Computers, Materials & Continua CMC, vol.65, no.1, pp.653-681, 2020

Deep Learning-Based Intrusion System for Vehicular Ad Hoc Networks

Fei Li[1, *], Jiayan Zhang[1], Edward Szczerbicki[2], Jiaqi Song [1], Ruxiang Li [1] and Renhong Diao[1]

Abstract: The increasing use of the Internet with vehicles has made travel more convenient. However, hackers can attack intelligent vehicles through various technical loopholes, resulting in a range of security issues. Due to these security issues, the safety protection technology of the in-vehicle system has become a focus of research. Using the advanced autoencoder network and recurrent neural network in deep learning, we investigated the intrusion detection system based on the in-vehicle system. We combined two algorithms to realize the efficient learning of the vehicle's boundary behavior and the detection of intrusive behavior. In order to verify the accuracy and efficiency of the proposed model, it was evaluated using real vehicle data. The experimental results show that the combination of the two technologies can effectively and accurately identify abnormal boundary behavior. The parameters of the model are self-iteratively updated using the time-based back propagation algorithm. We verified that the model proposed in this study can reach a nearly 96% accurate detection rate.

Keywords: Internet of vehicles, safety protection technology, intrusion detection system, advanced auto-encoder, recurrent neural network, time-based back propagation algorithm.

1 Introduction

In recent years, intelligent vehicles, a fusion of Internet technology and the machinery manufacturing industry, have resulted in the development of comprehensive information services for travel and daily commutes [Unluturk, Oguz and Atay (2015); Contreras-Castillo, Zeadally and Guerrero-Ibañez (2017)]. Whether intelligent network vehicles can achieve a high security and complete availability of their information is crucial for the development of intelligent vehicles. Historically, the computer systems in cars have been isolated from the outside world, and so the safety of these systems has been ignored. However, in recent years, hackers have proven that cars that utilize networked computing platforms can be compromised [Sedjelmaci, Senouci and Abu-Rgheff (2014); Li and

[1] Chengdu University of Information Technology, Chengdu, 610225, China.

[2] Gdansk University of Technology , Gdansk, 80-803, Poland.

[*] Corresponding Author: Fei Li. Email: edward.szczerbicki@zie.pg.gda.pl.

Received: 29 April 2020; Accepted: 22 May 2020.

Yang (2015); Tariq (2019)]. Compared with other network devices with a high storage capacity and computing power, the automobile is more vulnerable to network attacks due to its limited storage and computing power [Taylor, Leblanc and Japkowicz (2016)]. The frequent occurrence of automobile safety problems has made people realize the importance of automobile safety protection [Feng, He, Li et al. (2017); Miller and Valasek (2015); Checkoway, McCoy, Kantor et al. (2011)].

The earliest attacks on vehicles through the On-Board Diagnostics-II (OBD-II) port were only possible if the OBD-II connector was directly connected to a laptop, so it was difficult to carry out continuous attacks on different types of vehicles [Han, Kwak and Kim (2018)]. In recent years, with the development of computer technology, new encryption dog applications via the OBD-II connection have appeared on the market at an affordable price, and these applications can be applied to various kinds of vehicles, regardless of the car's manufacturer, model, or release year [Bernardini, Asghar and Crispo (2017)]. The malicious hackers who use these applications pose a threat to the lives of car owners, passengers, and pedestrians. Along with the development of intelligent connected vehicles, the emergence of automobile safety problems has made people realize the importance of automobile safety protection.

In order to ensure vehicle safety, a variety of network security technologies using artificial intelligence have been deployed [Samad, Alam, Mohammed et al. (2018)]. On the Internet, the first security defense system is commonly firewalls. They are deployed between the internal network and external network according to specific rules to filter the packet flow. Other types of security protection methods, such as identity authentication, Public Key Infrastructure (PKI) system, and other encryption methods, can achieve a better defense effect in a specific network environment. However, in the face of complex communication scenarios, such as automobiles or processing the data frames with unique characteristics, it is difficult for these traditional network security technologies to be able to be directly deployed on the vehicle [Woo, Jo and Lee (2014)]. As a mainstream network security technology, intrusion detection technology also faces many problems encountered by traditional methods. In recent years, many scholars have advanced a large number of real-time intrusion detection systems based on specific behavior rules or statistical models based on the users' behavior. However, it is challenging to apply these intrusion detection models to vehicles. The reasons are as follows: (1) Vehicles have limited computing resources and storage resources, and cannot complete the monitoring and analysis of activities. Therefore, the effective allocation of computing and storage resources for the vehicle intrusion detection system is a problem. (2) The topological communication environment of a vehicle is complex, and driving is affected by many subjective and objective factors, such as weather. Since the initial IDS model did not consider the data associated with these factors at the beginning of the design, many IDS models proposed in the existing research cannot be directly or well applied to cars. (3) The existing intrusion detection models always have a high false alarm rate. Due to the complexity of a vehicle's communication topology, the existing IDS model easily produces a high false alarm rate in the face of an abnormal environment. For example, if a normally driving car suddenly encounters heavy rain or large-scale traffic congestion, the packet loss rate will rise rapidly because the data on the Control Area Network (CAN) bus cannot be sent to the destination address in time. At this time, the IDS-model-

based anomaly will predict this situation as abnormal behavior, so that IDS will produce a high false alarm rate. Therefore, we must consider the following factors when designing an IDS model for automobiles: (1) The designed IDS model can achieve a lower communication load and consume less storage space when deployed on platforms with limited computing and storage resources. It can adapt to the characteristics of a strong dynamic topology and high real-time processing power in automobile communications. (2) It is necessary to consider the complex communication topology of the vehicle. For example, when the vehicle encounters non-malicious abnormal behavior, it should realize this behavior and categorize it correctly through autonomous learning. (3) For the known types of attacks, we can achieve a higher alarm rate. While in the face of non-malicious abnormal behavior, we can identify the behavior through learning independently to achieve a lower false alarm rate.

We needed to solve the problems related to the existing IDS, which only performs efficient detection for specific types of attacks [Tyagi and Dembla (2014); Sun, Yan, Zhang et al. (2015)], and improve the detection efficiency of the IDS on the vehicle system. This study presents a new IDS model based on an in-vehicle system using an advanced autoencoder and recurrent neural network in deep learning. Because of the limited computing and storage capacity of the vehicle system and the need to avoid the running burden on the vehicle system, it is not suitable to use higher dimensional data in vehicle information systems. We used an advanced autoencoder network and the corresponding sparse term constraints to reduce the dimension of the data on the CAN bus. The advanced autoencoder can learn other data representations at a high dimension by supervised learning. The data, after performing the dimension reduction using the advanced autoencoder network, contain all the information so that it can be regarded as data with the noise removed. We used the matrix operation and activation functions to complete the data recovery. The data processed by the advanced autoencoder not only do not result in the loss of useful information but also eliminates the invalid noise of the data so that the data can be sent to the classifier in a lower dimension for learning after being processed, reducing the cost of the model training inference and storage, and correspondingly improving the performance of the model [Li (2019)]. We took into consideration the poor performance of the embedded hardware in a vehicle system and maximized the use of the vehicle terminal hardware resources; we used the recurrent neural network combined with the SoftMax classifier to achieve the classification of the feature data. This method ensures a shorter processing time and maximizes the classification ability of the model without adding too much of an extra burden. The data on the CAN bus is equal time sequence data. However, the recurrent neural network has a strong ability to process sequence data and includes a simple global shared parameter mechanism [Yang, Wu, Wang et al. (2018)]. In addition, it has a weak dependence on the data context background in the training model, so the training time is shorter than that of a traditional Convolutional Neural Network (CNN) and some Recurrent Neural Network (RNN) variant networks, such as the Long Short Term Memory (LSTM) [Yuan, Zhang, Shi et al. (2019)]. The trained model achieved a higher accuracy and lower false alarm rate compared with the traditional IDS model, even in the face of non-abnormal exceptional circumstances, because the recurrent neural network can learn the sequential characteristics of the behaviors of the vehicle data effectively when training the model.

The trained IDS model can also learn pre-order features autonomously to correctly identify the data characteristics of the driving environment. To solve the problem that the recurrent neural network may encounter during the training period, it is easy to cause the gradient to disappear with the increasing depth of the model. We introduced the time-based back propagation (BPTT) algorithm to complete the training process of the whole model. In terms of the overall training efficiency, unlike the traditional CNN or RNN variants, which have more door unit and parameter training, it is difficult to adapt to the limited storage and computing resources of the automobile. The IDS model proposed in this study can complete the training of the model in a short time. The model proposed in this study can achieve a higher efficiency than the traditional intrusion detection model in terms of its detection accuracy because it considers the mechanical characteristics and timing of the vehicle.

The rest of study is organized as follows: the second part elaborates on the relevant background knowledge of the Internet of Vehicles and the related work of deep learning in IDS. The third part describes the relevant methodology of this article, and how the autoencoder and recurrent neural network are applied to the IDS of the vehicle system. In the fourth part, we use real cars to evaluate and compare the performance of our proposed models with traditional models. Finally, we summarize the problems raised and explain future directions of this work.

2 Background knowledge

Here, we introduce the basics of the Internet of Vehicles and deep learning.

2.1 The architecture of the internet of vehicles

The Internet of Vehicles refers to the use of a new generation of mobile communication technology to achieve a full-scale network connection within the vehicle, vehicle-to-person, vehicle-to-vehicle, vehicle-to-road, and vehicle-to-service platform. A new format for automotive and transportation services was built by improving the level of automotive intelligence and enhancing self-driving capabilities. It can provide users with intelligent, comfortable, safe, energy-saving, and efficient comprehensive services by improving traffic efficiency and boosting the driving experience of cars [Li, Zhong, Chen et al. (2019)]. The topological diagram of the Internet of Vehicles communication is shown in Fig. 1. The main communication entities include the Road Side Unit (RSU), the intelligent connected vehicle, the pedestrians on the road, and the official traffic authority for road communication. Each intelligent connected vehicle is composed of several devices, such as the Telematic Box (T-Box), Electronic Control Units (ECU), and GPS, which can help the vehicles to make contact with other entities in the communication scene. For example, the T-Box can complete the corresponding communication process through the built-in communication module and special automobile SIM card, combined with the Dedicated Short-Range Communications (DSRC) protocol in 802.11p or the Long Term Evolution Vehicle.

