Hindawi
Security and Communication Networks
Volume 2022, Article ID 5827056, 11 pages
https://doi.org/10.1155/2022/5827056

Research Article

A GRU-Based Lightweight System for CAN Intrusion Detection in Real Time

Haoyu Ma, Jianqiu Cao, Bo Mi, Darong Huang, Yang Liu, and Shaoqian Li

School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China

Correspondence should be addressed to Bo Mi; mi_bo@163.com

Received 6 April 2022; Accepted 3 June 2022; Published 27 June 2022

Academic Editor: Chen Chen

With the rapid development of vehicular networking and intelligence, more interfaces are adopted by cars to interact with the external world. Accordingly, this also brings enormous security risks, which are potentially catastrophic due to communication loopholes. Since the Controller Area Network (CAN) is critical to the transmission of commands among vehicular components, it has become a prime target for hacker research and attack. Considering that the CAN bus is commonly used and its protocol is always flawed, how to efficiently detect the intrusions against it has become an evitable problem. In this paper, we presented an intrusion detection system that can be rapidly deployed inside the vehicle. Aiming at achieving the goal of real-time detection, we devised a feature extraction algorithm with low complexity and thoroughly exploited its advantages via a GRU-based lightweight neural network. The experiment was physically conducted on in-vehicle embedded devices using publicly available datasets. Experiment results illustrated that our intrusion detection system could be rapidly deployed with high classification and real-time performance. Moreover, we also discussed how an intrusion detection system could work with OTA services to improve the intelligence of vehicular operating systems and prevent potential attacks.

1. Introduction

The deep integration of high-tech network technology and automobile technology has greatly promoted the rapid development of intelligent connected vehicle technology [1]. As an important network under the vehicle network system [2], the vehicle intranet is responsible for the information interaction between the vehicle and the outside world, and the vehicle and its drivers and passengers [3]. In the vehicle intranet, CAN (Controller Area Network), as an important underlying control network of the vehicle, is mainly used to transmit the state information and control information of the vehicle, thus ensuring the smooth and safe running of the vehicle. But its communication mode is broadcast communication, with almost no encryption means. Therefore, once hackers invade cars, CAN is almost completely exposed to the outside world, which seriously affects driving safety [4]. In recent years, due to concerns about vehicle safety, some research institutions have conducted network security studies on cars using the Internet of Vehicles; for example, in September 2016, Keen Lab cracked Tesla's in-car central control to remotely control the car over long distances (up to 12 miles away) [5]. This caused the model *S* to suddenly stop while it was moving. The Keen Lab attack used a wireless network to hack into the CAN, exposing vulnerabilities in some automakers' network domain isolation.

As the case of cracking Tesla shows, CAN is directly related to the safety of cars. The safety of cars, as necessary vehicles used by people for travel, is particularly important for the whole society, transportation system, every family, and personal safety. Intrusion detection system detects network attacks by monitoring network traffic [6]. In order to improve the overall security of the vehicle by solving the security problems in the vehicle network, this paper intends to use the intrusion detection system deployed in embedded devices to detect network threats and provide early warning functions. The system can provide threat intelligence for the vehicle manufacturers and assist them to repair the vulnerabilities in the software.

In this study, we present a GRU-based lightweight system for CAN intrusion detection. The major contributions of our study are as follows:

(i) The architecture and process of the vehicle internal network intrusion detection system are proposed, which can update the detection model quickly based on the existing OTA system, improve the accuracy of the model, and reduce the failure rate.

(ii) A feature extraction algorithm with low computational complexity is proposed based on the weak computing and storage capability of vehicle-mounted computing devices.

(iii) GRU units with fewer computational parameters than LSTM are used as important hidden layers in the neural network model, and a neural network model with simple structure, low depth, and a few hidden layers is designed

(iv) In the experiment, the Jetson Xavier NX embedded computing device already mounted on the vehicle is used as the intrusion detection device to test whether the intrusion detection system has real-time performance under the condition of low computing power. And we use the evaluation indicators commonly used in classification tasks to evaluate the model classification performance, and illustrate its advantages by comparing it with those in other studies.

The rest of this paper is organized as follows. Section 2 introduces and discusses the CAN intrusion detection methods used in the existing literature; we present their shortcomings and under-researched areas of current detection models. Section 3 details our proposed intrusion detection system and discusses the low latency benefits brought by the unified memory model in embedded devices. Section 5 describes the experimental setup and experimental results. Based on the experimental results in Section 5, in Section 6 we conclude our work.

2. Related Works

In the field of CAN bus intrusion detection, researchers have focused their research on how to improve the classification performance of detection models. The existing detection models are mainly classified into statistical-based models and machine learning-based models. However, CAN bus intrusion detection systems eventually need to be deployed inside the vehicle, and previous researchers have rarely conducted simulated experiments using actual in-vehicle computing environments.

In recent years, some researchers have proposed different algorithms and models for intrusion detection system in CAN. In 2008, Larson et al. [7] proposed a CAN intrusion detection method based on vehicle communication protocol by studying CAN protocol, which could monitor the protocol-violating messages and abnormal message sending behaviors in the network. Muter [8] proposed a network anomaly detection method based on the multi-detection theory. Eight sensors were used to monitor frame ID, data load, message frequency, and message order, and finally the detection results of sensors were integrated to identify abnormal attacks. In addition, Muter introduced the entropy theory into CAN anomaly detection and calculated ID information entropy to judge the anomaly [9]. Han et al. [10] proposed a detection scheme based on ID packet cycle and designed experiments for verification. This scheme considers only the perspective of packet ID entropy or period and cannot resist tampering attacks that modify data domains.

