of RPM attacks predicted by the model, and the number in the diagonal represents the number correctly classified by the model. As shown in Figure 3, the trained model could correctly detect different attack types in most cases. However, when the dataset contained multi-attack types, one attack type was sometimes incorrectly classified as another type, and the model most likely misclassified the attack type as normal. On the other hand, when the dataset did not contain any attacks, the model may have mistakenly classified them as attacks. These false classifications slightly reduced the prediction accuracy of the model. We achieved similar results in Figure 4.

4.3. Performance Evaluations

4.3.1. Computational Performance

In the proposed IDS framework, as shown in Figure 1, by matching the patterns stored in the signature table, traffic was first filtered through signature-based detection. If a pattern matched, an alert was triggered to the driver and security management system administrator. When no attack was detected, the detection engine used the CNN classifier to classify traffic. If a new attack occurred, the attack log was sent to the cloud-based security management system, which basically comprised an attack signature generator, a database containing the attack signature information, and the signature list for each connected terminal. For new attack characteristics, the system sent the new attack signatures to all ICTs based on OTA. According to the general definition of OTA, each terminal connected to the network functioned as a packet checker to help identify attack characteristics, contribute to the cloud feature database, and then promote it to the newly connected terminals, in order to reduce the detection percentage of false positives. In addition, because both the old signature and the updated signature were stored in the cloud database, the local memory footprint and the consumption of computing resources were greatly reduced.

4.3.2. Detection Performance

In order to evaluate the prediction performance of the model more accurately, the evaluation indexes similar to the binary classification model were obtained according to the confusion matrix of each category, namely, Accuracy, Recall, Precision, and F1-score [44]. In the multiple classification model, a certain category (such as normal) was defined as a positive category, while the other categories (DoS, Fuzzy, Gear, and RPM) were defined as negative categories, and different positive and negative categories were selected in turn. Then, the true-positive (T_P) rate could be calculated for each type of attack, indicating that normal data were correctly identified; false-positive (F_P) indicates that the malicious attack data were identified as normal data, false-negative (F_N) indicates that normal data were identified as malicious attack data, and true-negative (T_N) indicates that normal data were correctly classified. The corresponding evaluation index could be calculated as:

$$\text{Accuracy} = \frac{\sum_{i \in C}\left(T_P^i + T_N^i\right)}{\sum_{i \in C}\left(T_P^i + T_N^i + F_P^i + F_N^i\right)} \tag{6}$$

$$\text{Recall} = \frac{\sum_{i \in C} T_P^i}{\sum_{i \in C}\left(T_P^i + F_N^i\right)} \tag{7}$$

$$\text{Precision} = \frac{\sum_{i \in C} T_P^i}{\sum_{i \in C}\left(T_P^i + F_P^i\right)} \tag{8}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

Equations (6)–(9) were adopted to calculate the evaluation indicators of the model in the dataset C containing different attack combinations. Figure 5 shows the model testing

results of the CNN based on an auto-encoder under different attack combinations. The abscissa numbers correspond to each combined attack dataset in Table 3.

As shown in Figure 5a, the auto-encoder performed well under different attack combinations in the 11-bit CAN ID. When the dataset contained fuzzy attacks, the performance degraded. From *C*1 to *C*3, *C*5 to *C*6, *C*9 to *C*10, and *C*12 to *C*13, the proportion of fuzzy attacks in the dataset gradually increased, and the performance of the model decreased in turn. The presence of fuzzy attacks in *C*3, *C*6, *C*10, and *C*13 indirectly reflected the significant impact of fuzzy attacks on detection accuracy. This meant that fuzzy attacks were the most challenging to detect among the test datasets used. Similarly, in Figure 5b, the auto-encoder performed well under different attack combinations in the 29-bit CAN ID. The increase in the number of bits in the CAN ID resulted in an increase in the number of non-zero elements in the grid data, which allowed for a more accurate representation of the bit encoding information and reduced data loss. As a result, the overall detection performance of the proposed framework was improved. Figure 5a shows that the inflection points of performance degradation were at *C*3, *C*6, *C*10, and *C*13, indicating that fuzzy attacks were still the most challenging to detect.