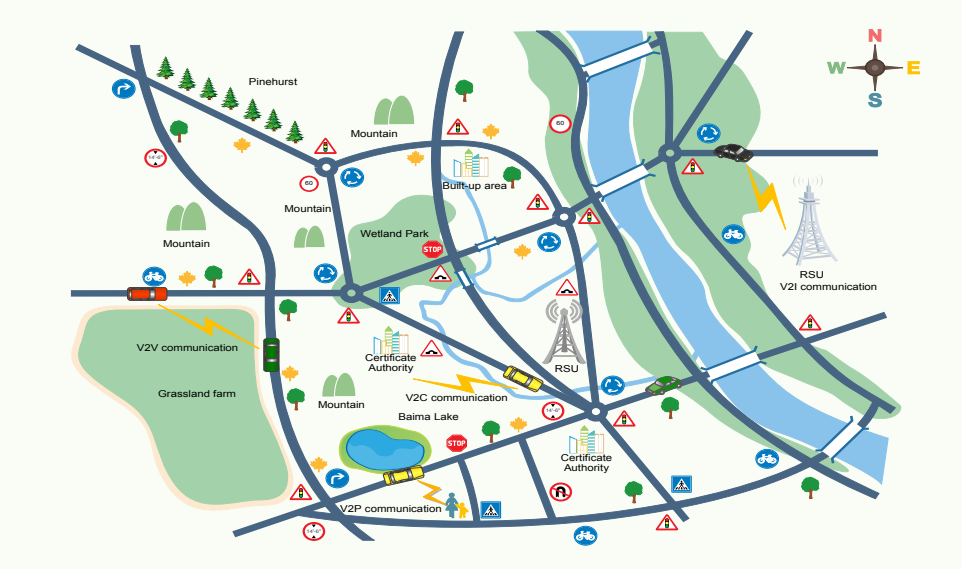

Figure 1: Network of vehicles communication scenario topology

In an intelligent connected vehicle, the bus network in the vehicle is formed using the bus communication protocol to connect the nodes of the vehicle ECU. Fig. 2 shows a simplified version of the intelligent network communication schematic diagram. ECU is not only the core component of the whole vehicle communication but also the essential communication unit of the in-vehicle communication. The node receives different message information from the bus to complete the specified command action [Li, Zhong, Chen et al. (2019)]. If different ECU nodes need to communicate with each other, they need to implement the bus protocol in the vehicle. The most famous bus protocol used in the vehicle is the CAN protocol. The CAN is a standard for the in-vehicle internal bus system, which can provide enough communication information for the ECU. The CAN bus is a reliable and economical serial bus of the vehicle network [Seo, Song and Kim (2018)]. When communicating, each ECU node sends data to the bus in a competitive way. At this time, the priority of the bus access control is obtained according to the priority domain in the data frame. ECU nodes with a low value in the arbitration field will get the priority to send the data, and the other nodes will wait for the bus to be idle and compete again. This way of broadcasting improves the real-time performance of data communications, which is why Robert Bosch GmbH6 introduced the CAN protocol in the 1980s. In modern cars, there are more than 50 ECUs, which make the communication speed between ECUs reach 1 Mbit/sec [Martinelli, Mercaldo, Orlando et al. (2018)]. However, the vehicle network uses a broadcast for the corresponding communication, and the communication process is often not authenticated, which makes it easy for attackers to access the CAN bus.

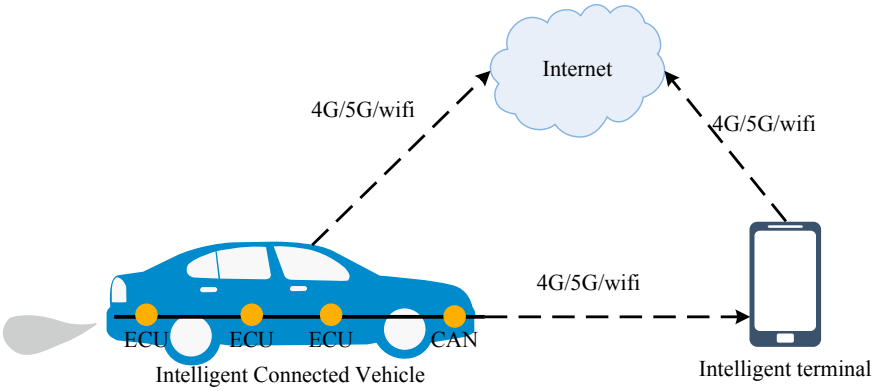

Figure 2: Simplified communication diagram of an intelligent connected vehicle

2.2 Research status of deep learning in intrusion detection

In the current application scenarios of the Internet of Vehicles, the in-vehicle IDS is usually deployed in the form of hardware or software on the vehicle for use. By completing the corresponding analysis process by collecting data from each ECU node or CAN bus, it can ensure the safety of drivers and passengers by timely notifying the emergency response background system in case of any abnormal behavior [Gao, Li, Xu et al. (2019)]. However, in the early stage of the development of the Intelligent Connected Vehicle (ICV), the ECU installed on vehicles only had a small storage space and low computing power because of the limited technology of a single-chip microcomputer. When facing a network attack, it is challenging to apply the existing IDS model directly to the vehicle system of an ICV.

Many scholars have proposed an IDS based on the in-vehicle system to address the fact that the existing IDS cannot be directly used in the vehicle system. Liu et al. [Liu, Li and Man (2015)] presented an anomaly-based intrusion detection model. In this model, the network layer and MAC address layer are detected to analyze the normal behavior characteristics of the mobile nodes, and the outlines are detected by data mining. In order to verify their work, used NS2 to verify this model. The proposed method can achieve anomaly detection with a high efficiency. However, with an increasing number of detection nodes, the overall detection efficiency of the model decreases. Besson et al. [Besson and Leleu (2016)] developed a distributed vehicle intrusion detection model based on AWISSENET. This model searched the trusted services and related paths. The proposed IDS model was tested on a heterogeneous test platform, and the experiment showed that the proposed model can be applied to different wireless networks. Lauf et al. [Lauf, Peters and Robinson (2010)] presented an anomaly intrusion detection model based on Vehicle Ad Hoc Networks (VANET). In this model, the contextual background of the interactive nodes on the network application layer is detected to enable the learning of the attack behaviors' characteristics. The model also uses the density function and the global behavior maximum function to realize the detection of abnormal behavior. The model was experimentally verified to have a relative improvement in the optimization of computation compared with the anomaly intrusion detection model using only a traditional context-based approach. However, when presented with some unusual traffic scenarios, the model still has a high

false alarm rate. A distributed IDS for wireless sensor networks based on reputation detection was proposed by Banković et al. [Banković, Moya, Araujo et al. (2010)]. In this model, each node in the network is assigned a reputation by the trained model to realize the evaluation and detection of malicious nodes. The corresponding experiments showed that the proposed intrusion detection system separated the malicious nodes from the network and inhibited the spread of malicious activities. However, the model still has a high alert rate in detecting node malicious refreshing node reputation. Cho et al. [Cho, Hong, Lee et al. (2013)] developed a local IDS model for wireless sensor networks. The model reduces the dimension of the data coding by introducing the Bloom filter proposed by Howard Bloom to reduce the storage and calculation requirements of the model. Experimental results show that the proposed method can detect potential DoS attacks by simulating the corresponding wireless communication environment.

Many scholars have found that using cryptography or the network layer related protocols to implement IDS for malicious behavior detection often emphasizes feature engineering and feature selection, which cannot effectively solve the problem of the massive intrusion data classification in the actual network application environment [Yin, Zhu, Fei et al. (2017)]. It is difficult to achieve the self-directed learning of attack features to identify the corresponding attacks in the face of a changeable attack strategy [Mershad and Artail (2012); Dong and Wang (2016); Yin, Zhu, Fei et al. (2017)]. With the successful application of machine learning and deep learning in many fields, scholars in various countries are now combining these technologies with intrusion detection systems to achieve the efficient detection of external attackers and internal malicious behaviors.

Medhat et al. [Medhat, Ramadan and Talkhan (2015)] presented an intelligent intrusion detection model based on a wireless sensor network. This model combines supervised learning with unsupervised learning to train the IDS model: supervised learning is used to train sensor nodes; unsupervised learning is used to train base station nodes and convergence nodes. During this period, a series of rules learned will form a decision binary tree to judge normal and abnormal behavior. The experimental results show that the model can achieve a lower time complexity and more accurate detection of the attack data. Ronak et al. [Ronak, Ganesh, Akshay et al. (2016)] provided a distributed intrusion detection system for wireless sensor networks based on the Naive Bayes and Apache mahout. The proposed model can detect multiple attacks autonomously and has a strong robustness. Peraković et al. [Peraković, Periša, Cvitić et al. (2017)] developed an intrusion detection model based on an artificial neural network. This model uses a supervised learning method to realize the corresponding learning of traffic labels to complete the recognition of abnormal traffic. However, this model only has obtained an 82% accuracy due to some similarities in the values of legitimate traffic and UDP DDoS attack parameters. Anzer et al. [Anzer and Elhadef (2018)] proposed an anomaly intrusion detection model base d on deep learning. The model can model the network traffic data using a fully connected neural network. Experiments show that the model proposed in this study can achieve more efficient learning of feature data compared with some traditional machine learning algorithms, such as Adaboosting, random forest, and SVM. It also has a higher accuracy and lower false alarm rate in the detection rate compared with some anomaly intrusion detection models based on traditional deep learning [Kwon, Kim, Kim et al. (2017)]. Pavani et al. [Pavani and Damodaram (2013)] proposed a neural network based on the

multi-layer perceptron to detect malicious behavior in Mobile Ad Hoc Network (MANET). Their experimental results showed that the proposed model can effectively detect the grey hole and black hole attacks. Leinmüller et al. [Leinmüller, Held, Schäfer et al. (2014)] proposed an intrusion detection system based on the Intelligent Proportional Overlapping Score (POS). Their IDS model uses the information received from the trace file of VANETs to reduce additional features to improve the performance and safety of the autonomous vehicle. The features extracted from the tracking files are helpful to distinguish normal vehicles from abnormal vehicles. The model also uses an artificial neural network to realize the learning of audit data and the detection of attack behavior. Experiments showed that their proposed model can effectively detect black hole attacks by learning the data characteristics of the trace files.