Some researchers choose data-driven research methods, which generate datasets by collecting massive normal CAN traffic and CAN traffic generated by intrusion behaviors. CAN traffic is classified based on the machine learning model trained by the above datasets. Cheng et al. proposed TCAN-ID [11] based on TCAN (Temporal Convolutional Attention Network). This method not only includes a new feature extraction algorithm, but also uses attention mechanism [12] in neural network to improve model performance. Taylor et al. [13] proposed an anomaly detection method based on LSTM [14]. The trained LSTM model can predict the message of the next state and compare the predicted value and the actual state value with the set threshold value to judge whether the anomaly is abnormal. The experimental results show that LSTM classified the sequential CAN traffic well, but the experimental platform used by the authors is a high computing performance server, so the intrusion detection system cannot guarantee the real-time performance when it is deployed in the actual vehicle-mounted computing equipment. Seo et al. [15] used generative adversarial network learning to detect unknown attacks using only normal data, and this IDS frame is called GIDS. The GAN characteristics of GIDS eliminate the need of intrusion detection system for a large number of abnormal data samples, but the GAN model has certain difficulties in training and deployment, and the detection model itself lacks the support of real data.

In addition to the above detection methods based on deep learning model, the study [16] proposed an intrusion detection algorithm based on SVM. However, experiments show that SVM algorithm performs poorly in multi-classification tasks and is suitable for binary classification detection system. For ensemble learning, AdaBoost, as a common ensemble learning algorithm, is often used to construct a strong classifier in intrusion detection tasks. Reference [17] proposed the idea of using decision tree as weak classifier and AdaBoost iterative integration for intrusion detection, which can reduce the overall computational complexity of the system.

Through the above related research, it can be seen that the vehicle internal network intrusion detection system needs to fit the actual computing power of the vehicle. In particular, for neural network-based intrusion detection models, it is necessary to test whether the detection model runs faster than the CAN traffic generation speed.

3. Proposed CAN Intrusion Detection System

3.1. System Architecture. We need to introduce CAN bus attack model first. In this paper, the attack is defined as

connecting to CAN bus through the vulnerability between the external network and the internal network of the vehicle and injecting attack messages through the external network to achieve the attack effect. As shown in Figure 1, attacks on CAN bus topology in this paper are divided into three categories: DoS Attack, Fuzzy Attack, and Spoofing Attack.

Among them, DoS (Denial of Service) means that the hacker uses 0000 to fill CAN ID and random data payload to encapsulate CAN message for injection. According to the CAN bus protocol, the smaller the CAN ID value is, the higher the message priority is, so the bus will be disturbed by the fake message injected by the attacker and cannot send and receive normal messages.

Fuzzy Attack is similar to DoS Attack, but the CAN ID and data payload of messages injected in Fuzzy Attack are completely random, making it more hidden than DoS and thus not easy to detect by conventional anomaly detection methods. However, this type of attack will also interfere with the onboard electronic equipment connected to the bus, which will also affect vehicle safety.

The Spoofing Attack is more targeted than the previous two attacks because it uses a specified CAN ID to populate the message. This attack can be performed on a specified in-vehicle device connected to the CAN bus and may cause the device to malfunction. Since vehicle manufacturers generally do not disclose the CAN ID and data load specification of the onboard device, this attack requires special tools (e.g., social engineering). If an attacker spoofs a specific in-vehicle device such as a wheel speed sensor, this can have a significant impact on vehicle driving safety.

In Figure 2, we show the difference between an x64 [18] architecture based server and an ARM [19] architecture based embedded device. With the improvement of the requirements for intelligent vehicles, some automobile companies have installed embedded devices with neural network reasoning ability in the interior of the newly produced vehicles, and these devices all use ARM architecture as the processor architecture. If we look at these two architectures from an evolutionary perspective, computational offloading may be the solution for the future [20].

In this paper, the neural network model will be used as the CAN bus traffic classifier. We assume that the central server stores a large number of CAN data frame time series samples and the server has high computing power to train the neural network model. The built-in GPU or the neural network module embedded in the SoC (System on a Chip) has the inference ability of the neural network model that meets the real-time detection.

The types of cybersecurity threats faced by in-vehicle networks are increasing, and intrusion detection systems deployed inside vehicles will not be able to cope with new types of attacks if they cannot be updated in a timely manner. Due to the development of wireless communication technology, many vehicles are equipped with OTA (Over-the-Air) technology [21] systems based on 4G/5G transmission. Furthermore, the OTA system can allow the vehicle to download software updates from a certified remote cloud server and can send back some of the original vehicle data to the vehicle manufacturer's backend server. The intrusion detection model proposed in this paper can be delivered to each vehicle through the OTA system, and model inference is performed based on the embedded device in the vehicle. On the other hand, the model could also be transmitted more quickly and securely through a potentially widespread vehicular ad hoc network [22].

When the CAN traffic generated by the suspected intrusion behavior is automatically identified by the detection system, the intrusion detection system will upload the original suspected malicious CAN traffic log to the OTA server, and then the OTA server will send the traffic log to the log analysis center for analysis to generate intrusion judgment and report and feed back the logs determined to be intrusion behaviors after analysis to the vehicle manufacturer's vulnerability crisis management department. The vulnerability threat treatment center will analyze the log content to find out the vulnerabilities in the vehicle software to be repaired, and push the repaired system to the vehicles affected by the vulnerability through the OTA server. All the above processes are based on the good communication efficiency between the vehicle and the remote server. If the communication conditions are poor, the intelligent caching strategy [23] can also be used for data exchange.