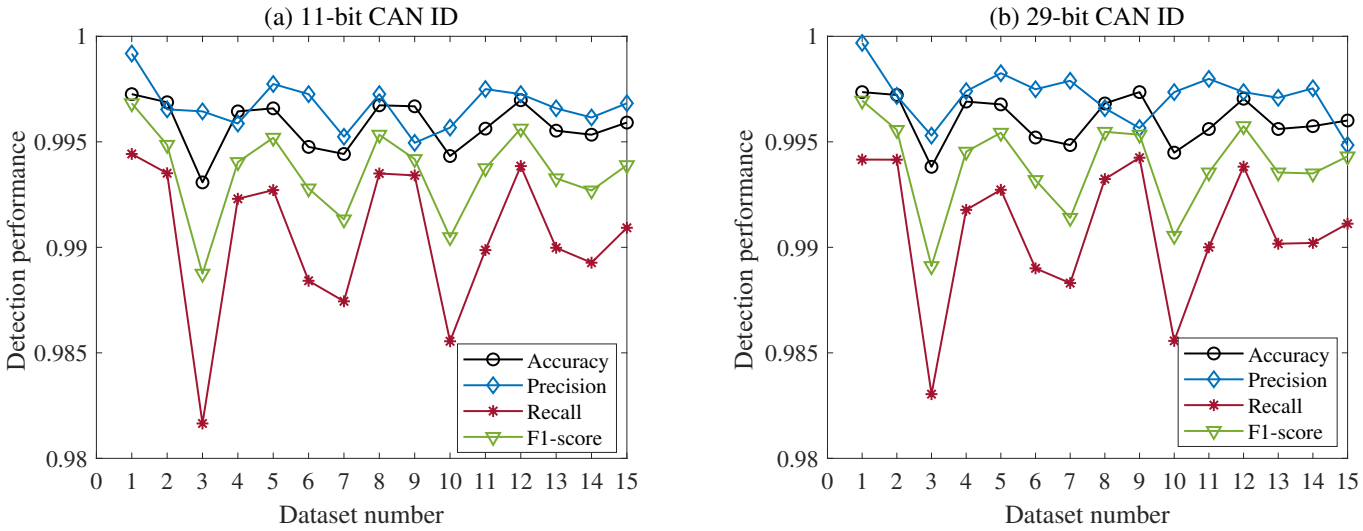

Figure 5. Results of the proposed method under different intrusion combinations. (**a**) Results for 11-bit CAN ID. (**b**) Results for 29-bit CAN ID.

After the auto-encoder training was completed, the 29-bit CAN ID was compressed into 9-bit features. Then, a 3×3 data grid was created using 9-bit data, and an 8×8 grid was used for CNN training. The trained CNN was adopted to test the data and obtain the relevant evaluation results. In order to evaluate the performance of the auto-encoder in terms of detection ability, we compared the results of the 8×8 spliced 3×3 data grid without auto-encoder feature extraction. Figure 6a,b show the comparison of the test run time and overall accuracy when adopting different combinations of the attack types listed in Table 2.

As shown in Figure 6a, for the model without the auto-encoder, the test run time was greatly affected by the number of multi-attacks. *C*15 contained four types of attacks, with the largest number of samples and the longest running time. However, *C*1, *C*2, *C*3, and *C*4 only contained one type of attack, with the lowest number of samples and the shortest running time. However, when the auto-encoder was adopted, the model operation not only greatly reduced the time, but it was also not affected by the number of samples. Because the number of connections used in the model was much smaller, the computational complexity of the model was greatly reduced, the detection time was effectively shortened with the reduction in the model parameters, and the real-time detection ability of the model for attacks was also improved. As shown in Figure 6b, the overall detection accuracy when adopting the auto-encoder also decreased slightly, although it remained at an acceptably

high level. It can be concluded that the dimensionality reduction of the auto-encoder could effectively reduce the model computational complexity without affecting the detection performance, which is of great significance for terminals with limited computing power and storage resources.