However, these technologies have many limitations, such as the need to be able to interact with a higher level of human experts, the demands for a large amount of expert knowledge in the processing of data, or the need for the differentiation of the data and operation mode to achieve accurate recognition needs [Tan, Li, Xia et al. (2019)]. They are not only a labor-intensive and expensive processes, but they are also error prone [Zhao, Yan, Chen et al. (2019)]. Using a large amount of training data results in too much system overhead, which may become a challenge to deploy in an Internet of Vehicles environment with heterogeneous characteristics and highly dynamic environment. Deep learning, as an advanced subset of machine learning, can overcome some limitations of shallow learning. Preliminary deep learning research has proved that its superior hierarchical feature learning can improve or at least match the performance of shallow learning technology [Hou, Saas, Chen et al. (2016)]. Abnormal behavior will change with time, and intruders will adjust their network attacks to avoid existing intrusion detection solutions [Kumar and Venugopalan (2017)]. Deep learning can analyze the network data at a deeper level, and identify any abnormal data and related patterns quickly. Moreover, artificial intelligence has a good black box feature. Therefore, it is difficult for attackers to manipulate the internal structure of the detection system.

Deep learning has been used for intrusion detection with the Internet of Things, especially in the Internet of Vehicles. The basic idea is to use deep learning to realize the learning of vehicle boundary behavior features, and then to design the corresponding classifier based on these boundary features. Using this classifier, we can achieve the efficient classification and anomaly detection of an entity's behavior data. Due to the limited computing and storage capacity of traditional automobile ECUs, many advanced and efficient algorithms in deep learning cannot be directly applied to the automobile [Kang and Kang (2016)]. However, the development of intelligent vehicle information systems has improved the calculation and storage capacity of the on-board ECU, and the efficiency in processing real-time tasks has also been greatly improved [Johansson, Törngren and Nielsen (2015)]. The recurrent neural network and the autoencoder based on sparse item constraints serve as a deep development of deep learning. By learning the boundary features, a multilayer recurrent neural network with learning ability was established to predict and perceive the unknown behavior. The relevant optimization algorithm was used to optimize the parameters of the model. Through experiments, we found that the model designed by deep learning has a better robustness than the traditional IDS based on statistics or signatures [Qiao, Li and Chen (2018); Mohammadi and Namadchian (2017); Liang (2017)].

3 Research on intelligent vehicular intrusion detection system based on deep learning

3.1 Structural design of intrusion detection algorithm

The detection flow chart of the in-vehicle intrusion detection model based on the advanced autoencoder and recurrent neural network is shown in Fig. 3. The proposed model consists of three steps: (1) data pre-processing, (2) the advanced autoencoder is used to extract the correlation features between the data, and (3) the combination of the recurrent neural networks and the SoftMax classifier is used to classify the corresponding data. Since the relevant experimental data collected from the vehicles were composed of continuous type data and discrete type data, we needed to normalize the data using some pre-processing methods. The specific data introduction is described in the fourth section. After the data pre-processing, the standard format data was used as the input of the sparse auto-encoder to complete the data feature extraction. We obtained data with high sparse characteristics. We used these data with great sparse features as the input of the recurrent neural network classifier, and then used the recurrent neural networks and SoftMax to learn and classify the corresponding feature data. Finally, we used the sorted results as the output to judge whether the relevant vehicle CAN bus data was abnormal.

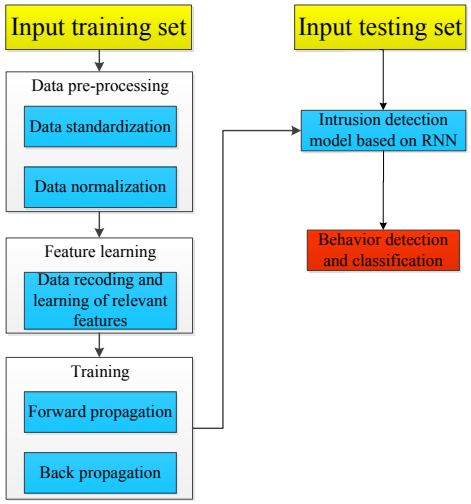

Figure 3: Flow chart of the Intrusion Detection System based on deep learning

3.2 Methodology

The autoencoder is a kind of deep learning network structure used to learn the coding structure of data. Its primary purpose is to learn high-dimensional complex data and extract a suitable coding expression mode to realize the dimension reduction processing and related feature learning of high-dimensional data [Yuan, Zhang, Shi et al. (2019)]. Fig. 4 shows the model diagram of the autoencoder. We can see that the network structure is composed of two parts: One is the data encoder represented by the function $h = f(Wx + b)$, and the other is the decoder for data generation and reconstruction by the function $x = g(W^T h + b)$. We used the unsupervised learning algorithm to optimize the constraint weight matrix W and reconstructed the weight matrix W^T to minimize the error between the input and

output of the model, which makes $x^{(i)} = x'^{(i)}$. In this work, we used the autoencoder with constraints on sparse regular terms to limit on the number of features extracted from data and complete the process of data dimension reduction. The sparse autoencoder was obtained by adding L1 regular term constraints to the fundamental loss function to achieve the effective extraction of features. The specific algorithm steps are shown in Algorithm 1.

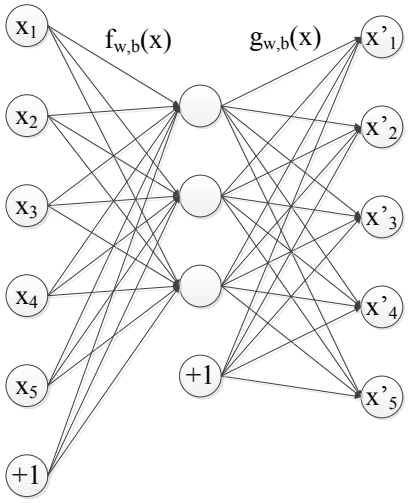

Figure 4: Simplified auto-encoder model

Algorithm 1 Using the sparse autoencoder to extract the data features

Input:$\boldsymbol{x_i}$(i=1, 2, …, n)
Output: $\boldsymbol{x_i'}$(i=1, 2, …, n)
Note: $\boldsymbol{t_i}$ is the training times of the sparse auto-encoder, $\boldsymbol{T_i}$ is the total number of times to train, and $\boldsymbol{s_l}$ is the node number of layer l.
1: Coding:$\boldsymbol{h = Wx + b}$
2: While $\boldsymbol{t < T}$:
3: **While** $\boldsymbol{i < n}$:
4: $\Delta \boldsymbol{W^l} = \Delta \boldsymbol{W^l} + \nabla \boldsymbol{W^l} \cdot [\boldsymbol{J(W, b)} + \boldsymbol{\beta} \cdot \sum_{j=1}^{s_2} \boldsymbol{KL(\rho||\hat{\rho})}]$
where $\boldsymbol{J(W, b)}$ is the loss function of the traditional autoencoder, and the corresponding expression is
$\boldsymbol{J(W,b)} = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{1}{2} \cdot \left\| \boldsymbol{h_{W,b}(x^i)} - \boldsymbol{y^i} \right\|^2\right) + \frac{\lambda}{2} \cdot \sum_{i=1}^{s_l - 1} \sum_{j=1}^{s_l} \boldsymbol{W_{ij}^2}$, the second term in the expression, is a regular term, which can effectively avoid overfitting. However, due to the sparse encoder used in this study, the sparse constraints $\boldsymbol{\beta} \cdot \sum_{j=1}^{s_2} \boldsymbol{KL(\rho||\hat{\rho})}]$ were added to the right side of the equation. Using the KL distance to measure the difference between codes, the corresponding expression is as follows: $\boldsymbol{KL(\rho||\hat{\rho})} = \boldsymbol{\rho} \log \frac{\boldsymbol{\rho}}{\hat{\boldsymbol{\rho}}} + (\boldsymbol{1} - \boldsymbol{\rho}) \log \frac{\boldsymbol{1-\rho}}{\boldsymbol{1-\hat{\rho}}}$, where $\hat{\boldsymbol{\rho}} = \frac{1}{m} \cdot \sum_{i=1}^{n} [\boldsymbol{a_j^2} \cdot \boldsymbol{x^{(i)}}]$ is the average value of the output of the hidden layer node, and $\boldsymbol{a_j^2}$ is the activation of the input vector $\boldsymbol{x^{(i)}}$ to the hidden layer unit of j.
5: $\Delta \boldsymbol{b^l} = \Delta \boldsymbol{b^l} + \nabla \boldsymbol{b^l} \cdot [\boldsymbol{J(W, b)} + \boldsymbol{\beta} \cdot \sum_{j=1}^{s_2} \boldsymbol{KL(\rho||\hat{\rho})}]$
6: $\boldsymbol{W^l} = \boldsymbol{W^l} - \boldsymbol{\alpha}[\left(\frac{1}{m} \cdot \Delta \boldsymbol{W^l}\right) + \boldsymbol{\lambda W^l}]$
7: $\boldsymbol{b^l} = \boldsymbol{b^l} - \boldsymbol{\alpha}[\left(\frac{1}{m} \cdot \Delta \boldsymbol{b^l}\right)]$
8: $\boldsymbol{i = i + 1, t = t + 1}$
9: Decoding: $\mathbf{x' = g(W^T h + b)}$

After using Algorithm 1 to perform multiple iterative trainings on the autoencoder, the optimal constraint weight matrix and reconstruction matrix were obtained. Using these 2 matrices minimizes the loss error between the data obtained after the dimension reduction and the original data. The low-dimensional representation of high-dimensional data features are obtained, and these data are used as the input data for subsequent recurrent neural network classifiers.

The recurrent neural network is usually composed of three parts: input unit, hidden unit, and output unit. The essential aspect of the model is a one-way flow process from the input unit to the hidden layer unit, and then to the output unit after inputting the relevant time series data. In the hidden layer unit, the RNN has stored the data status information from the previous time. When the current information data stream enters the relevant hidden layer unit, the hidden layer unit can use the mixed calculation operation of the current data stream and the previously saved digital data stream to obtain the behavior state that may occur in the next state. Hence, we usually regard the hidden layer unit in the network as the storage in the whole network structure. It is used to store the state data of the previous part of the behavior and to calculate the state data of the next behavior. The corresponding network structure diagram of the current network is shown in Fig. 5.