At the same time, the CAN bus data frame sample group after log analysis will also be uploaded to the model training server through the OTA server. In the model training server, new samples will be added to the training set of existing labels or as samples of a new attack type label. After the server uses the latest training set to train the model, the deep learning model will be sent to the OTA server, and the OTA server will update the intrusion detection system deployed on the embedded device in the vehicle. Then, the updated intrusion detection system will have better classification performance. The specific data transmission schematic diagram is shown in Figure 3.

Figure 4 shows the process of the intrusion detection system proposed in this paper, and the specific implementation of each process will be described in the following sections.

3.2. Data Structure of the CAN Frame. In the CAN2.0 B technical specification [24], CAN messages are divided into four types: data frames, remote frames, error frames, and overload frames. This article will focus on the data frames used in normal communication.

In Figure 5, the number of each region represents the number of the binary digits occupied by that region. The meaning of each field in a CAN frame is described below:

(a) SOF: It is the start flag bit of the frame. On the CAN bus, a dominant bit indicates the start of a packet.

(b) CAN ID: It is the unique identifier of the ECU to which the message is passed. A smaller value in this field indicates a higher priority for the message.

(c) DLC: It represents the length of the message frame data payload.

(d) Data payload: It represents the actual data passed by the data frame.

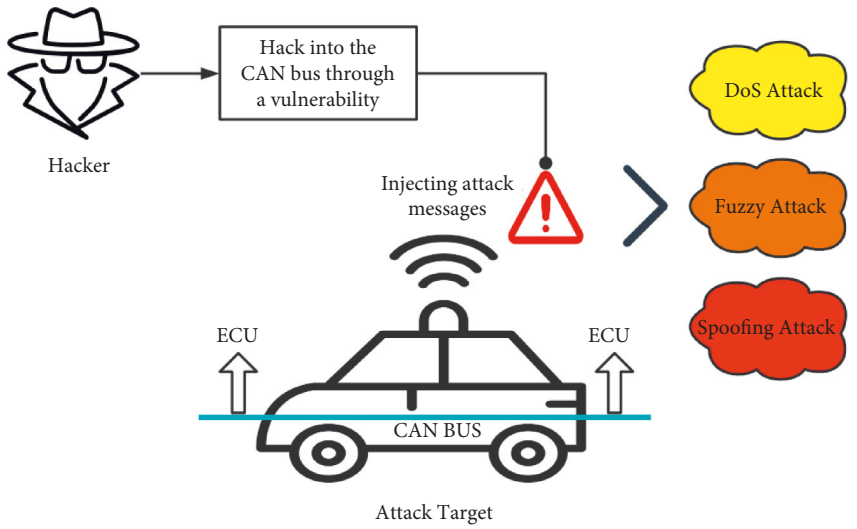


Figure 1: Schematic diagram of CAN bus attack.

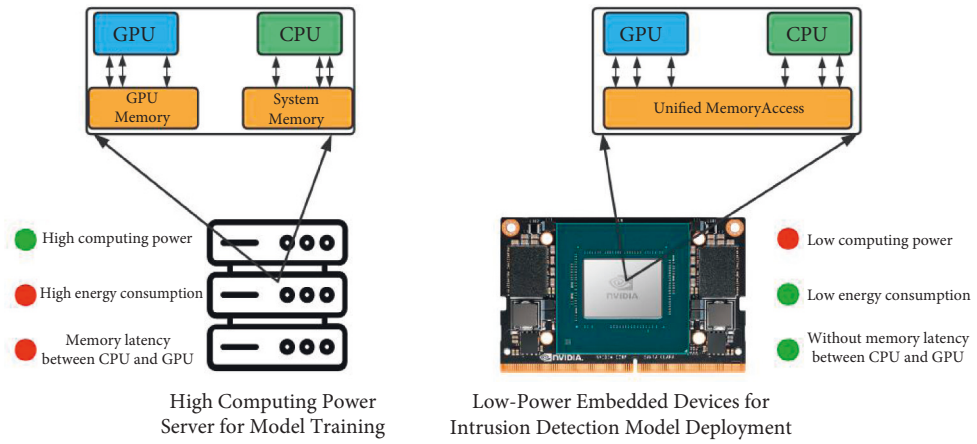


Figure 2: Intrusion detection system model training and deployment device.

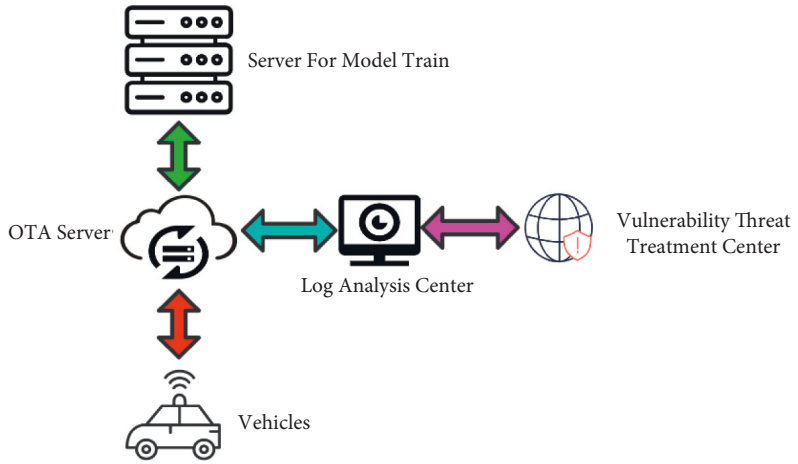


Figure 3: Scalable CAN bus intrusion detection system.