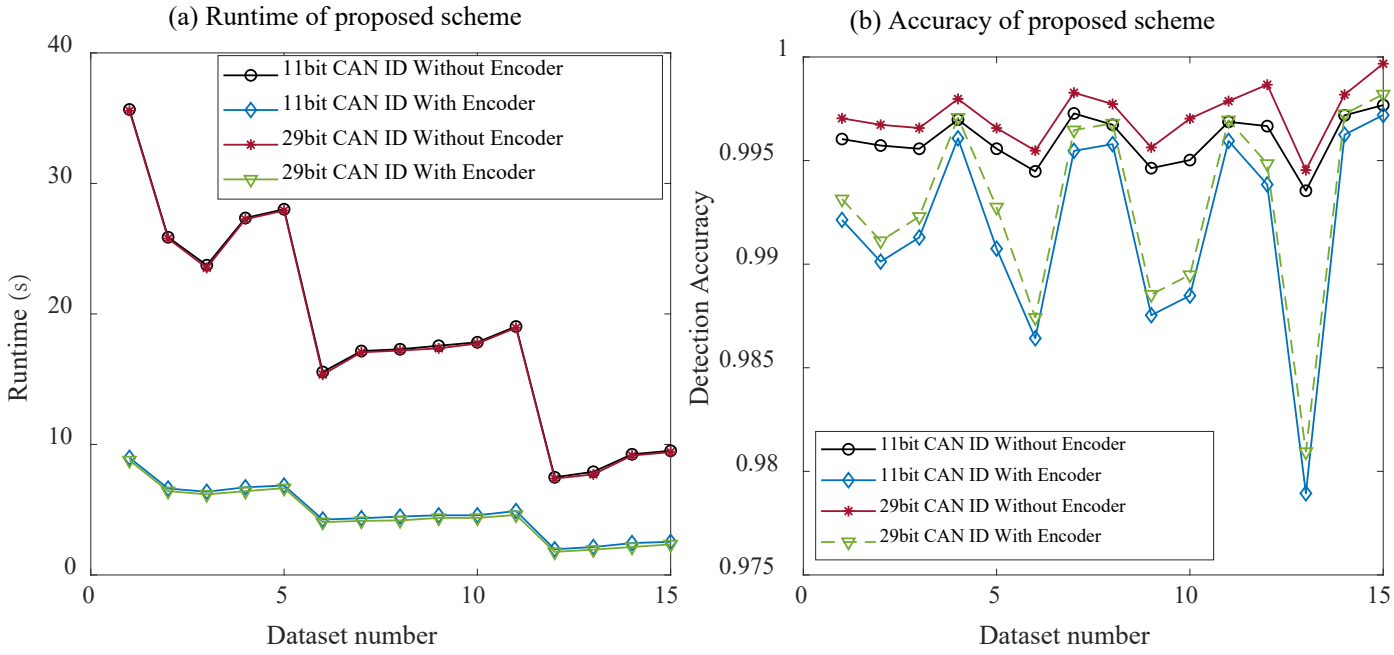

Figure 6. Comparison of the test run time and overall accuracy of the framework under different combinations of attack types with or without auto-encoder dimensionality reduction. (**a**) Time taken by the proposed framework in both 11-bit and 29-bit CAN ID. (**b**) The overall accuracy of the proposed framework in both 11-bit and 29-bit CAN ID.

Finally, in Table 4, we compared the performance of the proposed framework with that of four widely used schemes, namely TL (transfer learning), RNN, DNN, and SVM (support vector machine), on the same test datasets.

Table 4. Performance of different schemes; detection time is in seconds (s).

	11-bit CAN ID					29-bit CAN ID				
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Detection Time**	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Detection Time**
TL	94.51%	95.14%	95.46%	94.15%	7.268	93.95%	93.18%	94.16%	93.51%	7.219
RNN	89.90%	89.24%	91.18%	90.52%	13.268	10.288	92.52%	93.54%	92.03%	12.347
DNN	90.31%	92.24%	90.15%	90.18%	11.248	80.37%	82.18%	82.71%	80.15%	11.194
SVM	88.69%	91.82%	89.28%	89.15%	4.924	73.62%	77.16%	76.54%	74.16%	4.915
Our method	99.56%	98.95%	98.21%	97.75%	4.783	99.24%	99.95%	99.04%	98.84%	4.556

From Table 4, it can be seen that the proposed framework could successfully detect various types of attack data in both the 11-bit and 29-bit CAN ID, and the values of accuracy, precision, recall, and F1 score in the 11-bit CAN ID were 99.56%, 98.95%, 98.21%, and 97.75%, respectively. In the 29-bit CAN ID, the values of accuracy, precision, recall, and F1 score were 99.24%, 99.95%, 99.04%, and 98.84%, respectively. The detection results further verified our findings that the combination of OTA signature updates and an automatic encoding CNN could not only detect various types of mixed attack data but also improve the detection accuracy of intrusion detection systems.