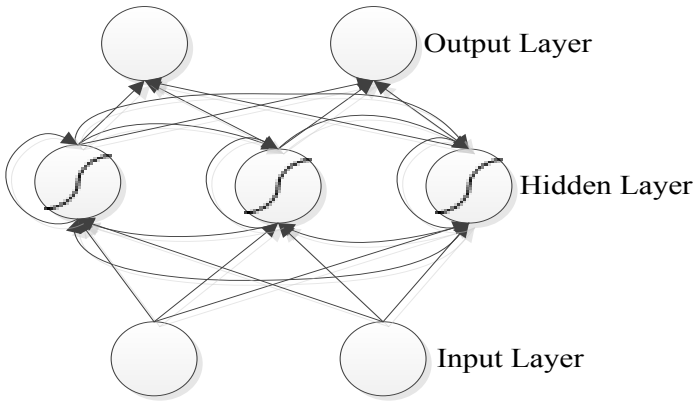

Figure 5: Folded recurrent neural network structure

From Fig. 5, we can see that introducing a ring structure helps "remember" the previous relevant information and apply it to the current output calculation. The structure of the RNN is different from a traditional CNN. The sequence result calculated by the corresponding hidden layer unit in the current layer is related to the output result of the hidden layer unit in the previous layer, and the neurons between each hidden layer unit have a specific information exchange process. We used the advanced autoencoder designed in the previous section to reduce the data dimensions to complete the learning of the corresponding features. Then we used the recoded data to train the RNN model and then used the relevant test data set to evaluate the accuracy of the model.

From Fig. 3, we can also see that our RNN training process is divided into two corresponding stages: forward propagation training and back-propagation training. In this research, we used the BPTT algorithm to complete this process. Forward propagation is responsible for calculating the predicted value of samples under the corresponding weight

matrix for a given sample. In contrast, back-propagation updates the relevant weight matrix using the differential to calculate the accumulated residual.

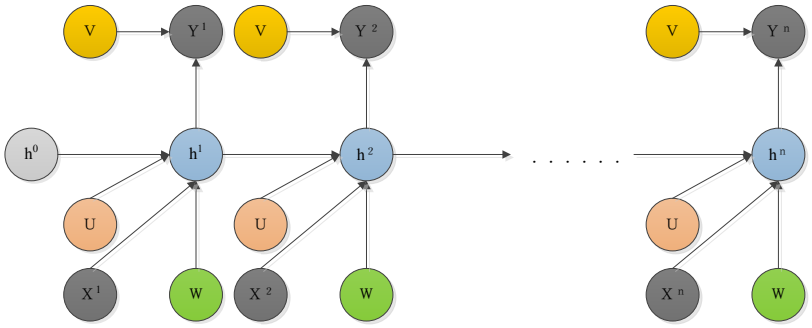

Figure 6: Unfolded recurrent neural network structure

Fig. 6 is a complete unfolded structure of the RNN. We separated the structure of the standard recurrent neural networks shown in Fig. 6 into the elemental composition of the following three elements: (1) A given series of training samples x_i (where i=1, 2, …, n); (2) the hidden layer state unit sequence h_i of the corresponding layer (where i=1, 2, …, n); and (3) a series of predicted output values y_i (where i=1, 2, …, n). The other relevant parameters in the structure that participate in the calculation are as follows. U is the connection weight between the hidden layer unit at the previous time and the hidden layer unit at the current time. V is the link weight between the hidden layer unit of the corresponding layer and the output layer unit. W is the connection weight between the corresponding layer input unit and the hidden layer unit. For the RNN shown in Fig. 6, we used the BPTT algorithm to complete the corresponding training process. The specific operation process is shown in Algorithms 2 and 3. We used the following objective function to evaluate the loss on each of the input RNN model training samples (x_i, y_i): $f(\theta) = L(y_i, \hat{y}_1)$ [Martens and Sutskever (2011)], where L can evaluate the actual deviation distance value between the actual label y_i and the predicted value label $\hat{y}_1$. The function used to evaluate the loss used was the cross-entropy function $L_t = \sum_{i=1}^{i=n} -\hat{y}_1^T \log(y_t)$.

Algorithm 2 Forward Propagation Algorithm

Input: $\mathbf{x_i}$ (i =1, 2, …, n)
Output: $\hat{\boldsymbol{y}}_\mathbf{1}$
1: for i form 1 to n do:
2: $\boldsymbol{s_i} = \boldsymbol{U h_{i-1}} + \boldsymbol{w_{x_i}} + \boldsymbol{b}$
3: $\boldsymbol{h_i} = \boldsymbol{\sigma}(\boldsymbol{s_i})$ $\boldsymbol{\sigma}$ is the hyperbolic tangent activation function used in this study
4: $\boldsymbol{z_i} = \boldsymbol{V h_i} + \boldsymbol{c}$
5: $\boldsymbol{y_i} = \boldsymbol{softmax}(\boldsymbol{z_i})$

Algorithm 3 Back-Propagation Through Time

Input:$< \boldsymbol{y_i}, \widehat{\boldsymbol{y}}_{\boldsymbol{i}} >$(i=1, 2, …, n)
Output:$\boldsymbol{\theta} = \{\boldsymbol{W'}, \boldsymbol{U'}, \boldsymbol{V'}, \boldsymbol{b'}, \boldsymbol{c'}\}$
1: for i from 1 to n do:
2: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{V}} = \frac{\partial \boldsymbol{L_{ti}}}{\partial \boldsymbol{V}} = \frac{\partial \boldsymbol{L_t}}{\partial \boldsymbol{z_i}} \cdot \frac{\partial \boldsymbol{z_i}}{\partial \boldsymbol{V}} = (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) \cdot (\boldsymbol{h_t})^{\boldsymbol{T}}$
3: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{c}} = \frac{\partial \boldsymbol{L_t}}{\partial \boldsymbol{c}} = \frac{\partial \boldsymbol{L_i}}{\partial \boldsymbol{z_i}} \cdot \frac{\partial \boldsymbol{z_i}}{\partial \boldsymbol{c}} = \widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_t}$
4: $\boldsymbol{V'} = \boldsymbol{V} - \boldsymbol{\theta} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) \cdot (\boldsymbol{h_t})^{\boldsymbol{T}}$
5: $\boldsymbol{c'} = \boldsymbol{c} - \boldsymbol{\theta} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_t}$
However, although the parameters W, U, and b are shared, they not only contribute to the
output of the time of t but also contribute to the input $\boldsymbol{s_{t+1}}$ of the hidden layer at the time of t+1. Therefore, when deriving the parameters W, U, and b, we need to start the derivation step-by-step from the back.
6: for i from 1 to n do:
7: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{h_i}} \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}} = \left(\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{z_i}} \cdot \frac{\partial \boldsymbol{z_i}}{\partial \boldsymbol{h_t}} + \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{h_{i+1}}} \cdot \frac{\partial \boldsymbol{h_{i+1}}}{\partial \boldsymbol{h_i}}\right) \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}}$

$= \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T \delta^{i+1}} \odot \boldsymbol{\sigma'}(\boldsymbol{z_{i+1}})\right] \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}}$

$= \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}(\boldsymbol{\delta^{i+1}}) \cdot \boldsymbol{\sigma'}(\boldsymbol{z_{i+1}})\right] \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}}$

$= \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}(\boldsymbol{\sigma'}(\boldsymbol{z_{i+1}})) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}}$

$= \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}(\boldsymbol{1 - h_{i+1}^2}) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{W}}$

$= \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot (\boldsymbol{x_i})^{\boldsymbol{T}}$

8: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{b}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{h_i}} \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{b}} = \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right]$
9: $\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{U}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{h_i}} \cdot \frac{\partial \boldsymbol{h_i}}{\partial \boldsymbol{s_i}} \cdot \frac{\partial \boldsymbol{s_i}}{\partial \boldsymbol{U}}$

$= \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot (\boldsymbol{h_{i-1}})^{\boldsymbol{2}}$

10: $\boldsymbol{W'} = \boldsymbol{W} - \boldsymbol{\theta} \cdot \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot (\boldsymbol{x_i})^{\boldsymbol{T}}$
11: $\boldsymbol{U'} = \boldsymbol{U} - \boldsymbol{\theta} \cdot \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right] \cdot (\boldsymbol{h_{i-1}})^{\boldsymbol{2}}$
12: $\boldsymbol{b'} = \boldsymbol{b} - \boldsymbol{\theta} \cdot \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{diag}(\boldsymbol{1} - (\boldsymbol{h_i})^{\boldsymbol{2}}) \cdot \left[\boldsymbol{V^T} \cdot (\widehat{\boldsymbol{y}}_{\boldsymbol{t}} - \boldsymbol{y_i}) + \boldsymbol{U^T diag}\left(\boldsymbol{1 - h_{i+1}^2}\right) \cdot \boldsymbol{\delta^{i+1}}\right]$

4 Simulation results and discussion

4.1 Data set description

In order to verify that the model proposed in this study can achieve the efficient detection of vehicle behavior, we cooperated with a domestic automobile information security laboratory and carried out the simulation experiments on real intelligent vehicles. There are usually two ways to collect the internal parameters of the vehicle: 1. Passively monitor the data on the Electronic Control Suspension (ECS) network through the OBD-‖ diagnosis interface or obtain the data from the car through the OBD diagnosis interface using the standard communication protocol. 2. By using the CAN converter to access the vehicle CAN bus to complete the monitoring of the vehicle CAN bus.

We chose the second approach to collect data by connecting the USB-to-CAN converter directly to the OBD-‖. There are two reasons: 1. After directly using the USB-to-CAN converter to access CAN bus, we could easily collect the required data by passive monitoring. 2. When obtaining data through the OBD port, it is often necessary to send

request parameters to obtain the corresponding data, which is a complicated process. Through statistical analysis, we found the propagation of various essential parameters on the bus is shown in Tab. 1. These parameters are sent when the car is working normally without any request process. After weighing, we chose to obtain data on the Engine Control Module (ECM) bus, because there are many important parameters of the sender on this bus. For example, the engine speed is directly calculated by the ECU and shared with other modules through this bus.