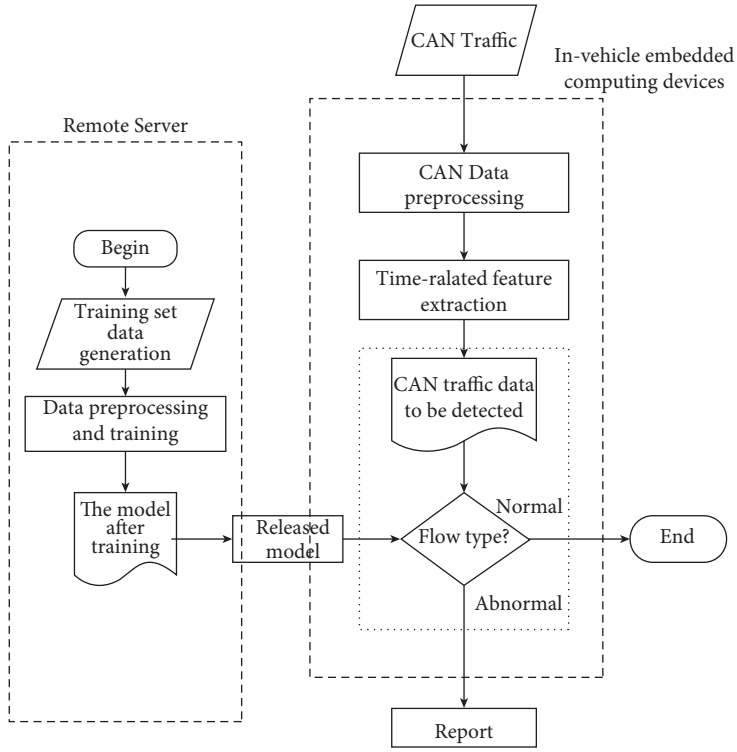


Figure 4: Intrusion detection flowchart.

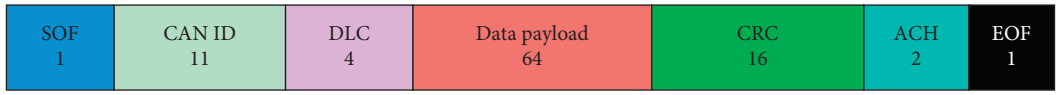


Figure 5: Structure of a CAN bus packet.

(e) CRC (Cyclic Redundancy Check): It ensures that the sender and receiver receive the correct data.

(f) ACK: It is the response field of the receiving end.

(g) EOF: It is the flag bit that indicates the end of a CAN bus frame.

In the structure of the CAN bus data packet, SOF and EOF are the flag bits of the start and end of the data frame, so they do not have the value of intrusion behavior characteristics or feature engineering. In the same way, CRC and ACK, as the fields for verifying the correctness of data transmission, can ensure that no errors will occur in the data transmission process. Since this paper only considers the intrusion detection and classification of correctly transmitted frames, this part of the data is not considered.

3.3. Data Preprocessing. In order to facilitate the subsequent feature extraction steps and deep learning model operations, we need to perform data preprocessing on the original CAN data frames. For ensuring the universality of our proposed intrusion detection algorithm, this paper uses public datasets to design data preprocessing and feature extraction for intrusion detection systems and divide the processed data into training sets and test sets. The open source in-vehicle CAN intrusion dataset used in this article is provided by a study in Korea [15]. The dataset includes the three attack types described in the previous sections of this article, and the samples in this dataset are raw data on the CAN data bus.

This public dataset includes the following attributes: TimeStamp, CAN ID, DLC, and Data Payload. The first row of the dataset is a Unix timestamp, which is the decimal number of seconds since January 1, 1970 (midnight in UTC/GMT). Since the time series features of CAN data frames are important features input to the intrusion detection model, we need to relativize the timestamps. In the process of dataset collection, all types of CAN data frames are collected continuously without interruption, so the timestamp of the first data frame can be used as the reference time, and the time difference of other data frames can be calculated by this reference. So, it is assumed that there are n samples in the dataset and the timestamp in each sample is represented as t; then, the timestamp of the $i-$ th sample is processed as follows:

$$t_i = t_i - t_0. \tag{1}$$

For CAN ID and Data Payload, the original dataset is stored in hexadecimal. In order to facilitate subsequent feature extraction and neural network training, we need to convert it to decimal. The conversion formula is as follows:

$$\text{CANID}(H) = \{H_3, H_2, H_1, H_0\}, \tag{2}$$

$$\text{CANID}(D) = H_3 * 16^3 + H_2 * 16^2 + H_1 * 16^1 + H_0 * 16^0, \tag{3}$$

$$\text{DataPayload}(H) = \{(H_1, H_0)_1, \ldots (H_1, H_0)_n\}, \tag{4}$$

$$\text{DataPayload}(D) = \{(H_1 * 16^1, H_0 * 16^0)_1, \ldots, (H_1 * 16^1, H_0 * 16^0)_n\}. \tag{5}$$

In formula (5), the value of n is determined by DLC, DLC ≤ 8.The original CAN data frame dataset collected in time series is shown in Figure 6.

3.4. Feature Extraction. As can be seen from Figure 4, the feature extraction algorithm runs on the embedded device in the vehicle, and the corresponding computing power is provided by the CPU based on the ARM architecture. Because the CPU computing power of the embedded device is weak, and the time consumed by the feature extraction algorithm will directly affect the response time of the intrusion detection system, this paper proposes a sliding window strategy based on a fixed number of messages and extracts features in the statistical domain of time series by extracting features from the CAN time series data frames passing through the window.