In addition, by analyzing the attack detection time of TL, RNN, DNN, SVM, and the proposed scheme, it can be seen that the proposed scheme did not sacrifice much detection time while improving the accuracy of system detection. Furthermore, the detection

time of TL and SVM was relatively low. However, due to the impact of multi-functional classification tasks, the completion time of attack detection was slightly slower than that of the proposed scheme. Although RNN and DNN were more effective in tasks that required sequential or complex pattern recognition, the detection time was relatively high due to the high computational resources required by the model.

5. Conclusions

In this paper, we performed OTA signature upgrades in both intra-terminal and inter-terminal networks and established a CNN IDS based on an auto-encoder for multi-attack detection. We proposed an IDS by adopting a CNN to identify multi-attacks, which had a significant positive impact on improving ICT security. Extensive numerical results showed that the proposed framework not only ensured ICT security in both intra-terminal and inter-terminal networks, but could also effectively detect various attack combinations. In particular, the proposed framework could achieve a detection accuracy over 98.84%, which was higher than that of the four common ML algorithms: TL, RNN, DNN, and SVM. The results also demonstrated a high and stable multiple classification capability. Moreover, we corroborated the non-negligible superiority of our scheme over all widely used benchmarks.

However, this paper proposed an IDS framework that was validated only on the existing public datasets, instead of real-time data and real ICTs. Further research may focus on developing a practical implementation of the framework and evaluating its performance in real-world scenarios. Moreover, the proposed framework only focused on several typical attacks, and further investigations could simulate various types of attacks (e.g., DDoS attacks and malware attacks) and evaluate the performance of the framework under different conditions, such as varying traffic loads or network topologies.

Author Contributions: B.L. proposed the idea, conceived and designed the methodology and model, performed the experiments, analyzed the data, and edited and proofread the manuscript. W.H. conceived and designed the methodology and model, polished the language, and edited and proofread the manuscript. X.Q. polished the language and edited and proofread the manuscript. Y.L.: polished language and edited and proofread the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under grant 62074131, and in part by the Key R&D Program of Shaanxi Province under grant 2021GXLH-01-12.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: This work was undertaken during doctoral research at the School of Cybersecurity, Northwestern Polytechnical University, China.

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

References

1. Zhou, H.; Xu, W.; Chen, J.; Wang, W. Evolutionary V2X Technologies Toward the Internet of Vehicles: Challenges and Opportunities. *Proc. IEEE* **2020**, *108*, 308–323. [CrossRef]
2. Wang, B.; Han, Y.; Wang, S.; Tian, D.; Cai, M.; Liu, M.; Wang, L. A Review of Intelligent Connected Vehicle Cooperative Driving Development. *Mathematics* **2022**, *10*, 3635. [CrossRef]
3. Bhatia, R.; Kumar, V.; Serag, K.; Celik, Z.B.; Xu, D. Evading Voltage-Based Intrusion Detection on Automotive CAN. In Proceedings of the Network and Distributed System Security Symposium, Online, 21–25 February 2021. [CrossRef]