Table 1: Broadcast of several important parameters on each bus

	Parameter	Speed	Revolutions Per minute	Accelerator Pedal Position	Intake Pressure	Brake Pedal Status	Steering Angle	Gear Position
Bus Category	Motor Meters	o	o	x	x	o	o	o
	Convenience System	o	o	x	x	o	x	x
	Power System	o	o	o	o	o	x	x
	Engine Control Module	o	o	o	o	o	x	o
	Electronic Stability Controller	o	o	o	x	o	o	o

Fig. 7 shows the environment configuration of the data collection from connecting the acquisition adapter to the bus where the engine ECU is located (Fig. 8). The adopted CAN acquisition equipment was a KvaserCAN Leaf LightV2 and the software for the data collection on the computer was the open-source software Vehicle Spy. We collected nearly 300,000 pieces of data related to the vehicle. We used Vehicle Spy to generate the corresponding data log file for the data collected on the CAN bus. Part of the data sample we collected is shown in Fig. 9. It can be seen from the figure that the collected data consist of a timestamp, data domain, data length, arbitration domain, description domain, and vehicle network status information. Because the arbitration domain and some other parameters involve confidential information being sent between the experimental center and the automobile manufacturer, we blurred some regions in the graph

Figure 7: The vehicle used for data acquisition in the experiment

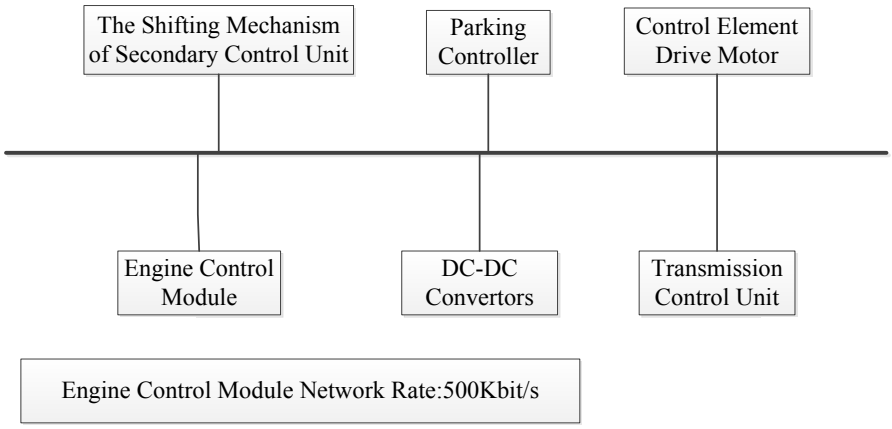

Figure 8: Architecture of the engine's CAN bus

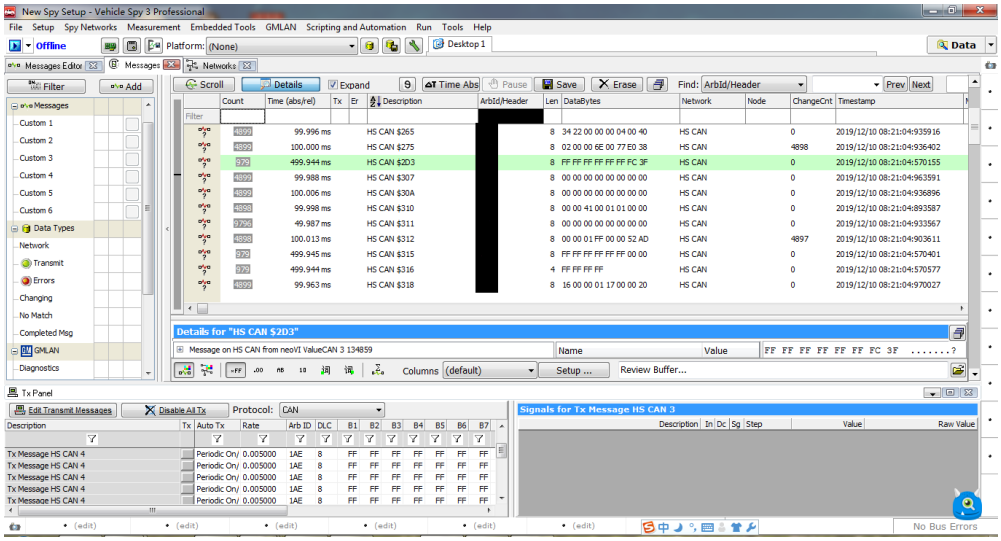

Figure 9: Sample data collected by vehicle spy

4.2 Data pre-processing

4.2.1 Data extraction and calibration

We used the sklearn package in Python to initialize the data. It can be seen from Fig. 10 that most of the data transmitted on the engine's CAN bus were transmitted in the basic format. Messages with different IDSs send 1 or more parameters. For example, in the 8 byte data of a CAN data package, bytes 1 and 2 represent the high 8 bits and low 8 bits of data information, respectively, while bytes 3 and 4 represent the high 8 bits and low 8 bits of speed, respectively. Moreover, the range of these data may be from 0×0000 to 0×FFFF. Therefore, in order to get the real speed or other parameters, we needed to convert them, generally through Eq. (1). V is the actual value. X is the value transmitted through the can packet. B is the deviation.

$$v = ax + b \tag{1}$$

```
***PROTOCOL CAN***
***NOTE: PLEASE DO NOT EDIT THIS DOCUMENT***
***[START LOGGING SESSION]***
[redacted]
***HEX***
***SYSTEM MODE***
***START CHANNEL BAUD RATE***
***CHANNEL 1 - Kvaser - Kvaser Leaf Light v2 #0 (Channel 0), Serial Number- 0,
***END CHANNEL BAUD RATE***
***START DATABASE FILES (DBF/DBC)******START DATABASE FILES (DBF/DBC)***
***C:\Users\Administrator\Desktop[redacted]CAN DATABASE.DBF***
***END OF DATABASE FILES (DBF/DBC)******END OF DATABASE FILES (DBF/DBC)***
***<Time><Tx/Rx><Channel><CAN ID><Type><DLC><DataBytes>***
22:42:21:2045 Rx 1 [redacted] s 8 00 00 28 0A D1 AF FE 4F
22:42:21:2065 Rx 1 [redacted] s 8 12 3F 64 E0 77 33 00 00
22:42:21:2075 Rx 1 [redacted] s 8 0C 0C 4B 00 A0 FF 03 67
22:42:21:2075 Rx 1 [redacted] s 8 0A 00 C7 0F 0F 00 0D 1E
22:42:21:2075 Rx 1 [redacted] s 8 00 02 00 00 08 00 0C E9
22:42:21:2085 Rx 1 [redacted] s 8 00 40 04 06 FC FF FF BB
22:42:21:2095 Rx 1 [redacted] s 8 00 FF 29 74 FF FF 0E 57
22:42:21:2115 Rx 1 [redacted] s 8 20 00 0F 20 4E 00 1E 44
22:42:21:2125 Rx 1 [redacted] s 8 FF FF 5E 00 F8 00 3F 6C
22:42:21:2135 Rx 1 [redacted] s 8 C0 14 F1 0C 00 00 14 2A
```

Figure 10: Format of the vehicle spy log file

The parameters acquired by Vehicle Spy could not be directly sent to the network for training. In order to get the parameters that we needed, we also needed to carry out the corresponding data analysis process. We used the script written in the sklearn package in Python to extract the parameters of the CAN log of Vehicle Spy and converted them into the CSV files, as shown in Fig. 11.

time_in_ms	Msg_ID	AccPedal	RPM	Speed	Throttle	IntakePressure	AirFlowRate
380504.9	577	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380504.9	577	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380508.9	527	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380509.9	269	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380518.9	527	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380519.9	269	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380524.9	577	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380529.9	527	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380529.9	269	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380538.9	577	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380538.9	527	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380539.9	269	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380545.9	577	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380548.9	527	0.021255	0.02149	0.023041	0.169643	0.000419287	0.096
380549.9	269	0.021773	0.02149	0.023041	0.169643	0.000419287	0.096

Figure 11: Data after pre-processing

4.2.2 Data normalization

The parameters in Fig. 11 (after column 3) vary from 0 to 1, which is because these parameters vary in different ranges. The standard method is data normalization. The purpose of data normalization is to standardize the data of different dimensions and units to solve the differences between the data indicators. After normalization, different pieces of data are at the same level, which is convenient for comprehensive comparative evaluation.

There are many ways to normalize, such as the Z-score (Eq. (2)) and Min-Max (Eq. (3)):

$$X^* = \frac{x-\mu}{\sigma} \tag{2}$$

$$X^* = \frac{x-min}{max-min} \tag{3}$$

Because there was no special requirement for this application scenario, the Min-Max normalization method was selected because this method is the simplest. It is worth noting that when the Min-Max method is used to normalize the data, it is necessary to use the unified fixed maximum and minimum values. Otherwise, the prediction is not accurate due to the difference between the two values in the experiment.

4.2.3 Data interpolation

Because these parameters were transmitted serially via the CAN datagram, as shown in Fig. 11, data interpolation was also needed. There are many interpolation methods. We chose the forward interpolation method, as shown in Fig. 12. The horizontal axis in the figure is the time, and the vertical axis is the parameter. The italicized and underlined values in the figure show the actual received parameter value, and the rest are the values that were inserted using the forward interpolation method.

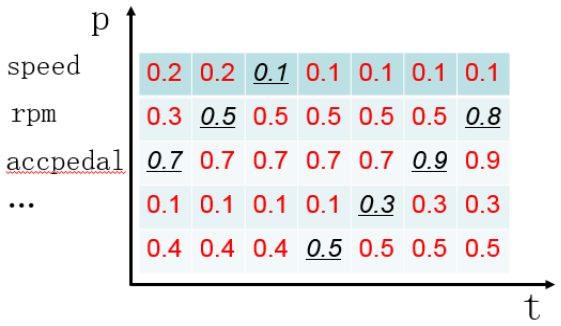

Figure 12: Sample data interpolation

4.2.4 Data sampling

After analysis, we found that the received data had much redundancy after interpolation. The redundant data were due to the interpolation of the data. The channel arbitration mechanism of CAN had a certain randomness, so the received data were not evenly distributed in time. For these reasons, the data needed to be sampled. The sampling method adopted here was to round to the nearest 10 milliseconds, that is, each column of data was sampled in milliseconds. When sampling, the data closest to the whole number 10 was taken, so the amount of training data could be significantly reduced.

4.3 Data characteristics and correlation analysis

The characteristics of the data in the ICV and the corresponding correlation analysis are usually completed in the following two ways. (1) Analyze the data flow generated by the car under normal driving conditions. For example, the analysis process is completed by analyzing the data packet transmission frequency, state change rate, and CAN bus

utilization rate under normal driving conditions. (2) With the help of the working principles of the car, the parameters of the car in different states and the correlations between them are analyzed (such as the relationships among the RPM (Revolutions Per Minute) and the speed of the car as well as the air intake of the car under the normal driving conditions). In the experiment, we used the Pearson correlation coefficient to analyze the collected data and find a group of data with the strongest correlation. As shown in Fig. 13, the overall trend of these data is relevant.