In the research [24], Bozdal et al. used the idea of wavelet transform to extract features from time series data frames in CAN bus, but the calculation of wavelet transform is more complicated, which may affect the real-time performance of the system. Therefore, the feature extraction method in this paper aims to reduce the computational complexity as the first purpose. First, feature extraction is performed on the preprocessed payload field. The decimal values of each byte in the field are added to obtain the sum feature of the payload field named payload_sum.If the size of the sliding window is defined as w, the feature extraction algorithm will extract the features of w packets at a time. The data payload contains n bytes in total, and each byte corresponds to a decimal number d. Therefore, the sum feature of packet m_i is calculated as follows:

$$\text{payload\_sum} = d_0 + d_1 + \cdots + d_n. \tag{6}$$

Since the messages contained in the sliding window are sorted by timestamp, in order to measure the time deviation of the samples to be detected in the window, this paper uses the variance formula to measure the deviation of the preprocessed timestamps. Defining the timestamp of a message as t, we calculate the variance as follows:

$$\text{time\_var} = \frac{(t_1 - \bar{t})^2 + (t_2 - \bar{t})^2 + \cdots + (t_w - \bar{t})^2}{w}, \tag{7}$$

$$\bar{t} = \frac{t_1 + t_2 + \ldots + t_w}{w}. \tag{8}$$

To measure the time interval between two messages with adjacent timestamps, we use the difference between two adjacent messages as the time difference feature of the message. For a message m in the window, it is necessary to calculate the timestamp difference between it and the previous message:

$$\text{time\_}d = t_m - t_{m-1}. \tag{9}$$

At the same time, in order to avoid the interference of aperiodic messages on the feature extraction algorithm, it is also necessary to calculate the median absolute error of the timestamp.

$$m(\text{time\_}d_i) = \frac{\text{time\_}d_1 + \text{time\_}d_2 + \cdots + \text{time\_}d_w}{w}, \tag{10}$$

$$\text{time\_}d\ \text{MEAD} = \text{median}(\text{time\_}d_i - m(\text{time\_}d_i)). \tag{11}$$

The intrusion detection system proposed in this paper will use the features calculated above and the CAN ID (D), Data Payload (D), and DLC of the dataset itself as input to the neural network for training and evaluation.

Time Stamp	CAN ID	N	Data Payload
1478193190.056566,	0140,	8,	00, 00, 00, 00, 10, 29, 2a, 24
1478193190.056817	02c0,	8,	15, 00, 00, 00, 00, 00, 00 ,00
1478193190.057058,	0350,	8,	05, 20, 44, 68, 77, 00, 00, 7e
1478193190.057304,	0370,	8,	00, 20, 00, 00, 00, 00, 00 ,00
1478193190.057542,	043f,	8,	00, 00, 00, 00, 00 ,00, 00, 00

Figure 6: CAN data frame with time correlation.

3.5. CAN Message Classification and Model Optimization. The detection model proposed in this paper will work on the in-vehicle embedded device, where the GPU provides the computing power for model inference and the unified memory provides the data cache space. Since the computing power and storage space provided by in-vehicle devices are limited, the design goal of our proposed neural network structure is to reduce the computational overhead as much as possible while ensuring the model detection performance.

GRU (Gate Recurrent Unit) [25] is a type of Recurrent Neural Network (RNN) [26]. Like LSTM (Long Short-Term Memory) [27], it is also proposed to solve problems such as long-term memory and gradients in backpropagation. GRU, a variant of LSTM, combines the forget gate and the input gate into a single update gate. It also mixes the cell state and the hidden state, plus some other detailed changes; the final model is simpler than the standard LSTM model. According to the experimental results of the literature [28–30], LSTM and GRU have little difference in performance on datasets with time series correlation, and GRU models even have higher performance evaluations in some datasets. From the earliest GRU literature [25], it is shown that the number of parameters calculated by GRU element is less than that by LSTM, which indicates that the time of model training and inference can be reduced. Moreover, the research [31–33] shows that GRU takes less time in training and inference than LSTM under several different types of datasets.

For the above reasons, we adopt GRU as the main hidden layer of the neural network model. The internal structure of GRU is shown in Figure 7(a). In Figure 7(a), h_{t-1} is the state

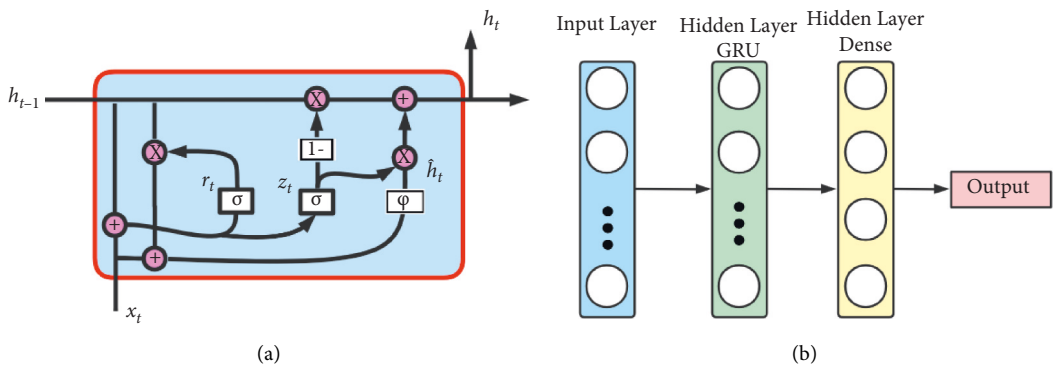


Figure 7: GRU cell structure and neural network structure.