4. Nicolae-Gabriel, V.; Paul, P.; Mihai, D. IoT Security Challenges for Smart Homes. In Education, Research and Business Technologies: Proceedings of the 21st International Conference on Informatics in Economy (IE 2022), Bucharest, Romania, 26–27 May 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 41–49. [CrossRef]
5. Hassen, S. An Integrated Multilayered Framework for IoT Security Intrusion Decisions. *Intell. Autom. Soft Comput.* **2023**, *36*, 429–444. [CrossRef]
6. Valasek, C.; Miller, C. *Who's behind the Wheel? Exposing the Vulnerabilities and Risks of High Tech Vehicles*; IOActive: Seattle, WA, USA, 2015.
7. Miller, C.; Valasek, C. Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA* **2015**, 1–91.
8. Lab, K.S. Car Hacking Research: Remote Attack Tesla Motors by Keen Security Lab. 2016. Available online: https://www.youtube.com/watch?v=c1XyhReNcHY (accessed on 20 September 2016).
9. New China TV. See How Chinese Researchers Hacking Tesla at Hackers Conference. 2017. Available online: https://www.youtube.com/watch?v=VH4KgW-GchU (accessed on 29 July 2017).
10. Keen Security Lab of Tencent. New Car Hacking Research: 2017, Remote Attack Tesla Motors Again. 2017. Available online: https://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/ (accessed on 27 July 2017).
11. Cimpanu, C. Chinese Researchers Hack Tesla Model X in Impressive Video. 2017. Available online: https://www.bleepingcomputer.com/news/security/chinese-researchers-hack-tesla-model-x-in-impressive-video/ (accessed on 28 July 2017).
12. Nie, S.; Liu, L.; Du, Y.; Zhang, W. Over-the-Air: How We Remotely Compromised the Gateway, Bcm, and Autopilot Ecus of Tesla Cars. 2018. Available online: https://data.hackinn.com/ppt/BlackHat-USA-2018/us-18-Liu-Over-The-Air-How-We-Remotely-Compromised-The-Gateway-Bcm-And-Autopilot-Ecus-Of-Tesla-Cars-wp.pdf (accessed on 31 July 2018).
13. Lab, K.S. Contactless Attack against Xiaomi No.9 Balance Car. 2017. Available online: https://keenlab.tencent.com/zh/2017/04/01/remote-attack-on-mi-ninebot/ (accessed on 1 April 2017).
14. Dee, T.; Tyagi, A. Secure CAN for Connected Vehicles. In Proceedings of the 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2–16 June 2020. [CrossRef]
15. Lukasiewycz, M.; Mundhenk, P.; Steinhorst, S. Security-Aware Obfuscated Priority Assignment for Automotive CAN Platforms. *ACM Trans. Des. Autom. Electron. Syst.* **2016**, *21*, 1–27. [CrossRef]
16. Shreejith, S.; Fahmy, S.A. Security aware network controllers for next generation automotive embedded systems. In Proceedings of the Design Automation Conference, San Francisco, CA, USA, 8–12 June 2015. [CrossRef]
17. Lin, C.W.; Yu, H. Invited—Cooperation or competition?: Coexistence of safety and security in next-generation ethernet-based automotive networks. In Proceedings of the ACM/IEEE Design Automation Conference, Austin, TX, USA, 5–9 June 2016. [CrossRef]
18. Zheng, B.; Deng, P.; Anguluri, R.; Zhu, Q.; Pasqualetti, F. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Trans.-Comput.-Aided Des. Integr. Syst. Publ. IEEE Circuits Syst. Soc.* **2016**, *35*, 699–711. [CrossRef]
19. Murvay, P.S.; Popa, L.; Groza, B. Securing the controller area network with covert voltage channels. *Int. J. Inf. Secur.* **2021**, *20*, 817–831. [CrossRef]
20. Yu, D.; Hsu, R.H.; Lee, J.; Lee, S. EC-SVC: Secure can bus in-vehicle communications with fine-grained access control based on edge computing. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1388–1403. [CrossRef]
21. Roca, I.; Wang, J.; Du, J.; Wei, S. A Semi-centralized Security Framework for In-Vehicle Networks. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020. [CrossRef]
22. Macher, G.; Sporer, H.; Brenner, E.; Kreiner, C. Signal-Layer Security and Trust-Boundary Identification based on Hardware-Software Interface Definition. *J. Ubiquitous Syst. Pervasive Netw.* **2018**, *10*, 1–9. [CrossRef]
23. Waszecki, P.; Mundhenk, P.; Steinhorst, S.; Lukasiewycz, M.; Karri, R.; Chakraborty, S. Automotive Electrical and Electronic Architecture Security via Distributed In-Vehicle Traffic Monitoring. *IEEE Trans. Comput. Aided Des. Integr. Syst.* **2017**, *36*, 1790–1803. [CrossRef]
24. Meng, W.; Zeng, H.; Chao, W.; Yu, H. Safety Guard: Runtime Enforcement for Safety-Critical Cyber-Physical Systems: Invited. In Proceedings of the Design Automation Conference, Austin, TX, USA, 18–22 June 2017. [CrossRef]
25. Dutta, R.G.; Guo, X.; Zhang, T.; Kwiat, K.; Kamhoua, C.; Njilla, L.; Jin, Y. Estimation of Safe Sensor Measurements of Autonomous System Under Attack. In Proceedings of the Design Automation Conference, Austin, TX, USA, 18–22 June 2017; pp. 1–6. [CrossRef]
26. Choi, W.; Joo, K.; Jo, H.J.; Park, M.C.; Dong, H.L. VoltageIDS: Low-Level Communication Characteristics for Automotive Intrusion Detection System. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2114–2129. [CrossRef]
27. Katragadda, S.; Darby, P.; Roche, A.; Gottumukkala, R. Detecting Low-Rate Replay-based Injection Attacks on In-Vehicle Networks. *IEEE Access* **2020**, *8*, 54979–54993. [CrossRef]
28. Taylor, A.; Japkowicz, N.; Leblanc, S. Frequency-based anomaly detection for the automotive CAN bus. In Proceedings of the 2015 World Congress on Industrial Control Systems Security (WCICSS), London, UK, 14–16 December 2015. [CrossRef]
29. Cho, K.T.; Kang, G.S. Error Handling of In-vehicle Networks Makes Them Vulnerable. In Proceedings of the 2016 ACM SIGSAC Conference, Vienna, Austria, 24–28 October 2016. [CrossRef]