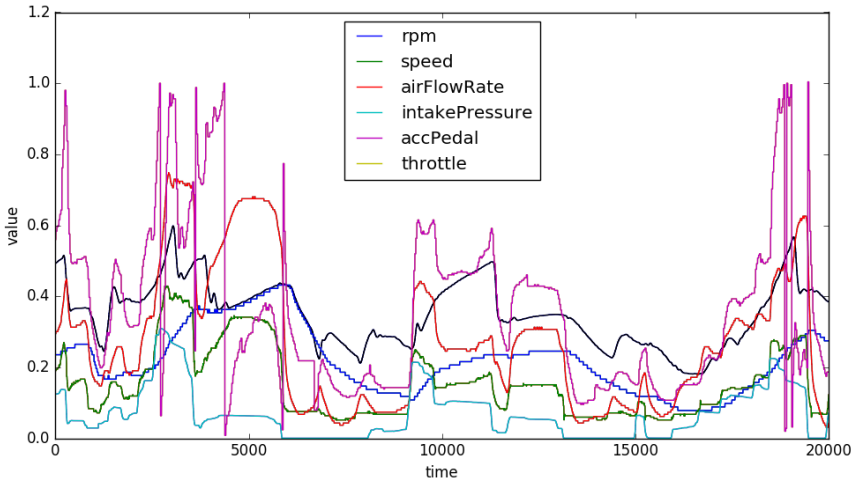

Figure 13: Changing trend of the automobile parameters

Through analysis, we can usually divide the data transmitted on a CAN bus into two categories according to the apparent characteristic relationships: the data with obvious mechanical rules, such as the pedal position of the car and the speed and acceleration of the car. Or the data without apparent rules, such as the information of the air intake of the engine and the status of the brake pedal. The reason these data do not have transparent characteristic relationships is that the set parameters corresponding to these data need to be adjusted by the driver according to the road conditions in real time.

Different vehicle parameters with corresponding relationships will also present a corresponding normal threshold range under certain normal driving conditions, such as the driving speed of the vehicle, the automatic gear of the vehicle, and the speed of the vehicle engine under certain driving conditions. Their change rate is the upper limit of the average threshold range. Fig. 14 shows that there is a limited change rate between the speed of the car and the speed of the engine and the gear speed under normal driving conditions. However, when we tried to implement a replay attack or forge the driving state data to the vehicle CAN bus using Vehicle Spy, the change rate and the corresponding parameter threshold range among the three also changed. As shown in Fig. 15, the standard speed information and the replayed speed information are mixed due to the replaying attack on the engine speed of the vehicle, so that may lead to the phenomenon that the waveform of the image vibrates. As a result, we can speculate that at the inflexion point of the curve where the oscillation occurs is when the abnormality occurs. Although we can see from the figure that the replay speed does not lead to significant changes in other parameters, we

cannot deny that the replay attacks or a forgery attack will not affect other parameters. When we replay the corresponding abnormal data at a higher frequency, we can easily see a black curve by setting the low-pass filter at the corresponding receiver, as shown in Fig. 16. When the engine speed is replayed at a higher frequency, the corresponding vehicle speed has a breakpoint, which also has a particular impact on the vehicle speed.

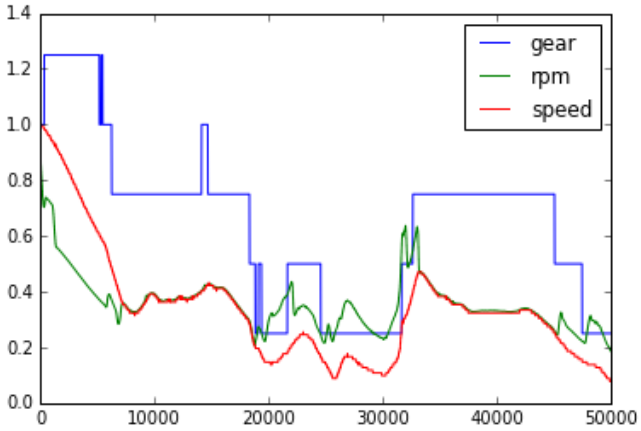

Figure 14: Under normal driving conditions, the relationship among the driving speed, gear position, and RPM of the vehicle

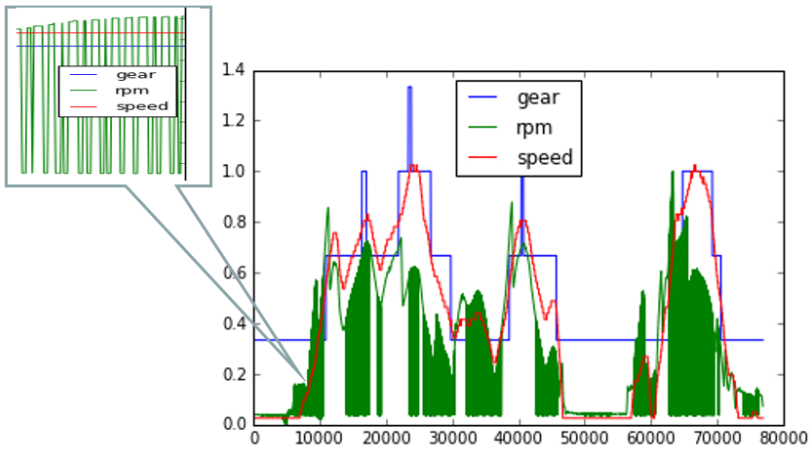

Figure 15: During the replay attack: the relationship among the driving speed, gear position, and RPM of the vehicle

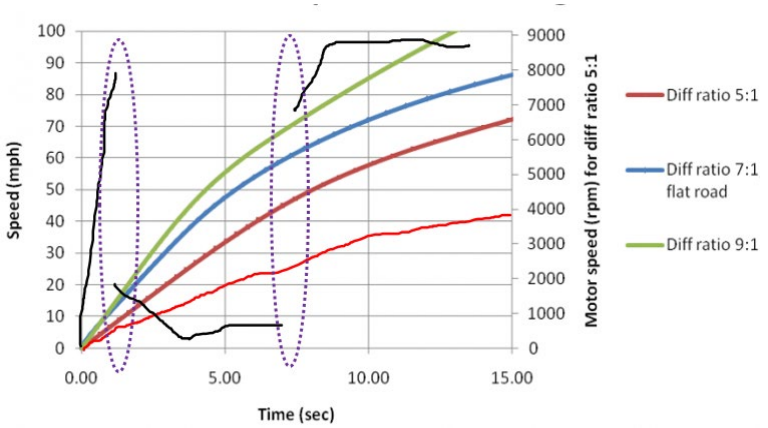

Figure 16: Change process of the vehicle speed after sending abnormal data at different frequencies

4.4 Feature selection

After analyzing the strong correlations of the data in Section C, we selected several auto parameters with a strong correlation as the data vectors of the whole model, such as the speed of the car, engine speed, engine intake pressure, automobile accelerator pedal position, and automatic transmission gear. The time correlation of these parameters and their relationship with each other have definite characteristics. In order to verify that there is a strong correlation between the parameters we selected, we chose Eq. (4) to calculate the covariance coefficient of the selected parameters, as shown in Tab. 2. We can see that these parameters are positively correlated:

$$cov(X,Y) = \frac{\sum_{i-1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{n-1} \tag{4}$$

Table 2: Covariance matrix of selected parameters

	RPM	Speed	MAP	MAF	AccPedal	Throttle
RPM	1	0.793538	0.525027	0.313592	0.438199	0.565193
Speed	0.793538	1	0.495519	0.365531	0.215287	0.494654
MAP	0.525027	0.495519	1	0.832216	0.654406	0.484693
MAF	0.313592	0.365531	0.832216	1	0.496407	0.553881
AccPedal	0.438199	0.215287	0.654406	0.496407	1	0.38837
Throttle	0.565193	0.494654	0.484693	0.553881	0.38837	1

4.5 Experiment evaluation

The purpose of this research was to improve the detection efficiency using the sparse feature-based autoencoder and the recurrent neural network, respectively, and improve the

convergence speed of the whole network using the BPTT. We used the alarm rate and false alarm rate to evaluate the overall performance of the proposed model. The corresponding alarm rate and false alarm rate were calculated using Eqs. (5) and (6), respectively. The true positive (TP) is the number of records identified as abnormal, false positive (FP) is the number of records identified as normal, true negative (TN) is the number of records identified as normal, and false negative (FN) is the number of records identified as abnormal:

$$TPR = \frac{TP}{TP+FN} * 100\% \tag{5}$$

$$FPR = \frac{FP}{FP+TN} * 100\% \tag{6}$$

We used the currently popular deep learning framework Keras to complete the training process of the model. We completed the corresponding experiment on a laptop. The experimental configuration included an ASUS fl8000u, core i7-8550u CPU, 16 GB memory, and a GPU that was not used for the acceleration process. In order to compare the scheme proposed in this study with the machine learning method [Medhat, Ramadan and Talkhan (2015); Ronak, Ganesh, Akshay et al. (2016); Peraković, Periša, Cvitić et al. (2017); Pavani and Damodaram (2013)], we also designed the corresponding comparative experiments to compare the schemes.

Table 3: Accuracy rate and convergence time of the model for different learning rates and hidden layer nodes

	Train Sets	Test Sets	Convergence Time (s)	Detection time on Test Sets (ms)
Hidden Nodes=40 Learning Rate=0.01	95.20%	87.24%	5340	6.47
Hidden Nodes=40 Learning Rate=0.05	95.47%	89.74%	5041	6.12
Hidden Nodes=40 Learning Rate=0.1	96.95%	89.92%	4885	5.94
Hidden Nodes=60 Learning Rate=0.01	96.32%	83.25%	4991	6.03
Hidden Nodes=60 Learning Rate=0.05	97.64%	85.96%	4764	5.76
Hidden Nodes=60 Learning Rate=0.1	98.17%	86.43%	4683	5.21
Hidden Nodes=80 Learning Rate=0.01	98.52%	89.94%	4954	5.84
Hidden Nodes=80 Learning Rate=0.05	98.94%	92.37%	4892	5.32
Hidden Nodes=80 Learning Rate=0.1	99.36%	95.84%	5103	4.97
Hidden Nodes=100 Learning Rate=0.01	97.25%	87.67%	5701	6.15
Hidden Nodes=100 Learning Rate=0.05	97.86%	89.54%	5394	5.87
Hidden Nodes=100 Learning Rate=0.1	98.58%	91.21%	5248	5.63

In the experiment, we mapped the 16-dimensional data features to 48-dimensional data features using the one-hot encoding coding technology, which is used as the input of the

autoencoder. Therefore, the neural network classifier in this study has 48 input nodes and 2 output nodes. In order to get a better training process, we set the number of hidden layer nodes in the neural network to 40, 60, 80, and 100 and the learning rate is also setting to 0.01, 0.05, and 0.1 in the training process. Tab. 3 shows the classification accuracy and convergence time of the model under different parameters.