of the previous moment relative to the current moment t. x_t and h_t are the input and output of the GRU module at the current moment, respectively. r_t and z_t are two key structures in the GRU module, namely, reset gate and update gate. Each gate is a simple neural network. And, in order to make the output of the gate fixed between 0 and 1, the activation function of the neural network is using the sigmoid function (as shown in (12). $\hat{h}_t$ is the output candidate value after reset gate processing. The structure of the GRU module is expressed by the formula as follows:

$$S(x) = \frac{1}{1+e^{-x}}, \tag{12}$$

$$r_t = \sigma\left(W_{rh}h_{t-1} + W_{rx}x_t\right), \tag{13}$$

$$z_t = \sigma\left(W_{zh}h_{t-1} + W_{zx}x_t\right), \tag{14}$$

$$\hat{h}_t = \tanh\left[W_{hh}\left(r_t \circ h_{t-1}\right) + W_{hx}x_t\right], \tag{15}$$

$$h_t = \left(1 - z_t\right) \circ \hat{h} + z_t \circ h_{t-1}. \tag{16}$$

Among them, W_{rh} and W_{rx} are the parameters in the reset gate; W_{zh} and W_{zx} are the parameters in the update gate; W_{hh} and W_{hx} are the parameters in the process of obtaining the output candidate value $\hat{h}_t$; and the operator "$\circ$" means to multiply the array elements in turn.

As shown in Figure 7(b), after the input 3-Dim data passes through the hidden layer composed of GRU units, it will be input to the fully connected layer for classification. Since the CAN intrusion detection task in this paper is a multi-classification task, the SoftMax function [34] will be used as the activation function in the last layer.

$$a_j^L = \frac{e^{z_j^L}}{\sum_k e^{z_k^L}}. \tag{17}$$

We use the common cross-entropy loss function as the basis for network backpropagation; in information theory, cross-entropy is defined as (18). When the last layer in the network uses the SoftMax function as the activation function, the output of the last layer can be regarded as a distribution, and the cross-entropy loss function used by the neural network is shown in (19):

$$j = -\int p(x)\log\, g(x)\mathrm{d}x, \tag{18}$$

$$\text{loss} = -t_j \log\left(y_i\right). \tag{19}$$

Among them, j indicates that the sample belongs to the $j-$th class and y indicates the output value of the last layer.

4. Experiment

4.1. Experimental Purpose and Configuration

4.1.1. Experimental Purpose 1. First, it is necessary to verify the effectiveness of the CAN intrusion detection process proposed in this paper by evaluating the performance of the trained intrusion detection model on the test set. And, in order to fairly evaluate the performance differences between various machine learning models, this paper selects three representative machine learning models in the field of intrusion detection and compares them with the CAN message classification model proposed in this paper. The three machine learning models are AdaBoost [17], SVM (Support-Vector Machines) [16], and LSTM. Among them, AdaBoost is an ensemble learning algorithm. In this paper, a decision tree suitable for multi-classification tasks is used as the learner integrated in it. Unlike AdaBoost, the original SVM is not suitable for the multi-classification task in this paper, so we use linear classification SVM for comparative experiments. Different from AdaBoost and SVM, LSTM has strong similarity with the GRU model used in this paper, but the calculation is more complicated.

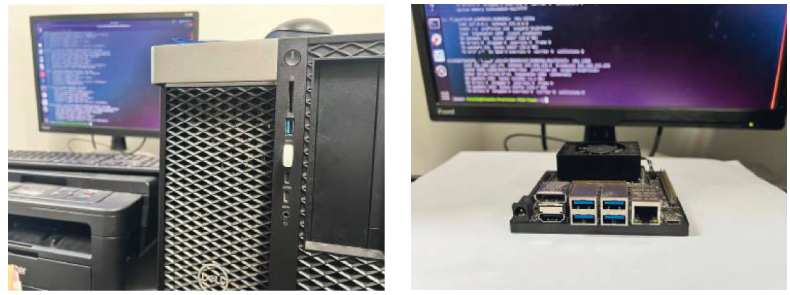
4.1.2. Experimental Purpose 2. To evaluate whether server-trained models can perform real-time inference on embedded devices, we use the Nvidia Jetson NX device used in the vehicle to test the actual deployment of the model, and the test content is the number of CAN messages detected every 100 ms. The test timing starts from the original test set data input, after data preprocessing and feature extraction; it is input to the neural network for prediction; and the timing ends after the classification result is output. In order to reflect the efficiency of the neural network model used in this article, we will use LSTM as the baseline to identify the

Table 1: Server configuration used for model training.

Category	Parameters
CPU	Intel Xeon 4210R
RAM	32 GB
GPU	Nvidia RTX 3090
Operation system	Ubuntu 18.04 LTS
CUDA version	11.1
Machine learning platform	TensorFlow 2.6 + Scikit-learn 0.23

Table 2: Embedded device configuration for model inference.

Category	Parameters
CPU	Nvidia Carmel ARM
RAM	8 GB unified memory
GPU	384-core Nvidia Volta GPU
Operation system	Ubuntu for Jetson
CUDA version	10.2
Machine learning platform	TensorFlow 2.4 + Scikit-learn 0.24



(a) (b)

Figure 8: Actual experimental device.

difference in detection speed between the two neural network models.

We use the server platform based on x64 architecture as the model training server in Figure 2. The hardware and software configuration of the platform are shown in Table 1.

To evaluate the inference performance on in-vehicle embedded devices, this paper uses an ARM-based Nvidia Jetson Xavier NX system as the experimental platform for the deployment of the intrusion detection system in Figure 2. The hardware and software configuration of the experimental platform are shown in Table 2.

Figure 8(a) shows the high computing power server used for intrusion detection model training, with average energy consumption of 800 watts. Figure 8(b) shows the experimental equipment deployed for the intrusion detection system. Under the radiator of the equipment is the Jetson Xavier NX embedded system, which has average power consumption of 10 watts.

4.2. Hyperparameter Settings. In order to ensure that subsequent researchers can reproduce the experimental results of this paper, we list the hyperparameters used for each machine learning training in the experiment. As shown in Tables 3 to 6, for a fair side-by-side comparison, we list the model hyperparameters used for the experiments.