30. Suda, H.; Natsui, M.; Hanyu, T. Systematic Intrusion Detection Technique for an In-vehicle Network Based on Time-Series Feature Extraction. In Proceedings of the 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL), Linz, Austria, 16–18 May 2018; pp. 56–61. [CrossRef]
31. Wu, W. Sliding Window Optimized Information Entropy Analysis Method for Intrusion Detection on In-Vehicle Networks. *IEEE Access* **2018**, *6*, 45233–45245. [CrossRef]
32. Kang, M.J.; Kang, J.W. A Novel Intrusion Detection Method Using Deep Neural Network for In-Vehicle Network Security. In Proceedings of the 2016 IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, China, 15–18 May 2016. [CrossRef]
33. Khan, Z.; Chowdhury, M.; Islam, M.; Huang, C.Y.; Rahman, M. Long Short-Term Memory Neural Network-Based Attack Detection Model for In-Vehicle Network Security. *IEEE Sens. Lett.* **2020**, *4*, 1–4. [CrossRef]
34. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access* **2017**, *5*, 18042–18050. [CrossRef]
35. Lopez-Martin, M.; Nevado, A.; Carro, B. Detection of early stages of Alzheimer's disease based on MEG activity with a randomized convolutional neural network. *Artif. Intell. Med.* **2020**, *107*, 101924. [CrossRef] [PubMed]
36. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198.1–100198.13. [CrossRef]
37. Liang, H.; Jagielski, M.; Zheng, B.; Lin, C.W.; Kang, E.; Shiraishi, S.; Nita-Rotaru, C.; Zhu, Q. Network and system level security in connected vehicle applications. In Proceedings of the 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Diego, CA, USA, 5–8 November 2018. [CrossRef]
38. Douiba, M.; Benkirane, S.; Guezzaz, A.; Azrour, M. An improved anomaly detection model for IoT security using decision tree and gradient boosting. *J. Supercomput.* **2023**, *79*, 3392–3411. [CrossRef]
39. Saba, S.J.; Al-Nuaimi, B.T.; Suhail, R.A. A review of traditional, lightweight and ultra-lightweight cryptography techniques for IoT security environment. *AIP Conf. Proc.* **2023**, *2475*, 070001. [CrossRef]
40. Fan, L.; Du, J.; Guo, Y.; Wang, H. A Security Defense Scheme for Encryption and Network Isolation Gateway in Power System. In Proceedings of the 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 14–16 December 2018; Volume 1, pp. 1243–1246. [CrossRef]
41. Choi, H.; Kwon, H.; Lee, J.; Lee, Y.; Kim, K. Hardware-Based Isolation Technique to Guarantee Availability of Security Controls in a Gateway for Industrial Networks. In Proceedings of the 2023 International Conference on Electronics, Information, and Communication (ICEIC), Singapore, 27–28 November 2023; Volume 1, pp. 1–4. [CrossRef]
42. Otoum, Y.; Nayak, A. Signature-Over-The-Air with Transfer Learning IDS for Intelligent Connected Vehicles (ICV). In Proceedings of the 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
43. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based Intrusion Detection System for In-Vehicle Network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, Ireland, 28–30 August 2018; pp. 1–6. [CrossRef]
44. Hu, R.; Wu, Z.; Xu, Y.; Lai, T. Multi-attack and multi-classification intrusion detection for vehicle-mounted networks based on mosaic-coded convolutional neural network. *Sci. Rep.* **2022**, *12*, 6295. [CrossRef]