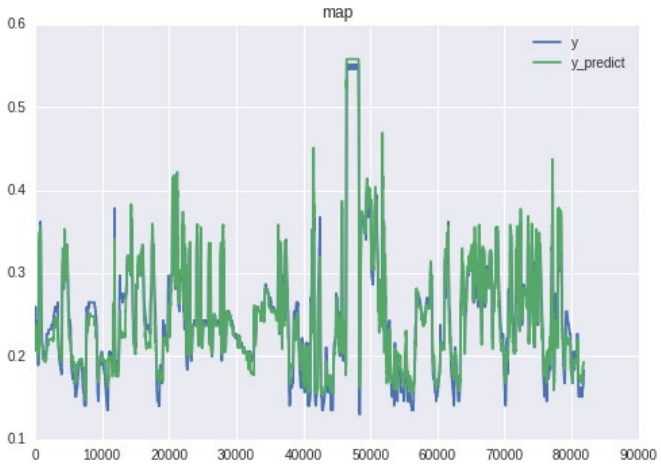

Figure 17: The predicted results of the RPM using the proposed model

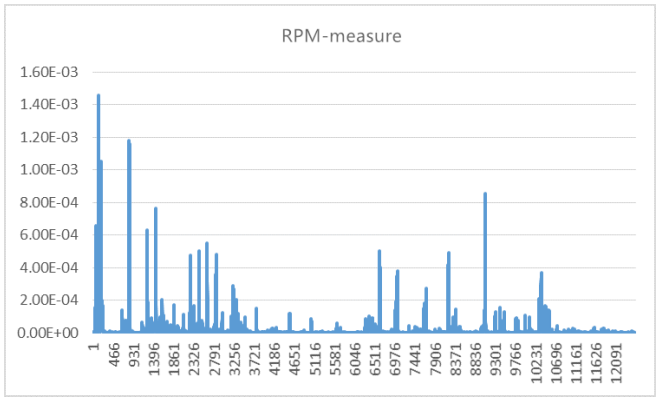

Figure 18: Variance sequence of the RPM between the actual value and predicted value

The experimental results in Tab. 3 show that when the hidden layer node is set to 80 nodes and the learning rate is set to 0.1, and the model has achieved a high accuracy regardless of whether it is the test set or the training set. Although the methods discussed in this study spend more time on the training model, we tried to use the GPU or offline method to carry out the corresponding training process for the model [Yin, Zhu, Fei et al. (2017); Kang and Kang (2016)]. At the same time, Tab. 3 also shows the autonomous feature learning ability of the sparse autoencoder and the consideration of the recurrent neural network for data timing characteristics. We can realize the real-time detection of abnormal data at the millisecond level. The comparison results between the predicted engine speed value and the actual value

using the proposed model are shown in Fig. 17. A small error is achieved between these values, and the accurate prediction of the data is realized. In order to make the error results in Fig. 17 easier to observe, Fig. 18 shows the variance sequence between the actual value and the predicted value about the engine speed. Therefore, using the advantages of deep learning in data set feature extraction and model classification, the model can significantly improve the accuracy of the detection efficiency and achieve a high accuracy and low false alarm rate.

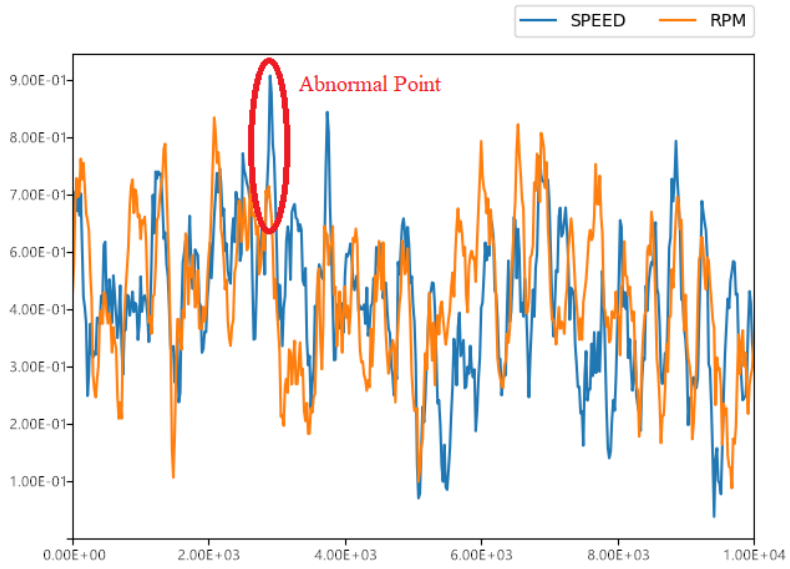

Figure 19: Influence of fake vehicle RPM data on vehicle speed

Fig. 19 shows the signal graph generated when the vehicle driving data pass through the low-pass filter after the forgery attack on the vehicle in the normal driving state. We see that after injecting the RPM forged data into the CAN bus, the calculated results showed they would have an impact on the other vehicle parameters, as well as an abnormal performance on the other parameters' prediction. At this time, we can calculate the variance between the real value and the predicted value as a judgment value index. If the calculated variance is higher than a specific safety-critical value, such as the abnormal point in Fig. 19, we can determine the corresponding abnormal behavior. In this way, no matter which parameter of the vehicle is forged or attacked by the attacker, our model can detect the attack accurately with the help of the mechanical characteristics between the vehicle data. Based on these test results, we can take emergency response measures, such as informing the administrator to control the communication link of the vehicle, so as to ensure the external network security of the vehicle.

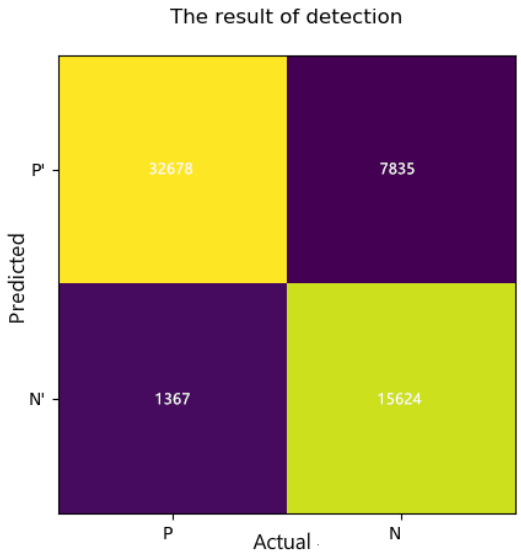

Figure 20: Confusion matrix corresponding to the test results

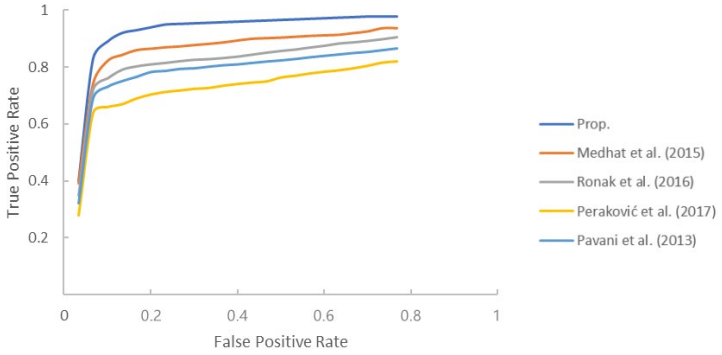

Figure 21: ROC curves for in-vehicle intrusion detection performance

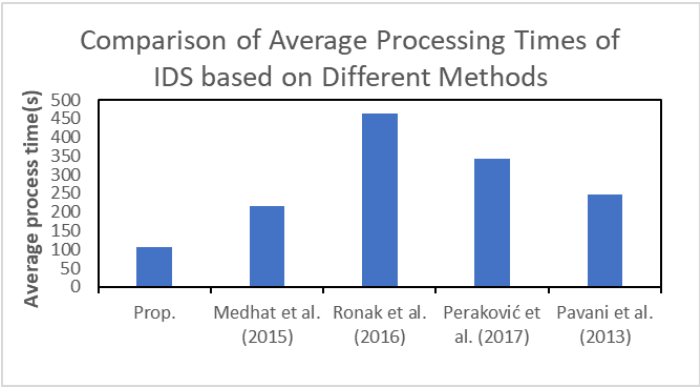

Figure 22: Comparing the average processing time of different IDS models

Fig. 20 shows the confusion matrix of the detection results in terms of the proposed model for the test data set under the optimal parameter conditions. The experimental results show that the accuracy of the model is about 96%. In addition, we compared our IDS models with those in the literature [Medhat, Ramadan and Talkhan (2015); Ronak, Ganesh, Akshay et al. (2016); Peraković, Periša, Cvitić et al. (2017); Pavani and Damodaram (2013)], and the results are shown in Fig. 22. From Fig. 22, we can see that the average processing time of the proposed IDS model is significantly less than the relevant IDS model based on deep learning in several studies [Medhat, Ramadan and Talkhan (2015); Ronak, Ganesh, Akshay et al. (2016); Peraković, Periša, Cvitić et al. (2017); Pavani and Damodaram (2013)]. Similarly, we compared the proposed IDS model with the IDS model proposed in the literature [Medhat, Ramadan and Talkhan (2015); Ronak, Ganesh, Akshay et al. (2016); Peraković, Periša, Cvitić et al. (2017); Pavani and Damodaram (2013)] in terms of the detection efficiency. The experimental results are shown in Fig. 21. From the figure, we can see that the proposed scheme can achieve nearly 96% in terms of the TPR index, while only 2%-3% in FPF, which means that it is achieving a lower false alarm rate. The figure shows that the model proposed in this study achieves a higher detection efficiency and lower false alarm rate compared with the IDS based on deep learning proposed by its predecessors.

5 Conclusion

Although our model has achieved encouraging results, we acknowledge that it is not perfect, and there is room for further improvement. The proportion of the Internet of Vehicles in people's lives will increase with the continuous development and combination of information technology and automotive machinery technology. However, due to the special limitations of the Internet of Vehicles technology and the incompleteness of its existing security technology, government agencies and people of all countries should pay closer attention to the development trend of the Internet of Vehicles security issues. Therefore, based on the analysis of the current security problems of the vehicle network, we propose to use advanced autoencoder and recurrent neural networks to improve the detection rate of abnormal behaviors in the vehicle system. Through experiments that evaluated our proposed model using real ICV data, we found that the model can well classify vehicle behavior and improve the safety of the vehicle system to a large extent. In the future, we hope to further consider the use of other kinds of deep learning technology to ensure the safety of the in-vehicle system to find a more efficient solution, as well as to promote the use of these technologies with artificial intelligence methods in network security. We believe that this work can also improve the efficiency of network security problems.