4.3. Experimental Result. In the performance evaluation of several supervised learning models, in order to reduce the time consumption during training and ensure the fairness of the experiment, part of the original CAN bus dataset was extracted as training set and test set. The specific numbers are listed in Table 7. In addition, all models were trained and evaluated using the same dataset. In Table 8, we evaluate the effectiveness of the intrusion detection system against three attack behaviors using precision, recall, and $F1$-score. For each attack, precision is defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{20}$$

The recall rate represents how many of all attack messages were detected:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{21}$$

$F1$-score (equilibrium average) is a calculation result that comprehensively considers the precision and recall of the model, and the value is more inclined to the index with a smaller value. The larger the $F1$-score, the higher the quality of the model.

$$F1 - \text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{22}$$

Table 3: Parameters used in SVM.

Parameter name	Parameter
Penalty coefficient of error term	0.3
Kernel	SVC-linear
Probability	False
Max_iter	1000000
Random state	None

Table 4: Parameters used in AdaBoost.

Parameter name	Parameter
Base estimator	Decision tree
Number of base classifier cycles	50
Probability	False
Learning rate	1.0
Random state	None
Algorithm	SAMME.R

Table 5: Training parameters used in LSTM.

Parameter name	Parameter
Epoch	20
Num of LSTM units	40
Num of dense units	4
Batch size	64
Learning rate	0.01
Weight decay	$1e-8$
Optimizer	Adam [35]

Table 6: Training parameters used in GRU.

Parameter name	Parameter
Epoch	20
Num of LSTM units	40
Num of dense units	4
Batch size	64
Learning rate	0.01
Weight decay	$1e-8$
Optimizer	Adam [35]

Table 7: CAN intrusion dataset categories and corresponding sample numbers.

Dataset label	Num of original samples	Num of training set samples	Num of test set samples
Benign	14037000	200000	50000
DoS	587500	40000	10000
Fuzzy	491000	40000	10000
Spoofing	1134000	40000	10000

In Table 9, we present the experimental results for experimental purpose 2. A continuous period of timestamps is selected in the test set, and a total of 5000 consecutive CAN messages are available in this time block in the experiment. The real-time performance of the intrusion detection system is measured by calculating the time elapsed for message preprocessing and feature extraction as well as classification by the neural network model. After calculating the test set within forty consecutive time intervals, the CAN bus has about **200** messages or 200 CAN data frames every 100 ms. This paper uses the embedded devices in Table 2 for the experiments.

Table 8: Classification performance statistics.

DoS Attack	Precision	Recall	*F*1-Score
SVM [16]	0.8723	0.8744	0.8733
AdaBoost [17]	0.9898	0.9923	0.9910
LSTM [13]	0.9923	0.9901	0.9912
Ours	**0.9993**	**0.9991**	**0.9992**
Fuzzy Attack	Precision	Recall	*F*1-score
SVM [16]	0.8423	0.8312	0.8367
AdaBoost [17]	0.8995	0.9123	0.9059
LSTM [13]	**0.9976**	**0.9924**	**0.9950**
Ours	0.9932	0.9913	0.9922
Spoofing Attack	Precision	Recall	*F*1-score
SVM [16]	0.9312	0.9215	0.9263
AdaBoost [17]	0.9289	0.9216	0.9252
LSTM [13]	0.9932	0.9912	0.9922
Ours	**0.9995**	**0.9931**	**0.9963**

Table 9: Intrusion detection system real-time experimental results.

DL model	Num of detected messages in 100 ms	Time of detecting 5000 messages
LSTM	470	1120 ms
Ours	**650**	**890 ms**

5. Conclusion

In order to improve the security of CAN bus and avoid the attack against the important equipment on the vehicle due to the vulnerability, we study a lightweight intrusion detection system which can be deployed on the embedded equipment on the vehicle. In view of the problems of insufficient real-time performance and integration difficulties of existing intrusion detection systems, this study designs a lightweight real-time intrusion detection system suitable for vehicle-mounted CAN, which can carry out online intrusion detection of message data in the network in real time, identify intrusion messages, and secure vehicle-mounted CAN. It is proved that the intrusion detection system proposed by us has the possibility of practical deployment by conducting two contrast experiments with different purposes and setting uniform performance evaluation index.

The focus of this research is to ensure that the intrusion detection system has high sensitivity to the three attack methods in the CAN bus on the basis of simplifying the neural network structure and reducing the computational complexity. The purpose of this design is to successfully deploy the intrusion detection system into the embedded computing device on the vehicle, and we also describe the process of updating the intrusion detection model and analyzing the intrusion log.

In the future, we plan to study intrusion detection models that are simpler to compute and do not rely on GPUs, while completing the experiment on a platform with weaker computing power than Jetson. In CAN message feature extraction, we will consider using presentation learning to form end-to-end detection [36]. At the same time, considering that user privacy data on CAN bus may cause user privacy disclosure, we will consider using some privacy protection frameworks [37] to protect user privacy.

Data Availability

The CAN intrusion dataset used to support the findings of this study was supplied by Hacking and Countermeasure Research Lab under license and so cannot be made freely available. Requests for access to these data should be made to Huy Kang Kim, cenda@korea.ac.kr.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61903053; the Science and Technology Research Program of Chongqing Municipal Education Commission under Grants KJZD-K201800701, KJCX2020033, and the Opening Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security under Grant AGK2020006.

References

[1] C. Chen, L. Liu, S. Wan, X. Hui, and Q. Pei, "Data dissemination for industry 4.0 applications in internet of vehicles based on short-term traffic prediction," *ACM Transactions on Internet Technology*, vol. 22, pp. 1–18, 2022.