Funding Statement: This work was supported by Research on the Influences of Network Security Threat Intelligence on Sichuan Government and Enterprises and the Development Countermeasure (Project ID 2018ZR0220), Research on Key Technologies of Network Security Protection in Intelligent Vehicle Based on (Project ID 2018JY0510), the Research on Abnormal Behavior Detection Technology of Automotive CAN Bus Based on Information Entropy (Project ID 2018Z105), the Research on the Training Mechanism of Driverless Network Safety Talents for Sichuan Auto Industry Based on Industry-University Synergy (Project ID 18RKX0667), Research and implementation of traffic cooperative

perception and traffic signal optimization of main road (Project ID 2018YF0500707SN), Research and implementation of intelligent traffic control and monitoring system (Project ID 2019YGG0201), Remote upgrade system of intelligent vehicle software (Project ID 2018GZDZX0011).

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

References

Anzer, A.; Elhadef, M. (2018): Deep learning-based intrusion detection systems for intelligent vehicular ad hoc networks. *Advanced Multimedia and Ubiquitous Engineering*, vol. 518, pp. 109-116.

Banković, Z.; Moya, J. M.; Araujo, Á.; Fraga, D.; Vallejo, J. C. et al. (2010): Distributed intrusion detection system for wireless sensor networks based on a reputation system coupled with kernel self-organizing maps. *Integrated Computer Aided Engineering*, vol. 17, no. 2, pp. 87-102.

Bernardini, C.; Asghar, M. R.; Crispo, B. (2017): Security and privacy in vehicular communications: challenges and opportunities. *Vehicular Communications*, vol. 10, pp. 13-28.

Besson, L.; Leleu, P. (2016): A distributed intrusion detection system for ad-hoc wireless sensor networks: the AWISSENET distributed intrusion detection system. *16th International Conference on Systems*, *Signals and Image Processing*, pp. 1-3.

Checkoway, S.; McCoy, D.; Kantor, B.; Anderson, D.; Shacham, H. et al. (2011): Comprehensive experimental analyses of automotive attack surfaces. *USENIX Security Symposium*, vol. 4, pp. 447-462.

Cho, E. J.; Hong, C. S.; Lee, S.; Jeon, S. (2013): A partially distributed intrusion detection system for wireless sensor networks. *Sensors*, vol. 13, no. 12, pp. 15863-15879.

Contreras-Castillo, J.; Zeadally, S.; Guerrero-Ibañez, J. A. (2017): Internet of vehicles: architecture, protocols, and security. *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701-3709.

Dong, B.; Wang, X. (2016): Comparison deep learning method to traditional methods using for network intrusion detection. *8th IEEE International Conference on Communication Software and Networks*, pp. 581-585.

Feng, Z.; He, M.; Li, B.; Deng, M. (2017): Research progress on key technologies of automobile information security attack and defense. *Journal of Information Security*, vol. 2, no. 2, pp. 1-14.

Gao, L.; Li, F.; Xu, X.; Liu, Y. (2019): Intrusion detection system using SOEKS and deep learning for in-vehicle security. *Cluster Computing*, vol. 22, no. 6, pp. 14721-14729.

Han, M. L.; Kwak, B. I.; Kim, H. K. (2018): Anomaly intrusion detection method for vehicular networks based on survival analysis. *Vehicular communications*, vol. 14, pp. 52-63.

Hou, S.; Saas, A.; Chen, L.; Ye, Y. (2016): Deep4maldroid: A deep learning framework

for android malware detection based on linux kernel system call graphs. *IEEE/WIC/ACM International Conference on Web Intelligence Workshops*, pp. 104-111.

Johansson, K. H.; Törngren, M.; Nielsen, L. (2015): Vehicle applications of controller area network. *Handbook of Networked and Embedded Control Systems*. Birkhäuser Boston, USA.

Kang, M. J.; Kang, J. W. (2016): Intrusion detection system using deep neural network for in-vehicle network security. *PLoS One*, vol. 11, no. 6, pp. 1-17.

Kumar, D. A.; Venugopalan, S. R. (2017): Intrusion detection systems: a review. *International Journal of Advanced Research in Computer Science*, vol. 8, no. 8, pp. 356-370.

Kwon, D.; Kim, H.; Kim, J.; Suh, S. C.; Kim, I. et al. (2017): A survey of deep learning-based network anomaly detection. *Cluster Computing*, vol. 22, pp. 1-13.

Lauf, A. P.; Peters, R. A.; Robinson, W. H. (2010): A distributed intrusion detection system for resource-constrained devices in ad-hoc networks. *Ad Hoc Networks*, vol. 8, no. 3, pp. 253-266.

Leinmüller, T.; Held, A.; Schäfer, G.; Wolisz, A. (2014): Intrusion detection in VANETs. *12th IEEE International Conference on Network Protocols Student Poster Session*, pp. 482-486.

Li, L. (2019): *Research and Implementation of Intrusion Detection Technology for Internet of Vehicle based on Machine Learning (M.S. thesis)*. University of Electronic Science and Technology of China, China.

Li, X.; Yang, B. (2015): Analysis of safety protection in vehicle networking. *Mobile Communication*, vol. 39, no. 11, pp. 32-35.

Li, X.; Zhong, C.; Chen, Y.; Zhang, H.; Weng, J. (2019): Survey of internet of vehicles security. *Journal of Cyber Security*, vol. 4, no. 3, pp. 17-33.

Liang, J. (2017): *Application of Intrusion Detection System in Vehicle Networking (M.S. Thesis)*. Shenzhen University, China.

Liu, Y.; Li, Y.; Man, H. (2015): Short paper: a distributed cross-layer intrusion detection system for ad hoc networks. *First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pp. 418-420.

Martens, J.; Sutskever, I. (2016): Learning recurrent neural networks with hessian-free optimization. *28th International Conference on Machine Learning*, pp. 1033-1040.

Martinelli, F.; Mercaldo, F.; Orlando, A.; Nardone, V.; Santone, A. et al. (2018): Human behavior characterization for driving style recognition in vehicle system. *Computers & Electrical Engineering*, vol. 4, no. 31, pp. 1-14.

Medhat, K.; Ramadan, R. A.; Talkhan, I. (2015): Distributed intrusion detection system for wireless sensor networks. *9th International Conference on Next Generation Mobile Applications*, *Services and Technologies*, pp. 234-239.

Mershad, K.; Artail, H. (2012): A framework for secure and efficient data acquisition in vehicular ad hoc networks. *IEEE Transactions on vehicular technology*, vol. 62, no. 2, pp. 536-551.

Miller, C.; Valasek, C. (2015): Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, vol. 91, pp. 1-93.

Mohammadi, S.; Namadchian, A. (2017): A new deep learning approach for anomaly base IDS using memetic classifier. *International Journal of Computers Communications & Control*, vol. 12, no. 5, pp. 677-688.

Pavani, K.; Damodaram, A. (2013): Intrusion detection using MLP for MANETs. *Third International Conference on Computational Intelligence and Information Technology*, pp. 440-444.

Peraković, D.; Periša, M.; Cvitić, I.; Husnjak, S. (2017): Model for detection and classification of DDoS traffic based on artificial neural network. *Telfor Journal*, vol. 9, no. 1, pp. 26.

Qiao, H.; Li, G.; Chen, Y. (2018): Research on key technologies of vehicle networking system architecture. *Electronic Production*, vol. 352, no. 11.

Ronak, A.; Ganesh, T.; Akshay, C.; Shriganesh, M.; Deeplakshmi, Z. (2016): Distributed intrusion detection system using bayesian learning and apache mahout. *Journal of Advanced Research in Computer and Communication Engineering*, vol. 5, no. 1, pp. 361-364.

Samad, A.; Alam, S.; Mohammed, S.; Bhukhari, M. U. (2018): Internet of vehicles (IoV) requirements, attacks and countermeasures. *5th International Conference on "Computing for Sustainable Global Development"*, pp. 4037-4040.

Sedjelmaci, H.; Senouci, S. M.; Abu-Rgheff, M. A. (2014): An efficient and lightweight intrusion detection mechanism for service-oriented vehicular networks. *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 570-577.

Seo, E.; Song, H. M.; Kim, H. K. (2018): GIDS: GAN based intrusion detection system for in-vehicle network. *16th Annual Conference on Privacy, Security and Trust*, pp. 1-6.

Sun, X.; Yan, B.; Zhang, X.; Rong, C. (2015): An integrated intrusion detection model of cluster-based wireless sensor network. *PLoS One*, vol. 10, no. 10, pp. 1-16.

Tan, L.; Li, C.; Xia, J.; Cao, J. (2019): Application of self-organizing feature map neural network based on k-means clustering in network intrusion detection. *Computers, Materials & Continua*, vol. 61, no. 1, pp. 275-288.

Tariq, U. (2019): Intrusion detection and anticipation system (IDAS) for IEEE 802.15.4 devices. *Intelligent Automation and Soft Computing*, vol. 25, no. 2, pp. 231-242.

Taylor, A.; Leblanc, S.; Japkowicz, N. (2016): Anomaly detection in automobile control network data with long short-term memory networks. *IEEE International Conference on Data Science and Advanced Analytics*, pp. 130-139.

Tyagi, P.; Dembla, D. (2014): Investigating the security threats in vehicular ad hoc networks (VANETs): towards security engineering for safer on-road transportation. *International Conference on Advances in Computing, Communications and Informatics*, pp. 2084-2090.

Unluturk, M. S.; Oguz, K.; Atay, C. (2015): Emotion recognition using neural networks. *10th WSEAS International Conference on Neural Networks*, Prague, Czech Republic, pp. 82-85.

Woo, S.; Jo, H. J.; Lee, D. H. (2014): A practical wireless attack on the connected car and security protocol for in-vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 1-14.

Yang, L.; Wu, Y.; Wang, J.; Liu, Y. (2018): Research on recurrent neural network. *Journal of Computer Applications*, vol. 38, no. s2, pp. 6-11, 31.

Yin, C.; Zhu, Y.; Fei, J.; He, X. (2017): A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, vol. 5, pp. 21954-21961.

Yuan, F.; Zhang, L.; Shi, J.; Xia, X.; Li, G. (2019): Theories and Applications of Auto-Encoder Neural Networks: a literature survey. *Chinese Journal of Computers*, vol. 42, no. 1, pp. 203-230.

Zhao, R.; Yan, R.; Chen, Z.; Mao, K.; Wang, P. et al. (2019): Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, vol. 115, pp. 213-237.