[2] C. Chen, Y. Zeng, H. Li, Y. Liu, and S. Wan, "A multi-hop task offloading decision model in MEC-enabled internet of vehicles," *IEEE Internet of Things Journal*, p. 1, 2022.

[3] S. Wan, S. Ding, and C. Chen, "Edge computing enabled video segmentation for real-time traffic monitoring in internet of vehicles," *Pattern Recognition*, vol. 121, Article ID 108146, 2022.

[4] C. J. Castillo, S. Zeadally and J. A. Guerrero-Ibanez, Internet of vehicles: architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[5] S. Nie, L. Liu, and Y. Du, "Free-fall: hacking tesla from wireless to can bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[6] R. A. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, pp. supl27–supl30, 2002.

[7] U. E. Larson, D. K. Nilsson, and E. Jonsson, "An approach to specification-based attack detection for in-vehicle networks," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 220–225, IEEE, Eindhoven, Netherlands, June 2008.

[8] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Proceedings of the 2010 Sixth International Conference on Information Assurance and Security*, pp. 92–98, IEEE, Atlanta, GA, USA, August 2010.

[9] M. Müter and A. Naim, "Entropy-based anomaly detection for in-vehicle networks," in *Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1110–1115, IEEE, Baden-Baden, Germany, June 2011.

[10] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular communications*, vol. 14, pp. 52–63, 2018.

[11] P. Cheng, K. Xu, S. Li, and Mu Han, "TCAN-IDS: intrusion detection system for internet of vehicle using temporal convolutional attention network," *Symmetry*, vol. 14, no. 2, p. 310, 2022.

[12] E. Choi, M. T. Bahadori, J. Sun, J. Kulas, A. Schuetz, and W. Stewart, "Retain: an interpretable predictive model for healthcare using reverse time attention mechanism," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[13] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly detection in automobile control network data with long short-term memory networks," in *Proceedings of the 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 130–139, IEEE, Montreal, QC, Canada, 2016 October.

[14] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[15] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN based intrusion detection system for in-vehicle network," in *Proceedings of the In 2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–6, IEEE, Belfast, Ireland, Aug2018.

[16] D. Jing and H.-B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," in *Proceedings of the 2019 IEEE 13th International Conference on ASIC (ASICON)*, pp. 1–4, IEEE, Chongqing, China, Nov2019.

[17] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008.

[18] M. Matsui, *How far can we go on the x64 processors."In International Workshop on Fast Software Encryption*, pp. 341–358, Springer, Berlin, Heidelberg, 2006.

[19] D.. Jaggar, "ARM architecture and systems," *IEEE micro*, vol. 17, no. 4, pp. 9–11, 1997.

[20] C. Chen, Y. Zhang, Z. Wang, S. Wan, and Q. Pei, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Applied Soft Computing*, vol. 103, Article ID 107108, 2021.

[21] S. Halder, A. Ghosal, and M. Conti, "Secure over-the-air software updates in connected vehicles: a survey," *Computer Networks*, vol. 178, Article ID 107343, 2020.

[22] B. Liu, D. Jia, J. Wang, K. Lu, and L. Wu, "Cloud-assisted safety message dissemination in VANET–cellular heterogeneous wireless network," *Ieee Systems Journal*, vol. 11, no. 1, pp. 128–139, 2017.

[23] C. Chen, J. Jiang, R. Fu, L. Chen, C. Li, and S. Wan, "An intelligent caching strategy considering time-space characteristics in vehicular named data networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021.

[24] M. Bozdal, M. Samie, and I. K. Jennions, "WINDS: a wavelet-based intrusion detection system for Controller Area Network (CAN)," *IEEE Access*, vol. 9, no. 2021, pp. 58621–58633.

[25] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1597–1600, IEEE, Boston, MA, USA, Aug2017.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *California Univ San Diego La Jolla Inst for Cognitive Science*, 1985.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] P. T. Yamak, Y. Li, and P. K. Gadosey, "A comparison between arima, lstm, and gru for time series forecasting," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, pp. 49–55, Sanya China, Dec 2019.

[29] A. Sethia and P. Raut, *Application of LSTM, GRU and ICA for stock price prediction." In Information and Communication Technology for Intelligent Systems,*pp. 479–487, Springer, Singapore, 2019.

[30] M. R. Raza, W. Hussain, and J. Maria Merigó, "Cloud sentiment accuracy comparison using RNN, LSTM and GRU," in *Proceedings of the In 2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–5, IEEE, Elazig, Turkey, October 2021.

[31] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: taking yelp review dataset as an example," in *Proceedings of the 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, pp. 98–101, IEEE, Shanghai, China, June 2020.

[32] C.-Bi Lin, Z. Dong, W.-K. Kuan, and Y.-Fa Huang, "A framework for fall detection based on OpenPose skeleton and LSTM/GRU models," *Applied Sciences*, vol. 11, no. 1, p. 329, 2020.

[33] X. Jiang, J. Sun, C. Li, and H. Ding, "Video image defogging recognition based on recurrent neural network," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3281–3288, 2018.

[34] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," *Proc. 8th Aust. Conf. on the Neural Networks*, vol. 181p. 185, Melbourne, 1997.

[35] D. P. Kingma and Ba. Jimmy, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[36] L. Wu, C. Quan, C. Li, Q. Wang, B. Zheng, and X. Luo, "A context-aware user-item representation learning for item recommendation," *ACM Transactions on Information Systems*, vol. 37, no. 2, pp. 1–29, 2019.

[37] Z. Wang, X. Pang, Y. Chen et al., "Privacy-preserving crowdsourced statistical data publishing with an untrusted server," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1356–1367, 2019.